Spezifikation von Gestensteuerung

Ein hybrider Ansatz

Dissertation
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)
durch die Fakultät für Wirtschaftswissenschaften
der Universität Duisburg-Essen, Campus Essen

vorgelegt von
Marc Hesenius, M.Sc.

Essen, 2018

Tag der mündlichen Prüfung: 19.12.2018
1. Gutachter: Prof. Dr. Volker Gruhn
2. Gutachter: Prof. Dr. Matthias Book

Zusammenfassung

Die Interaktion zwischen Mensch und Maschine (engl. *Human-Computer Interaction* (HCI)) entwickelt sich mit hoher Geschwindigkeit weiter, insbesondere durch den Fortschritt in der Sensortechnologie. Vor allem Multi-Touch-Gesten sind in den vergangenen Jahren durch die Verbreitung mobiler Geräte und zahlreicher anderer Geräte (z. B. berührungsempfindliche Tische oder digitale Whiteboards) zum Standard avanciert und haben mittlerweile auch den Weg zurück in die klassische Welt des PCs geschafft. Neue Sensoren ermöglichen auch räumliche Gesten, die vor allem für Anwendungen im Bereich der virtuellen und augmentierten Realität interessant sind: Benutzer können direkt mit virtuellen Objekten interagieren oder Kommandos absetzen, indem sie ihre Hände, Finger oder Werkzeuge wie Magic Wands im dreidimensionalen Raum bewegen.

HCI im Allgemeinen und Gestensteuerung im Besonderen sind außerdem Schlüsselfaktoren, um die individuelle Benutzererfahrung (engl. *User Experience* (UX)) zufriedenstellend zu gestalten. Benutzer erwarten Fehlerfreiheit und ein ansprechendes Design, aber auch ein *gutes Gefühl* – ein eher undefinierbares Merkmal, das abhängig ist von unterschiedlichen Faktoren, aber maßgeblich von HCI beeinflusst wird. Gestensteuerung ist somit wichtig für den wirtschaftlichen Erfolg einer Anwendung und muss bei der Entwicklung explizit betrachtet werden. Angedachte Gesten müssen korrekt definiert und zwischen den unterschiedlichen Projektbeteiligten kommuniziert werden.

Gesten sind jedoch schwierig zu implementieren und Entwickler müssen unterschiedliche Entscheidungen treffen, die nicht ohne Weiteres rückgängig gemacht werden können: Passende *Hardware* zur definierten Interaktion ist ebenso nötig wie *Sensoren*, die die notwendigen Daten liefern, und *Algorithmen*, die in dem Strom der Eingaben die ausgeführten Gesten zuverlässig erkennen – eine komplexe und fehleranfällige Aufgabe.

In der modernen Software-Entwicklung sind interdisziplinäre Teams und agile Methoden heute Standard, in denen die unterschiedlichen Projektbeteiligten – sowohl technische als auch fachliche Spezialisten – ihre jeweiligen Stärken ausspielen und kombinieren können. Skizzen und Prototypen spielen vor allem in frühen Phasen eines Projekts eine wichtige Rolle bei der Arbeit von UI-Designern und Software-Ingenieuren und sind insbesondere für die Entwicklung der Benutzerschnittstelle wichtig: Alle Projektbeteiligten können so schnell eine gemeinsame Vision der Anwendung aufbauen. Allerdings wird in Sketches und Prototypen die Interaktion nicht explizit definiert, sondern nur eine implizite Definition zur Verfügung gestellt, wodurch Missverständnisse und unterschiedliche Interpretationen möglich sind. Die Definition von Gesten ist jedoch wichtig, um die korrekten Entscheidungen für die Implementierung zu treffen, insbesondere wenn individuelle Gesten eingesetzt werden sollen.

Die Grundlage der Gestenbeschreibung sind die unterschiedlichen Eigenschaften, die moderne Sensoren ermöglichen. Neben der Form einer Geste können auch weitere Informationen für HCI verwendet werden, z. B. die Geschwindigkeit oder der Ort der Ausführung. Allerdings kann die optimale Notationsform für verschiedene Eigenschaften unterschiedlich sein. Existierende Notationen zur Beschreibung von Gesten basieren meist auf entweder textuellen oder graphischen Elementen und lassen somit die Möglichkeiten außer Acht, die eine Kombination beider Paradigmen bietet.

Diese Arbeit stellt GestureCards vor, eine hybride Notation zur Beschreibung von Gesten. GestureCards kombinieren graphische und textuelle Elementen, um die unterschiedlichen Eigenschaften von Gesten darzustellen. Die Notation soll zur Spezifikation von Gesten eingesetzt werden und ermöglicht die Beschreibung von Oberflächengesten und räumlichen Gesten. Die verschiedenen Aspekte, die bei der Entwicklung von GestureCards im Vordergrund standen, und die zugrundeliegenden Konzepte werden in mehreren Studien evaluiert. Die Studien zeigen, dass der hybride Ansatz, der für GestureCards gewählt wurde, vorteilhaft bei der Entwicklung gestenbasierter Anwendungen ist und Entwickler beim Verständnis definierter Gesten unterstützt.

Abstract

Human-Computer Interaction (HCI) is evolving quickly with the recent advances in sensor technology. Especially multi-touch gestures are now common for interacting with mobile and various other devices (e.g. tabletops or wall-sized displays) and also made their way back to classic desktop computing. New sensor technology allows for spatial gestures that are particularly interesting in virtual and augmented reality applications: users can interact directly with virtual objects or issue commands by moving their hands, fingers, or tools like pens or magic wands in three-dimensional space.

HCI in general and gestures in particular are also key factors shaping the individual User Experience (UX). Users expect flawlessness and a decent look but also a *good feeling* – a rather fuzzy feature depending on various aspects, but particularly affected by HCI. Gestures are therefore important for an application's economic success and should be considered explicitly during development. Envisioned gestures must be defined properly and communicated between stakeholders.

Unfortunately, gestures are also hard to implement and require several decisions that cannot be reverted easily: suitable *hardware* that matches the designed interaction, appropriate *sensors* providing the needed information, and reliable *recognition algorithms* to monitor and analyze the stream of incoming data encapsulating all information relevant to identify gestures – a complex and error-prone task.

Software development nowadays involves cross-functional teams and agile methods, thus combining the strengths of technical and non-technical stakeholders. Sketching low-fidelity prototypes has been proven to be an important and highly valued activity in design and engineering disciplines, especially in early development stages, and is considered beneficial for UI design: it allows all stakeholders to quickly foster a common vision of the future application. However, sketches and prototypes typically do not explicitly define interaction, but provide an implicit definition, leaving room for misunderstandings and misconception. Gesture definition is thus important, especially when custom gestures are used, to ensure the right decisions.

The foundation for gesture definition are the various traits that recent sensor technology makes possible. Besides the shape, different information can be used for HCI, e.g. execution speed or area. Unfortunately, the optimal notation for traits may differ. Existing gesture notations focus too narrowly on either textual or graphical elements and ignore the power of combining paradigms and describing different gesture traits with different notation styles.

This thesis presents GestureCards, a hybrid gesture notation using textual and graphical elements to describe different traits. GestureCards are used to specify and document gestures and allow to describe multi-touch and spatial gestures. Various studies eliciting different aspects of GestureCards were used to evaluate the notation and the underlying concept. The studies show that GestureCards hybrid approach to gesture description is beneficial when creating gesture-based applications and supports developers in understanding defined gestures.

Inhaltsverzeichnis

I Einführung 1

1 Einleitung und Motivation 3

1.1 Moderne Mensch-Maschine-Interaktion . . . 5

1.1.1 Historische Entwicklung von HCI . . . 5

1.1.2 Gestensteuerung . . . 7

1.2 User Experience . . . 9

1.2.1 User Experience und Gestensteuerung . . . 10

1.2.2 UX Engineering . . . 11

2 Problemstellung und Zielsetzung 13

2.1 Problemstellung . . . 14

2.1.1 Gestensteuerung und Software Engineering . . . 14

2.1.2 Interaktionsdesign . . . 18

2.1.3 Spezifikation von Interaktion . . . 22

2.2 Zielsetzung . . . 24

2.3 Aufbau der Arbeit . . . 25

II Grundlagen und verwandte Arbeiten 27

3 Sketching und Prototyping 29

3.1 Begriffsklärung und Abgrenzung . . . 30

3.1.1 Sketching . . . 31

3.1.2 Prototyping . . . 33

3.2 Werkzeuge für Sketching und Prototyping . . . 35

3.2.1 Sketches . . . 35

3.2.2 Storyboards . . . 42

3.3 Interaktion und Gestensteuerung in Sketches und Prototypen . . . 45

4 Implementierung gestenbasierter Anwendungen 47

4.1 Arten von Gesten . . . 49

4.1.1 Taxonomien von Gesten . . . 50
4.1.2 Gesten und Posen . . . 53
4.2 Gestenunterstützung in typischen Frameworks . . . 55
4.2.1 Frameworks für Oberflächengesten . . . 55
4.2.2 Frameworks für räumliche Gesten . . . 57
4.3 Notwendige technische Entscheidungen . . . 58
4.3.1 Hardware und Sensoren . . . 60
4.3.2 Gestenerkenner . . . 61

5 Eigenschaften gestenbasierter Interaktion 65
5.1 Atomare Eigenschaften . . . 67
5.1.1 Anzahl der Werkzeuge . . . 67
5.1.2 Auflagefläche . . . 68
5.1.3 Ausführender Benutzer . . . 70
5.1.4 Aus- und Bewegungsrichtung . . . 74
5.1.5 Dauer . . . 75
5.1.6 Druck . . . 76
5.1.7 Form . . . 78
5.1.8 Geschwindigkeit . . . 79
5.1.9 Multiplizität . . . 80
5.1.10 Ort . . . 80
5.1.11 Reaktion . . . 84
5.1.12 Rotation . . . 86
5.1.13 Werkzeug . . . 87
5.2 Beziehungen zwischen Teilgesten . . . 89
5.2.1 Zeitliche Relation . . . 89
5.2.2 Räumliche Relation . . . 91

6 Verwandte Arbeiten – Notationen zur Beschreibung von Gesten 93
6.1 Musterbasierte Ansätze . . . 93
6.2 Textuelle Notationen . . . 94
6.2.1 GDL (Echtler) / GISpL . . . 95
6.2.2 GDL (Hachaj) . . . 96
6.2.3 GDL (Khandkar) . . . 97

6.2.4 GeForMT . . . 98
6.2.5 GestIT . . . 99
6.2.6 Hand Gesture Notation . . . 100
6.2.7 LEG . . . 101
6.2.8 Midas . . . 102
6.2.9 Prepose . . . 103
6.2.10 Proton . . . 105
6.3 Graphische Notationen . . . 106
6.3.1 EventHurdle . . . 106
6.3.2 GISMO . . . 108
6.3.3 Monox . . . 109
6.4 Hybride Notationen . . . 110
6.5 Sonstige Notationen . . . 112
6.5.1 Beschreibung und Synthetisierung von Gebärdensprache . . . 112
6.5.2 Movement Notations . . . 113

III Lösungsansatz und Validierung 115

7 GestureCards – Eine hybride Notation zur Definition von Gesten 117
7.1 Grundlagen . . . 118
7.1.1 Spezifikation einfacher Gesten . . . 118
7.1.2 Definition von Eigenschaften . . . 120
7.1.3 Spezifikation zusammengesetzter Gesten . . . 121
7.1.4 Referenzierung definierter Gesten . . . 122
7.1.5 Integration virtueller Objekte . . . 123
7.1.6 Wiederverwendung definierter Gesten . . . 124
7.2 Integration von Gesteneigenschaften . . . 126
7.2.1 Atomare Eigenschaften . . . 127
7.2.2 Beziehungen zwischen Teilgesten . . . 143

8 Lesbarkeit und Verständlichkeit textueller und hybrider Notationen 151
8.1 Aufbau der Studie . . . 152
8.1.1 Ziele . . . 152
8.1.2 Probanden . . . 153

8.1.3 Eingesetzte Gesten . . . 154
8.1.4 Eingesetzte Anwendung . . . 158
8.1.5 Hypothesen und Variablen . . . 161
8.1.6 Studiendesign . . . 162
8.1.7 Aufbau der einzelnen Sitzungen . . . 163
8.1.8 Analyse . . . 164
8.2 Ergebnisse . . . 165
8.2.1 Quantitative Analyse: Statistische Auswertung der Ergebnisse . . . 166
8.2.2 Qualitative Analyse: Auswertung von Think-Aloud-Protokollen und Beobachtungen . . . 172
8.3 Diskussion . . . 174
8.4 Risiken für die Validität . . . 176
8.4.1 Allgemeine Gültigkeit . . . 176
8.4.2 Interne Gültigkeit . . . 176
8.4.3 Externe Gültigkeit . . . 177

9 Prägnanz und Genauigkeit verschiedener Paradigmen zur Beschreibung von Gesten 179
9.1 Aufbau der Studie . . . 180
9.1.1 Ziele . . . 180
9.1.2 Probanden . . . 181
9.1.3 Eingesetzte Gesten . . . 182
9.1.4 Fragebogen . . . 185
9.1.5 Hypothesen und Variablen . . . 185
9.1.6 Durchführung . . . 186
9.1.7 Analyse . . . 187
9.2 Ergebnisse . . . 188
9.2.1 Prägnanz . . . 188
9.2.2 Genauigkeit . . . 189
9.2.3 Persönliche Präferenz . . . 191
9.3 Diskussion . . . 192
9.4 Risiken für die Validität . . . 193

10 Einsatz in der Praxis 195
10.1 Aufbau der Studie . . . 196
10.1.1 Ziele . . . 196

10.1.2 Probanden . . . 197
10.1.3 Integration von GestureCards in den AugIR . . . 197
10.1.4 Szenarien . . . 199
10.1.5 Fragebogen . . . 201
10.1.6 Aufbau der Sitzungen . . . 202
10.1.7 Analyse . . . 203
10.2 Ergebnisse . . . 203
10.2.1 Verständnis . . . 203
10.2.2 Definition . . . 204
10.2.3 Gesamteindruck . . . 205
10.2.4 Beobachtungen . . . 205
10.3 Diskussion . . . 206
10.4 Risiken für die Validität . . . 208

IV Vorschlag zur graphischen Repräsentation räumlicher Gesten 209

11 Einsatz von Handmodellen und animierten Pfeilen 211
11.1 Darstellung von Posen . . . 212
11.2 Darstellung von Gesten . . . 216
11.2.1 Einsatz eines Handmodells . . . 217
11.2.2 Einsatz animierter Bewegungspfade . . . 219

12 Vergleich graphischer Repräsentationen für räumliche Gesten 221
12.1 Aufbau der Studie . . . 222
12.1.1 Ziele . . . 222
12.1.2 Probanden . . . 223
12.1.3 Eingesetzte Gesten . . . 223
12.1.4 Eingesetzte Anwendung . . . 227
12.1.5 Hypothesen und Variablen . . . 228
12.1.6 Aufbau der Sitzungen . . . 230
12.1.7 Analyse . . . 231
12.2 Ergebnisse . . . 232
12.2.1 Korrektheit . . . 232
12.2.2 Zuversicht . . . 232

12.2.3 Dauer . . . 233
12.2.4 Replays . . . 234
12.3 Diskussion . . . 234
12.4 Risiken für die Validität . . . 236

V Epilog **237**

13 Weitere Anwendungsgebiete für GestureCards **239**
13.1 Gestenerkennung . . . 239
13.2 Prüfung definierter Interaktionsmuster . . . 242
13.3 Testautomatisierung . . . 243
13.4 Training von Template- und ML-basierten Gestenerkennern . . . 247
13.5 Bug Reports . . . 248

14 Fazit und Ausblick **251**
14.1 Diskussion der Ergebnisse . . . 252
14.2 Erweiterungspotential und zukünftige Arbeiten . . . 253
14.2.1 Erweiterung auf Bewegungen und Posen des ganzen Körpers . . . 254
14.2.2 Integration von Multimodalität . . . 255

Anhänge **257**

A Zusammenfassung aller Eigenschaften und Werte **259**
A.1 Anzahl der Werkzeuge . . . 259
A.2 Auflagefläche . . . 259
A.3 Ausführender Benutzer . . . 260
A.4 Ausrichtung . . . 260
A.5 Bewegungsrichtung . . . 260
A.6 Dauer . . . 261
A.7 Distanz . . . 261
A.8 Druck . . . 262
A.9 Endpunkt . . . 262
A.10 Form . . . 262
A.11 Geschwindigkeit . . . 262

A.12 Multiplizität . . . 263
A.13 Ort der Ausführung . . . 263
A.14 Reaktion zur Ausführungszeit . . . 263
A.15 Rotation . . . 264
A.16 Startpunkt . . . 264
A.17 Werkzeug . . . 264
A.18 Größe . . . 264
A.19 Räumlicher Bezug . . . 265
A.20 Zeitlicher Bezug . . . 265

B Material zu Kap. 8: Lesbarkeit und Verständlichkeit textueller und hybrider Notationen 267

B.1 Einführung der Notationen . . . 267
B.2 Spickzettel . . . 271
B.3 Ergebnisse aller Probanden . . . 273

C Material zu Kap. 9: Prägnanz und Genauigkeit verschiedener Paradigmen zur Beschreibung von Gesten 277

C.1 Fragebogen . . . 277
C.2 Ergebnisse aller Probanden . . . 285

D Material zu Kap. 10: Einsatz in der Praxis 293

D.1 Einführung in GestureCards . . . 293
D.2 Fragebogen . . . 295
D.3 Ergebnisse aller Probanden . . . 299

E Material zu Kap. 12: Vergleich graphischer Repräsentationen für räumliche Gesten 301

E.1 Einführung in GestureCards . . . 301
E.2 Ergebnisse aller Probanden . . . 303

Verzeichnisse 307

Literatur 309

Abbildungen 327

Tabellen 335

Teil I

Einführung

1 Einleitung und Motivation

Der Umgang mit technischen Produkten unterliegt seit einigen Jahren einem tiefgreifenden Paradigmenwechsel: Maus und Tastatur werden zunehmend durch neue Möglichkeiten der Mensch-Maschine-Interaktion (engl. *Human-Computer Interaction* (HCI)) verdrängt. Auslöser dieser Entwicklung ist maßgeblich die fortschreitende Miniaturisierung von Computern in Form mobiler Geräte. Gleichzeitig sind deshalb Lösungen notwendig, um eine effiziente und passende Interaktion mit den neuen Geräten zu ermöglichen – Maus und Tastatur sind schlicht unpraktisch und entsprechen nicht mehr den Anforderungen, die Benutzer an diese Geräteklasse haben. Ausschlaggebend hierfür ist nicht nur der Formfaktor der Geräte, sondern auch der Einsatz zu jeder Zeit im Alltag. Im Gegensatz zum klassischen Personal Computer (PC) können mobile Geräte bei (fast) jeder Gelegenheit verwendet werden und so überwiegt auch die kurzfristige, gezielte Nutzung an über den Tag verteilten Zeitpunkten (Böhmer et al., 2011).

Die rasante Verbreitung mobiler Geräte seit dem Erscheinen der ersten *Smartphones* sorgte nicht nur dafür, dass Menschen von überall auf das Internet zugreifen konnten, sondern etablierte auch eine – im Massenmarkt – neue Art, mit Computern zu interagieren: berührungsempfindliche Bildschirme oder auch *Touchscreens*. Auch wenn das erste iPhone von Apple den Startpunkt der Verbreitung im Massenmarkt markiert, reichen die Ursprünge viel weiter zurück: IBM veröffentlichte 1992 den *Simon*, der schon verschiedene Aspekte, die wir von heutigen Smartphones kennen, umsetzte und daher als Ur-Smartphone betrachtet werden muss (Buxton, 2014). Mittlerweile sind mobile Geräte derartig tief im Alltag der meisten Menschen verwurzelt, dass sie nur schwer wieder wegzudenken sind.[1] Die Möglichkeiten, die berührungsempfindliche Bildschirme bieten, sind somit zum Standard avanciert und haben inzwischen auch den Weg zurück zum klassischen PC gefunden: Neuere PCs können längst mithilfe berührungsempfindlicher Oberflächen oder Bildschirme bedient werden.

[1] In Deutschland besitzen 95 % der 14- bis 29-Jährigen und 97 % der 30- bis 49-Jährigen ein Smartphone (Bitkom, 2017).

Die Bedienung berührungsempfindlicher Oberflächen ist als Teil einer Art von HCI zu verstehen, die *Gestensteuerung* genannt wird. Gestensteuerung bezeichnet allgemein eine Art der Interaktion mit Maschinen, bei der – im Gegensatz zur klassischen Steuerung von PCs mit Maus und Tastatur – kein Zeiger zu einem virtuellen Objekt bewegt wird, um dieses zu aktivieren, sondern die Bewegung als solche die Interaktion darstellt. Die Betätigung des Buttons *Zurück* in einem Browser veranlasst diesen, auf die vorige Seite zurück zu navigieren; die gleiche Aktion lässt sich auf mobilen Geräten oftmals mit einem Wisch nach rechts auf dem Bildschirm erreichen. In ersterem Fall gibt es zwar eine Bewegung hin zum Button, diese hat jedoch keine semantische Bedeutung für die Anwendung – einzig die Aktivierung des Buttons ist relevant. Dabei ist es unerheblich, ob der Button sich auf einem PC befindet und mit der Maus aktiviert wird oder auf einem mobilen Gerät mit dem Finger berührt wird. In letzterem Fall jedoch trägt die Be-

wegung nach rechts (vielleicht gepaart mit einer Mindestgeschwindigkeit) die Bedeutung für die Interaktion mit sich. Auch wenn Gestensteuerung auf berührungsempfindlichen Oberflächen erst in den letzten Jahren dank der Verbreitung mobiler Geräte der Durchbruch im Massenmarkt gelungen ist, so ist die zugrundeliegende Idee keineswegs neu und wird seit Jahrzehnten in der HCI-Forschung thematisiert: Schon Johnson (1965, 1967), Herot & Weinzapfel (1978) und Lee et al. (1985) zeigen berührungsempfindliche Oberflächen oder Bildschirme, mit denen interagiert werden kann.[2]

[2] Eine – nach eigenen Angaben – unvollständige Chronologie ist bei Buxton (2014) zu finden.

Gestensteuerung geht aber – dank moderner Sensoren – heute über den Einsatz auf berührungsempfindlichen Bildschirmen hinaus. So können Gesten auch im dreidimensionalen Raum erkannt und verarbeitet werden. Im Rahmen dieser Arbeit werden die Begriffe *räumliche Gesten* und *Oberflächengesten* zur Unterscheidung verwendet. Zwischen räumlichen Gesten und Oberflächengesten sind einige Unterschiede zu beachten, die für die Entwicklung gestenbasierter Anwendungen relevant sind. Als offensichtliche Merkmale sei an dieser Stelle auf die zusätzliche Dimension bei räumlichen Gesten sowie die Abhängigkeit von einer Oberfläche bei Oberflächengesten verwiesen – beide gehen mit unterschiedlichen Möglichkeiten und Einschränkungen einher, die in Kap. 4 näher betrachtet werden. Es ist aber davon auszugehen, dass mit der steigenden Verbreitung von Anwendungen im Bereich Virtual Reality (VR) und Augmented Reality (AR) auch die Bedeutung räumlicher Gesten zunehmen wird.

Räumliche Gesten erweitern die Möglichkeiten von Gestensteuerung noch um Posen, denen keine Bewegung, sondern eine Positionierung von Gliedmaßen zugrunde liegt. In Kap. 4.1.2 wird im Detail auf die Unterschiede eingegangen. Darüber hinaus ermöglichen moderne Sensoren außerdem die Erfassung des gesamten Körpers, sodass z. B. auch Bewegungen der Füße erkannt werden können. Im Rahmen dieser Arbeit wird der Fokus auf die Hände gelegt, wenn von Gesten (und später Posen) die Rede ist. Ungeachtet dessen können die vorgestellten Konzepte und der Lösungsansatz sicher auch auf den ganzen Körper erweitert werden.

Die unterschiedlichen Aspekte gestenbasierter Anwendungen, die im Rahmen dieser Arbeit näher erläutert werden, müssen in allen Phasen des Software Engineering (SE) thematisiert werden, denn Gesten haben u. U. entscheidenden Einfluss auf den wirtschaftlichen Erfolg eines Projekts – sie sind in einer gestenbasierten Anwendung inhärenter Teil der Erfahrung, die der Benutzer bei Einsatz der Anwendung macht. So muss die zugrundeliegende Interaktion frühzeitig erarbeitet und die für die Anwendung angedachten Gesten soweit definiert werden, dass die notwendigen Komponenten – sowohl Hard- als auch Software – passend ausgewählt werden können. Gleichzeitig muss die Anwendung in kleinen Schritten validiert werden können, und zwar vom ersten Entwurf über den ersten Prototypen bis hin zum finalen Produkt. Dabei müssen die unterschiedlichen Herangehensweisen, Werkzeuge und Ziele der Beteiligten in den einzelnen Entwicklungsphasen berücksichtigt werden, die letztlich zu für den weiteren Entwicklungszyklus wichtigen Entscheidungen führen. Ein besonderes Augenmerk soll im Rahmen dieser Arbeit auf die Tätigkeiten von Designern und Software-Entwicklern[3] gelegt werden – die

[3] Die Begriffe *Designer* und *UI-Designer* sowie *Software-Entwickler* und *Software-Ingenieur* werden in dieser Arbeit synonym verwendet.

Kommunikation zwischen beiden Gruppen und ein gemeinsames Verständnis der Anwendung ist von direkter Bedeutung für den gesamten Verlauf der Entwicklung und den erfolgreichen Abschluss des Projekts. Zusammengefasst treffen beide Gruppen Entscheidungen, die direkt das Erleben der Anwendung durch den Benutzer beeinflussen und Konsequenzen für die Wartbarkeit und Weiterentwicklung der Anwendung haben: Während Designer ein passendes Interaktionskonzept erstellen und die passenden Gesten auswählen müssen, müssen Entwickler darauf aufbauend die Gesten mit den passenden Komponenten implementieren. In Kap. 2, Kap. 3 und Kap. 4 soll diesbezüglich tiefer ins Detail gegangen und die zugrundeliegenden Herangehensweisen und Entscheidungen vorgestellt werden.

Im Folgenden soll zunächst ein Blick in die Geschichte von HCI und den Status Quo geworfen werden (Kap. 1.1), insbesondere mit Blick auf die Entwicklung und Anwendung von Gestensteuerung. Anschließend soll betrachtet werden, welchen Einfluss das Erleben einer Anwendung auf den Benutzer und die Software-Entwicklung hat (Kap. 1.2).

1.1 Moderne Mensch-Maschine-Interaktion

Die Interaktion mit Maschinen hat sich in den vergangenen Jahrzehnten weitreichend und schnell verändert. Moderne Sensoren erlauben heute den Einsatz unterschiedlicher *Interaktionsmodalitäten*,[4] die sich im Laufe der letzten Jahrzehnte immer weiter entwickelt haben. Im Folgenden soll kurz ein Überblick über die Entwicklung von HCI im Allgemeinen gegeben und anschließend auf Gestensteuerung im Speziellen eingegangen werden.

[4] Eine *Modalität* beschreibt eine Art und Weise, einen Gedanken zu formulieren – im Kontext technischer Geräte einen Weg, mit dem Gerät zu interagieren, z. B. per Sprache oder Tastatur (Coutaz & Caelen, 1991).

1.1.1 Historische Entwicklung von HCI

Seit Menschen Computer einsetzen, unterlag die Art und Weise, mit ihnen zu interagieren, einem steten Wandel. Die folgende Darstellung entspricht der Einteilung von van Dam (1997), der die Geschichte von HCI in vier Generationen mit unterschiedlichen Merkmalen aufteilt. Jede dominierte mehrere Jahre lang die Bedienung von Computern und war auf die jeweils vorhandene Hardware zugeschnitten. Er orientiert sich dabei jedoch in erster Linie am Stand industrieller Forschung und Entwicklung sowie der Verfügbarkeit am Markt und lässt die akademische Forschung, in der viele Techniken und Möglichkeiten zur Interaktion teils Jahrzehnte vor ihrer Marktreife erforscht wurden, größtenteils außen vor.[5] Die aufgezeigte Entwicklung ist eng verknüpft mit der Verfügbarkeit von Computern für die professionelle und insbesondere private Nutzung, die eine Fülle neuer Anforderungen an die Entwickler stellte. Der Entwicklung der Benutzerschnittstelle – des User Interface (UI) – kommt dadurch zwangsläufig eine besondere Bedeutung zu.

[5] Myers (1998) und Buxton (2014) hingegen listen verschiedene Projekte und akademische Arbeiten, die maßgeblich die Entwicklung von HCI in verschiedenen Bereichen geprägt haben.

50er und 60er Jahre

Die erste Generation aus den 50er und 60er Jahren des 20. Jahrhunderts reduzierte die Interaktion mit Computern auf Lochkarten, die Anweisungen enthielten. Maschinen konnten die Informationen auf den Lochkarten ablesen,

abarbeiten und das Ergebnis anzeigen. Ein UI und eine Interaktion mit dem Programm im heutigen Sinne gab es nicht: Die gesamte Kommunikation zwischen Mensch und Maschine reduzierte sich auf die Eingabe eines Programms und die Ausgabe eines Ergebnisses.

60er bis 80er Jahre

Die zweite Generation im Sinne von van Dam (1997), die in den 60er Jahren entstand und bis in die frühen 80er die Interaktion mit Computern dominierte, führte die Kommunikation über die Konsole ein. Benutzer konnten Kommandos und Parameter über eine textbasierte Schnittstelle an Großrechner schicken. Diese Ära hält bis heute an: In UNIX-Umgebungen gehört die Verwendung der Konsole zum Standard und ist insbesondere für erfahrene Benutzer oftmals der effizienteste Interaktionskanal.[6] Bis zum Ende dieser Ära standen Performanz und Funktionalität einer Anforderung im Vordergrund (van Dam, 1997). Die Interaktion mit der Anwendung hingegen war eher nebensächlich.

[6] Die unter unixoiden Betriebssystemen gebräuchliche Abkürzung *tty* in der Konsole erinnert auch heute noch an die Ära der Großrechner: tty steht für *teletyper*, also Fernschreiber.

80er Jahre bis heute

Das sollte sich mit der dritten Generation von van Dam (1997) ändern, die Mitte der 80er Jahre durch den Macintosh populär gemacht wurde und auch heute noch den Markt dominiert. Mit dem Eintritt von Computern in den Massenmarkt wurde *Benutzerfreundlichkeit* ein wesentlicher Faktor; letztlich bedeutete dies, dass Computer vor allem einfach zu bedienen sein sollten (van Dam, 1997). Der Interaktion kam somit sehr viel mehr Bedeutung zu, denn nun war die Interaktion mit einer Anwendung auch wichtig für den kommerziellen Erfolg. Das zugrundeliegende Interaktionsparadigma – Windows, Icons, Menus, Pointer (WIMP) – ist vor allem für viele Bürotätigkeiten Alltag: Programme laufen in *Fenstern* (*Windows*) und Benutzer wählen *Symbole* (*Icons*) als Kürzel für Kommandos aus *Menüs* (*Menus*) mit einem *Zeiger* (*Pointer*) aus. Der Zeiger wird typischerweise mithilfe eines Eingabegeräts gesteuert, dessen Bewegung in Bewegung des Zeigers auf dem Bildschirm übersetzt wird. Obwohl es verschiedene Alternativen gibt, haben sich Mäuse, so wie sie nach Buxton (2014) bereits in den 60er Jahren entworfen wurden, durchgesetzt. English et al. (1967) weisen auch früh nach, dass sich Mäuse besonders gut zur Manipulation von Text auf klassischen Computern eignen, wenn auch ein wenig Übung bei den Probanden notwendig war. Die Benutzeroberfläche von WIMP-UIs selbst ist zweidimensional und besteht in erster Linie aus Widgets, die einfache Aktionen erlauben (z. B. Buttons, die angeklickt werden, oder Regler, die verschoben werden).

Demnächst

Die vierte Generation, die van Dam (1997) definiert, wird durch intensive Nutzung von Sprach- und Gestensteuerung letztlich die WIMP-UIs verdrängen. Darüber hinaus lässt sich beobachten, dass sich Computer – vor allem in Form mobiler Geräte – von einem Werkzeug zu einem Lifestyle-Produkt entwickelt haben. Damit einher geht eine andere Erwartungshaltung, die auch etwas mit HCI und neuen Möglichkeiten zu tun hat. Die Überschrift dieses Abschnitts ist aber eigentlich irreführend, denn tatsächlich gibt es diese neue Art von HCI längst. Das dem WIMP-UI zugrundeliegende Paradigma des *Point'n'Click*

(PnC), des *Zeigens* und *Anklickens*, ist aber heute selbst auf mobilen Geräten, die auf die klassischen Interaktionsmodalitäten Maus und Tastatur weitgehend verzichten, noch vorherrschend – selbst wenn Buttons mit den Fingern angeklickt werden, so bleibt das zugrundeliegende Paradigma eben noch WIMP. Aktuell ist HCI geprägt von einem Nebeneinander unterschiedlicher Modalitäten, die sich z. T. erst noch etablieren müssen. Gestensteuerung hat diesen Sprung zwar zumindest im Falle berührungsempfindlicher Oberflächen durch die Verbreitung mobiler Geräte bereits weiter geschafft als andere, steht jedoch in vielen Anwendungsfällen noch am Anfang. Das äußert sich z. B. dadurch, dass sich erst allmählich eine Art De-Facto-Standard für einfache Gesten herauskristallisiert (vgl. Kap. 4.2), während WIMP-UIs weitgehend standardisiert sind.

1.1.2 Gestensteuerung

Zwar lässt sich Gestensteuerung weiter unterteilen (vgl. Kap. 4.1.2), jedoch ist ein wesentlicher Unterschied zu klassischer PnC-Interaktion der Fokus auf Bewegung. Natürlich wird bei PnC-Interaktion ebenfalls ein Zeiger bewegt, typischerweise mit der Maus, aber die Bewegung hat erstmal keine Bedeutung für die Anwendung, sie ist nur Mittel zum Zweck – die eigentliche Interaktion findet in der Regel durch das Anklicken eines Elements statt. Ausnahmen bestätigen allerdings die Regel: Es gibt auch in klassischen PnC-Anwendungen Interaktionen, die an der Bewegung hängen, z. B. das Markieren eines Bereichs. Darüber hinaus setzen einige Anwendungen auch auf *Mausgesten*, also Gesten, die mit der Maus ausgeführt werden. Mausgesten waren z. B. lange Zeit ein Merkmal des Browsers Opera (Opera Browser, oJ).

Im Vergleich zu klassischer Interaktion mit Maus und Tastatur bietet Gestensteuerung zahlreiche unterschiedliche Möglichkeiten, die Eingaben zu verarbeiten bzw. sie für die Interaktion einzusetzen. In Kap. 5 werden unterschiedliche Eigenschaften von Gesten näher betrachtet und ihr Einsatz in unterschiedlichen Anwendungen beleuchtet. Darüber hinaus entfernt Gestensteuerung in vielen Fällen die Notwendigkeit eines Mediators, mit dem der Zeiger auf dem Bildschirm bewegt wird. Jerald (2016) sieht sie als Beispiel für *direkte Interaktion*. Benutzer können ohne Umwege mit einem virtuellen Objekt interagieren, z. B. indem sie es auf einem berührungsempfindlichen Bildschirm berühren und an eine andere Stelle schieben. Bei *indirekter Interaktion* hingegen wird eine Modalität stets als Mediator eingesetzt, die die Interaktion des Benutzers übersetzt. Das muss nicht zwingend eine Maus sein; so kann nach einem Bild auch auf unterschiedliche Art und Weise mit der Tastatur gesucht werden (z. B. Bildname eingeben, mit den Pfeiltasten scrollen usw.). Benutzer müssen in dem Fall überlegen, wie sie Anfragen formulieren und das Ergebnis interpretieren. Bei Gestensteuerung kann das Bilderverzeichnis mit der Hand manipuliert werden, um das gewünschte Bild zu finden.

Die Vielseitigkeit von Gestensteuerung findet sich nicht nur in den unterschiedlichen Eigenschaften, die definieren, wie eine Geste ausgeführt wird, sondern spiegelt sich auch in den verschiedenen Sensoren wider, die zur Erkennung

von Gesten eingesetzt werden können. Mobile Geräte, die mit berührungsempfindlichen Bildschirmen ausgestattet sind, wurden bereits mehrfach erwähnt und sind sicher eine der verbreitetsten Geräteklassen. Kommerziell erhältlich sind auch interaktive Tische (*Multi-Touch-Tables*) oder digitale Whiteboards, die ebenso wie mobile Geräte über berührungsempfindliche Bildschirme verfügen, aber aufgrund ihrer Dimension eine andere Art von Gestensteuerung ermöglichen, z. B. weil beide Hände und alle Finger parallel eingesetzt werden können. Darüber hinaus ist – passende Sensoren vorausgesetzt – fast jede Oberfläche für Gestensteuerung nutzbar (Harrison & Hudson, 2008), sogar die menschliche Haut (Zhang & Harrison, 2015; Zhang et al., 2016). Auch zur Erkennung von Gesten im dreidimensionalen Raum ist heute eine Vielzahl unterschiedlicher Sensoren verfügbar. Außerdem können bestehende UI-Widgets mit Gesten quasi aufgewertet werden, z. B. zur Auswahl mehrerer Checkboxen auf einmal (Moscovich, 2009).

Die Interaktion mit Gesten hat gegenüber PnC bzw. klassischen WIMP-UIs einige Vorteile. So lassen sich unterschiedliche Gesten verwenden, um das gleiche Ziel zu erreichen. Je nach Anwendungsfall und Situation der Nutzer können entsprechend zielgerichtet bestimmte Eigenschaften von Gesten genutzt werden. Außerdem zeigen z. B. Appert & Zhai (2009), dass Gesten als Kürzel für Kommandos klassischen Tastaturkürzeln überlegen sind und nach etwas Training schneller erlernt und verwendet werden können. Die o. g. Mausgesten im Browser Opera sind in diese Kategorie einzuordnen. Allerdings gibt es nach Zhai et al. (2012, Kap. 8) bisher keine nachweisliche Erklärung, warum Gesten vor allem einfacher zu merken sind, sondern lediglich Erklärungsversuche. Dem steht gegenüber, dass es auch einige Anwendungsfälle gibt, bei denen klassische Interaktion mit Maus und Tastatur besser geeignet ist als Gestensteuerung. Saffer (2008) stellt als prägnantes Beispiel die Eingabe großer Datenmengen heraus – hier können klassische WIMP-UIs ihre Stärken ausspielen. Für viele typische Büroanwendungen werden Maus und Tastatur weiterhin die zentralen Modalitäten zur Interaktion sein. Für andere Anwendungen, vor allem auf mobilen Geräten, haben Gesten Vorteile gegenüber WIMP-UIs, insbesondere wenn Benutzer abgelenkt sind und sich nicht hundertprozentig auf die Interaktion mit dem Gerät konzentrieren können (Bragdon et al., 2011). Gleichzeitig existieren bei WIMP-UIs langjährige Erfahrungswerte sowohl bei jenen, die Software entwickeln, als auch bei jenen, die Software verwenden. Bei Gestensteuerung hingegen müssen viele Erfahrungen erst noch gesammelt werden.

Gestensteuerung hat also viele verschiedene Facetten. Außerdem gibt es Anwendungsfälle, für die unterschiedliche Modalitäten unterschiedlich gut geeignet sind. Gestensteuerung macht hier keine Ausnahme, hat aber diverse Vorteile. Darüber hinaus machen moderne Sensoren auch viele weitere Modalitäten möglich: Sprachsteuerung ist seit vielen Jahren ein Thema und ist mit der Verfügbarkeit virtueller Assistenten (ebenfalls) auf mobilen Geräten immer häufiger im Alltag anzutreffen. Diese neuen Modalitäten haben den Anspruch, eine *natürliche* Interaktion zu ermöglichen, jedoch wird dieser Begriff oftmals missverstanden: Statt einer Interaktion *wie in der realen Welt* ist hierunter eher eine Philosophie bei der Entwicklung gemeint, die darauf abzielt, Benutzern für ihren jeweiligen Anwendungsfall und ihre jeweilige

Situation die bestmögliche Interaktion zur Verfügung zu stellen, damit sie sich schnell zu Experten in der Bedienung entwickeln können (Wigdor & Wixon, 2011). Das ist nicht auf bestimmte Modalitäten beschränkt, denn auch klassische PnC-Interaktion mit Maus und Tastatur kann *natürlich* für die jeweilige Aufgabe sein – sie muss vor allem den Möglichkeiten der Benutzer entsprechen, ihre Anforderungen erfüllen, ihre Fähigkeiten voll ausschöpfen und zu ihrer Aufgabe und ihrem Kontext passen. Die Kunst bei der Entwicklung interaktiver Anwendungen ist somit, die passende Modalität für den jeweiligen Anwendungsfall zu finden und Benutzern das Gefühl zu vermitteln, die Anwendung beherrschen zu können. Erforderlich hierfür ist ein Entwicklungsprozess, der das Wissen unterschiedlicher Experten vereinigt und den Nutzer in den Mittelpunkt der Entwicklung stellt.

1.2 User Experience

Der Paradigmenwechsel im Bereich HCI von klassischer Interaktion mit Maus und Tastatur hin zu mehr unterschiedlichen Interaktionsmodalitäten – wobei Gestensteuerung dabei besonders hervorzuheben ist – geht einher mit einem neuen Qualitätsmerkmal für Applikationen: Benutzer erwarten ein nicht näher definiertes Erlebnis, die User Experience (UX). Ein allgemeiner Konsens, was genau UX eigentlich ist und wie sie definiert werden muss, existiert bisher nicht, jedoch hat sich weitgehend die Ansicht durchgesetzt, dass das Erleben einer Anwendung stark von der Interaktion beeinflusst wird und außerdem subjektiv und kontextabhängig ist (Law et al., 2009). Zwar wurde das Konzept der UX sowohl in der Forschung als auch der Industrie schnell aufgenommen, jedoch wurden genauso schnell kritische Stimmen laut, die jedweden Versuch einer Definition als zu vage und schwer fassbar bezeichneten (Hassenzahl & Tractinsky, 2006). Dass eine gute UX für den Erfolg und die Akzeptanz einer Anwendung von entscheidender Bedeutung ist, scheint jedoch allgemein akzeptiert zu sein. Entsprechend haben zahlreiche Autoren versucht, den Begriff zu definieren und die unterschiedlichen Komponenten genau festzuhalten, um eben eine optimale UX entwickeln zu können.

Hassenzahl & Tractinsky (2006) beschreiben drei Aspekte von UX, die sie aus der wissenschaftlichen Literatur ableiten:

1. Technologie als mehr als ein reines Werkzeug
2. Emotionen von Benutzern
3. Erfahrungen bei der Verwendung von Technologie

Zusammengefasst kommen sie zu dem Ergebnis, dass die drei Aspekte nicht streng voneinander getrennt werden können, sondern Überlappungen aufweisen. Gleichzeitig definieren sie darauf aufbauend auch Eigenschaften, die gute UX ausmachen. So müssen Anwendungen z. B. auch dem ästhetischen Empfinden des Nutzers entsprechen. Darüber hinaus müssen Anwendungen positive Emotionen wecken.[7] Und letztlich spielen Kontext und Situation eine entscheidende Rolle, um die UX des Benutzers mit der Anwendung zu verbessern. Somit sind zahlreiche Faktoren zu berücksichtigen, die für Hassenzahl & Tractinsky (2006) verschiedene Möglichkeiten bieten, gute UX

[7] Wodurch auch die Erkennung negativer Emotionen in diesem Kontext interessant wird, damit die Anwendung z. B. die Reaktion auf einen Benutzer anpassen kann.

zu entwerfen. Die unterschiedlichen Faktoren und Aspekte, die die Autoren nennen, lassen den Schluss zu, dass weit über die reine Funktionalität hinausgehende Elemente für die Entwicklung einer Anwendung wesentlich sind. Sie fassen zusammen, dass HCI über die reine Nützlichkeit von Anwendungen reichen muss.

Hinckley & Wigdor (2012) definieren UX mit besonderem Fokus auf die Interaktion mit der Anwendung.[8] Für sie spielen dabei Ein- und Ausgabe eine grundlegende Rolle, sie betonen aber auch andere Faktoren, die der Nutzer mit der Anwendung in Verbindung bringt. Dazu gehören die Erreichung seiner eigentlichen Ziele bei der Verwendung, seine Wahrnehmung dabei sowie die Emotionen, die durch die Anwendung ausgelöst werden. Für Watzman & Re (2012) ist UX das Herz des UI-Designs[9] – hier wird bestimmt, wie sich der Nutzer bei Verwenden der Anwendung fühlt. Die Autoren betonen insbesondere, dass ständiges ständiges Testen der Anwendung auf allen denkbaren Geräten wichtig ist, um eine gute UX sicherzustellen.

Dass eben genau dieses Erleben der Anwendung nur schwer zu fassen ist, zeigen Kaasinen et al. (2015). Sie definieren in einer Fallstudie zwei UX-Ziele für eine gestenbasierte Anwendung: *Verwendung fühlt sich wie Magie an* und *Benutzer hat das Gefühl der Kontrolle*. Ersteres soll Benutzer überraschen, aber auch intuitiv sein; Letzteres zielt insbesondere auf einfache Bedienung und robuste Gestenerkennung. Während diese Ziele auf den ersten Blick widersprüchlich erscheinen, so verdeutlichen sie auch, was UX erreichen will: Im Vordergrund stehen die Ziele und Aktionen des Benutzers, der sich um technische Zusammenhänge keine Gedanken machen muss. Diese Schlussfolgerung und die o. g. Definition von UX haben eine Gemeinsamkeit: Sie legen den Fokus auf die Interaktion mit der Anwendung. Daraus folgt, dass die Erarbeitung der Interaktion sowie ihre Umsetzung explizit im Entwicklungsprozess thematisiert und die erarbeiteten Konzepte zwischen den Beteiligten kommuniziert werden müssen.

[8] „User experience is the broad array of outputs perceived and inputs given by a user when interacting with a user interface, as well as the higher-level goals, cognitive states, emotions, and social interactions that such experiences support and engender.“ (Hinckley & Wigdor, 2012, Kap. 6.2.2)

[9] „The heart of interface design is to define the user's experience; what it is really like for people facing the monitor, using a cell phone, or an ATM.“ (Watzman & Re, 2012, Kap. 14.18)

1.2.1 User Experience und Gestensteuerung

Aus der Erkenntnis, dass die Interaktion mit einer Anwendung ein wesentlicher Teil der UX ist, folgt zwangsläufig, dass auch Gestensteuerung einen maßgeblichen Einfluss hat. Während klassische WIMP-UIs oftmals weitgehend dem gleichen Aufbau folgen, ermöglicht Gestensteuerung mehr Freiheiten und insbesondere unterschiedliche Variationen, wie Benutzer mit der Anwendung interagieren können: Einsatz mehrerer Finger und unterschiedlicher Werkzeuge oder Feinheiten wie schnelle und langsame Bewegungen oder starker und schwacher Druck. Dabei können z. B. unterschiedliche Varianten parallel implementiert werden, um dem Benutzer die Auswahl zu lassen.

Aber nicht nur die Ausführung, sondern auch die Auswahl von Gesten wird durch benutzerspezifische Faktoren beeinflusst. Hinrichs & Carpendale (2011) zeigen, dass neben der persönlichen Präferenz auch der soziale Kontext und der aktuelle Verwendungskontext die Interaktion mit der Anwendung beeinflussen und bestimmen, welche Gesten Benutzer bevorzugen. Besonders interessant in dem Zusammenhang: Die Autoren zeigen auch, dass Gesten

untereinander abhängig sind. Diese Erkenntnis hat weiterreichende Folgen, denn sie bedeutet, dass bei der Entwicklung gestenbasierter Anwendungen ein übergeordnetes Interaktionskonzept notwendig ist, das den verwendeten Gesten eine Zusammengehörigkeit gibt. Anders ausgedrückt: Es genügt nicht, nur die Verbindung eines Anwendungsfalls und der zugehörigen Geste zu betrachten, sondern es müssen stets alle Gesten im Blick gehalten werden, um ein gemeinsames Bild der Interaktion zu erreichen. Eine andere Konsequenz ist, dass u. U. mehrere Interaktionskonzepte entwickelt werden müssen, die den Fokus auf unterschiedliche Eigenschaften von Gesten legen und so den unterschiedlichen Vorlieben und Erfahrungsständen der Benutzer gerecht werden. Hinrichs & Carpendale (2011) kommen im Prinzip zum gleichen Schluss und fordern, dass für die gleiche Aktion mehrere Gesten definiert werden. Ähnliches gibt es auch bereits in klassischen WIMP-UIs: Neben der Bedienung mit der Maus werden oftmals Tastaturkürzel integriert, die zwar erst erlernt werden müssen, aber eine schnellere Bedienung ermöglichen.

Darüber hinaus hängt die Ausführung von Gesten an verschiedenen, teils benutzerspezifischen Faktoren. So wird bei Bevan & Fraser (2016) deutlich, dass Benutzer mit längerem Daumen Swipes – also Wischbewegungen auf dem Bildschirm – schneller ausführen als Benutzer mit kürzerem Daumen. Außerdem üben männliche Benutzer in den Beobachtungen der Autoren mehr Druck auf den Bildschirm aus als weibliche. Hier zeigt sich, dass die Ausführung von Gesten benutzerspezifischen Varianzen unterliegt, die aber während der Bedienung nicht zutage treten bzw. zu fehlerhaften Eingaben führen dürfen. In anderen Worten: Eine gestenbasierte Anwendung muss robust gegenüber der Bedienung durch unterschiedliche Benutzer sein.

Gleichzeitig zeigt die Forschung auch, dass Gesten im hohen Maße abhängig von sozialen Konventionen sind. So ist bei Hinrichs & Carpendale (2011) sichtbar, dass Faktoren wie das Alter der Benutzer und die Nähe zu anderen Benutzern die verwendeten Gesten beeinflussten. Die Konsequenz ist, dass der Kontext der Anwendung ebenfalls intensiv betrachtet werden muss. Die Nähe von Gestensteuerung zu zwischenmenschlicher Kommunikation hat auch zur Folge, dass ebendiese explizit betrachtet werden muss, um eine schlechte UX durch – zwischenmenschlich – unpassende Gesten zu vermeiden. So z. B. wird das Hochführen des ausgestreckten Daumens in unterschiedlichen Kulturen entweder als Bestätigung oder als Beleidigung aufgefasst.

1.2.2 UX Engineering

In Kap. 1.2.1 wird deutlich, dass UX wichtig für den Erfolg eines Projekts ist und somit explizit betrachtet werden muss. Dafür wurden in den vergangenen Jahren entsprechende Methoden, Werkzeuge und Prozesse zur ganzheitlichen Entwicklung von UX vorgestellt (z. B. von Buxton (2007), Greenberg et al. (2011) und Hartson & Pyla (2012)). Ihnen ist zumeist gemein, dass sie ein iteratives Vorgehen vorschlagen und eine intensive Einbindung der Benutzer zur Validierung der Iterationen vorsehen. Dass die Ergebnisse der UX-Entwicklung auch für den weiteren Verlauf des Projekts relevant sind und entsprechenden Einfluss üben, wird durch die dargestellte Bedeutung von UX

für den Erfolg eines Software-Produkts deutlich. Damit geht zwangsläufig einher, dass bei der Entwicklung der UX die Interaktion dediziert betrachtet und die Ergebnisse und Entscheidungen für das weitere Vorgehen im Projekt festgehalten werden müssen. Hartson & Pyla (2012, Kap. 23) weisen jedoch explizit darauf hin, dass sich die Informationen, die zwischen UX-Desigern und Software-Ingenieuren ausgetauscht werden, meist auf Anwendungsfälle und UI-Sketches reduzieren, jedoch vor allem die Interaktion genau spezifiziert werden muss, damit die notwendigen Informationen zwischen allen Beteiligten ausgetauscht werden können.[10]

Die Folge ist, dass die Entwicklung von UX explizit in den Gesamtprozess der Software-Entwicklung integriert werden muss. Verschiedene Autoren haben das auch erkannt und entsprechende Vorgehensmodelle vorgestellt. So erarbeiten Kuusinen & Mikkonen (2014) in einem empirischen Vorgehen, wie grundlegende Aktivitäten des UX Engineering[11] direkt in SCRUM integriert werden können. Peres & Meira (2015) gehen noch einen Schritt weiter und schlagen vor, die unterschiedlichen Aktivitäten des UX Engineerings nicht nur in SCRUM zu integrieren, sondern auch an den Regeln von Capability Maturity Model Integration (CMMI) auszurichten. Letztlich bleibt aber unklar, wie die Interaktion zwischen Mensch und Maschine definiert werden muss. Außerdem gehen die Autoren nicht darauf ein, wie die Faktoren berücksichtigt werden können, die eine gute UX für gestenbasierte Anwendungen ausmachen. Hier bieten sich unterschiedliche Ansätze an. Nielsen et al. (2003, 2004) unterscheiden zwischen einem technologie- und einem nutzerbasiertem Ansatz,[12] bei dem jeweils die Fähigkeiten der eingesetzten Technik oder die Anforderungen der Nutzer als Ausgangspunkt gewählt werden.

Somit wird deutlich, dass auch die vorgesehenen Gesten letztlich spezifiziert und zwischen Mitgliedern eines interdisziplinären Projektteams ausgetauscht werden müssen, um die notwendigen Entscheidungen zur Implementierung der Anwendung zu treffen. Die unterschiedlichen Herangehensweisen, Zielstellungen und Perspektiven auf die Interaktion müssen dabei berücksichtigt werden, um einen passenden Mittelweg zu finden. Dazu muss betrachtet werden, wie interaktive Anwendungen auf der einen Seite entworfen und auf anderen Seite implementiert werden. Die beteiligten Akteure setzen dabei unterschiedliche Werkzeuge ein, in denen die zu dokumentierenden Informationen über Gesten enthalten sind. Davon handelt diese Arbeit.

[10] „[...], in reality, there is a need for UX people to produce interaction design specifications and for SE people to make a connection with them in their lifecycle.“ (Hartson & Pyla, 2012, Kap. 23.5.1)

[11] Kuusinen & Mikkonen (2014) bauen ihre Arbeit auf vier Aktivitäten des UX Engineerings auf: UX-Vision definieren, Konzept (z. B. Sketch oder Prototyp) ableiten, Interaktion und Visualisierung entwerfen, UX evaluieren.

[12] Beide Ansätze haben Vor- und Nachteile; in Kap. 2.1.3 wird auf den Unterschied und die Folgen explizit eingegangen.

2 Problemstellung und Zielsetzung

Die steigende Verbreitung von Gestensteuerung auf unterschiedlichen Geräten und verschiedenen Plattformen eröffnet Benutzern zwar interessante Möglichkeiten für neue Arten von HCI und verspricht somit positiven Einfluss auf die UX, sorgt aber gleichzeitig für Herausforderungen in der Software-Entwicklung. Da die Interaktion mit der Anwendung die Akzeptanz ebenjener bei Benutzern maßgeblich beeinflusst, kann sie somit auch den Erfolg eines Software-Projekts gefährden. Daraus folgt, dass eine intensive Auseinandersetzung mit den für den jeweiligen Anwendungsfall denkbaren Möglichkeiten der Interaktion notwendig ist. Im Rahmen eines Entwicklungsprojekts muss bereits in frühen Entwicklungsphasen erarbeitet werden, wie Nutzer mit der Anwendung interagieren, um zum einen die Anwendung möglichst früh evaluieren und die Reaktion der Nutzer prüfen zu können, und zum anderen, um die Grundlagen für die notwendigen Entscheidungen zur Implementierung der Interaktion während der Programmierung zu legen.

In Software-Projekten sind aber frühe Phasen heute durch eine andere Herangehensweise und andere Schwerpunkte geprägt als späte Phasen. Während späte Phasen insbesondere technische Implementierungsdetails betrachten, fokussieren frühe Phasen eher die Arbeitsweise der Nutzer und die zugrundeliegenden Anwendungsszenarien. Am Ende der Designphase steht typischerweise ein mit potentiellen Nutzern evaluierter Prototyp, auf dessen Grundlage in der Engineering-Phase das finale Produkt implementiert wird. In diesem Prototypen ist die Interaktion mit der Anwendung implizit festgelegt. Somit zeigt sich an der Schnittstelle vom Design der Anwendung hin zur Implementierung ein Kommunikationsproblem, das sich in der mangelhaften Definition der Interaktion manifestiert. Diese Arbeit betrachtet ebendiese Schnittstelle in Bezug auf die Entwicklung gestenbasierter Anwendungen und schlägt eine Notation zur expliziten Spezifikation von Gesten vor.

Im Folgenden soll zunächst die Problemstellung, die dieser Arbeit zugrunde liegt, erörtert werden (Kap. 2.1). Dazu soll im ersten Schritt Gestensteuerung von klassischer Interaktion in WIMP-UIs aus Sicht eines Software-Ingenieurs abgegrenzt und die Folgen für die Software-Entwicklung dargestellt werden (Kap. 2.1.1). Es wird gezeigt, dass im Übergang vom UI-Design hin zur Implementierung eine Lücke herrscht, die geschlossen werden muss. Darauf aufbauend wird diese Schnittstelle genauer betrachtet und auf die Verantwortlichkeiten, Aufgaben und damit Interessen von Designern und Software-Ingenieuren näher eingegangen (Kap. 2.1.2). Letztlich wird daraus die Notwendigkeit der Spezifikation von Interaktion abgeleitet und deren Grundlagen erörtert (Kap. 2.1.3). Aus den beschriebenen Problemen werden dann die Zielstellung sowie die Forschungsfragen abgeleitet, die im Rahmen dieser Arbeit behandelt werden (Kap. 2.2). Das Kapitel schließt mit einer Beschreibung des Aufbaus der Arbeit (Kap. 2.3).

2.1 Problemstellung

Die steigende Bedeutung von UX für die Akzeptanz einer Anwendung sowie die fortschreitende Verbreitung neuer HCI-Technologie geht mit zwei wesentlichen Änderungen für die Entwicklung interaktiver Anwendungen einher. So steigen auf der einen Seite die Anforderungen an technisches Fachwissen, da die Verarbeitung von Gestensteuerung technisch anspruchsvoll ist, und auf der anderen Seite entstehen neue Rollen und Aufgabengebiete, die von Entwicklungsteams besetzt werden müssen.

Gestensteuerung ist schwieriger zu verarbeiten als klassische Interaktion mit WIMP-UIs und erfordert unterschiedliche Entscheidungen der verantwortlichen Entwickler. Es müssen sowohl passende Sensoren als auch Algorithmen zur Verarbeitung der Eingabedaten ausgewählt werden. Damit einher geht, dass u. U. auch die notwendigen Eingabedaten definiert werden müssen. Sollten die Sensoren bereits vorgegeben sein (z. B. durch die Anforderung, vorhandene Geräte zu nutzen), dann muss geprüft werden, ob die definierte Interaktion überhaupt mit den gelieferten Daten und gegebenen Mitteln umgesetzt werden kann.

Darüber hinaus erfolgt eine weitere Spezialisierung einzelner Aufgabengebiete und neuer Tätigkeiten innerhalb des gesamten Entwicklungsprozesses. Insbesondere Designer, die in frühen Phasen der Entwicklung das Interaktionskonzept der Anwendung festlegen und auf die Tauglichkeit im angedachten Nutzungskontext evaluieren, treffen bereits vor der eigentlichen Implementierung weitreichende Entscheidungen, die alle nachgelagerten Phasen maßgeblich beeinflussen. Mit der steigenden Bedeutung von Interaktion für die UX und somit den Erfolg einer Anwendung wird auch eine spezielle Ausbildung im Design von HCI notwendig. Somit treten im bereits interdisziplinären Feld des SE neue Akteure auf, die integriert werden müssen. Von dieser Annahme ausgehend wird im Folgenden betrachtet, was diese Entwicklung für die Arbeit im SE bedeutet.

2.1.1 Gestensteuerung und Software Engineering

Während sich die Interaktion des Nutzers in klassischen maus- und tastaturgesteuerten Oberflächen, die dem WIMP-Paradigma folgen, insbesondere auf die Abgabe konkreter, atomarer Kommandos reduziert (z. B. Buttons oder Links anklicken, Checkboxen aktivieren oder Regler (Slider) verschieben), sind bei Gestensteuerung zahlreiche unterschiedliche Eigenschaften relevant, um einzelne Gesten zielgenau voneinander zu unterscheiden. Diese müssen zur Laufzeit der Anwendung analysiert und interpretiert werden, um die ausgeführte Geste zu erkennen und die Anwendung korrekt reagieren zu lassen. Diese Reaktion kann nicht nur durch die Geste an sich ausgelöst, sondern auch in Abhängigkeit der Ausführung der Geste parametrisiert werden – z. B. kann die Geschwindigkeit einer Wischgeste beeinflussen, wie schnell sich ein Element auf dem Bildschirm bewegt. Gleichzeitig liegt es in der Natur von Gestensteuerung, dass es zu Ähnlichkeiten zwischen definierten Gesten kommen kann oder die Ungenauigkeit der Ausführung durch den Benutzer

Interpretationsspielraum lässt. Zwar bieten Frameworks zur Integration von Sensoren wie z. B. Touchscreens typischerweise integrierte Erkenner für einen grundlegenden Satz von Gesten an (z. B. in Android (Android Developers, oJ[d]) oder iOS (Apple Developer, 2017b), vgl. Kap. 4.2), bei der Implementierung individueller Gesten müssen Entwickler jedoch die eintreffenden Daten selber verarbeiten. Somit unterscheiden sich Gesten von klassischen UI-Elementen in wesentlichen Punkten:

- Ausgeführte Gesten liefern kontinuierliche Datenströme, die interpretiert werden müssen (z. B. die Erkennung eines konstanten Koordinatenstroms als koordinierte Bewegung eines Fingers), anstatt einfacher Eingabeereignisse, wie sie von klassischen UI-Elementen erzeugt werden. Während auf einfache Eingabeereignisse eine direkte Reaktion folgen kann, also eine 1-zu-1-Verknüpfung von Ereignis und zugehöriger Funktion besteht, müssen bei Gestensteuerung die eintreffenden Daten gruppiert, aggregiert und konstant analysiert werden, um die ausgeführte Geste aus dem Strom eintreffender Daten zu identifizieren und ggf. auf die Eingabe mit Aufruf der entsprechenden Funktionalität zu reagieren. Statt also wie bei WIMP-UIs üblich den Zeitpunkt der Interaktion zu betrachten, muss bei Gestensteuerung – abgesehen von einfachen Berührungen – ein Zeitraum betrachtet werden.

- Gestensteuerung erfordert eine explizite Berücksichtigung nicht nur der zeitlichen, sondern auch der räumlichen Komponenten der Interaktion, d. h. von Reihenfolge, Abstand oder Überlappung einzelner Gestenzüge und von möglichen Bezügen zwischen ihnen. Abb. 2.1 veranschaulicht exemplarisch, wie Berührungsinformationen auf einem Touchscreen zwar sequentiell eintreffen, jedoch unterschiedlichen, parallel aufliegenden Fingern zugeordnet werden müssen, um die einzelnen Elemente der Geste zu erkennen.

- Klassische UI-Elemente stellen vordefinierte, einfache Aktionen zur Verfügung, die sich entsprechend einfach spezifizieren und implementieren lassen. Für Gestensteuerung hingegen müssen weitere Eigenschaften wie z. B. die Geschwindigkeit oder der Ort der Ausführung u. U. explizit definiert werden. Die gleiche Geste kann unterschiedliche Bedeutung für die Anwendung haben, je nachdem wie schnell und wo sie ausgeführt wird.

- Die Eingabedaten sind *fuzzy*, d. h. es ist davon auszugehen, dass Anwender Gesten nicht immer exakt ausführen. Des Weiteren können sich Eingaben von Anwender zu Anwender in Nuancen unterscheiden (z. B. die Länge oder Geschwindigkeit einer Swipe-Geste, die sogar als biometrisches Identifikationsmerkmal verwendet werden können (Bevan & Fraser, 2016)). Toleranzen müssen berücksichtigt und Ähnlichkeiten zwischen definierten Gesten vermieden werden, um Verwirrung oder Frust bei Anwendern entgegenzuwirken. Entwickler müssen entsprechend beim Aufspüren dieser Probleme schon frühzeitig unterstützt werden, um aufwendige Korrekturen im Nachhinein zu vermeiden.

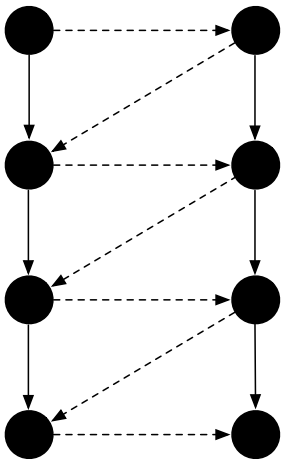


Abbildung 2.1: Exemplarisches Eintreffen (gestrichelte Linie) und logische Zuordnung (durchgezogene Linie) der Berührungspunkte zweier Finger auf einem berührungsempfindlichen Bildschirm

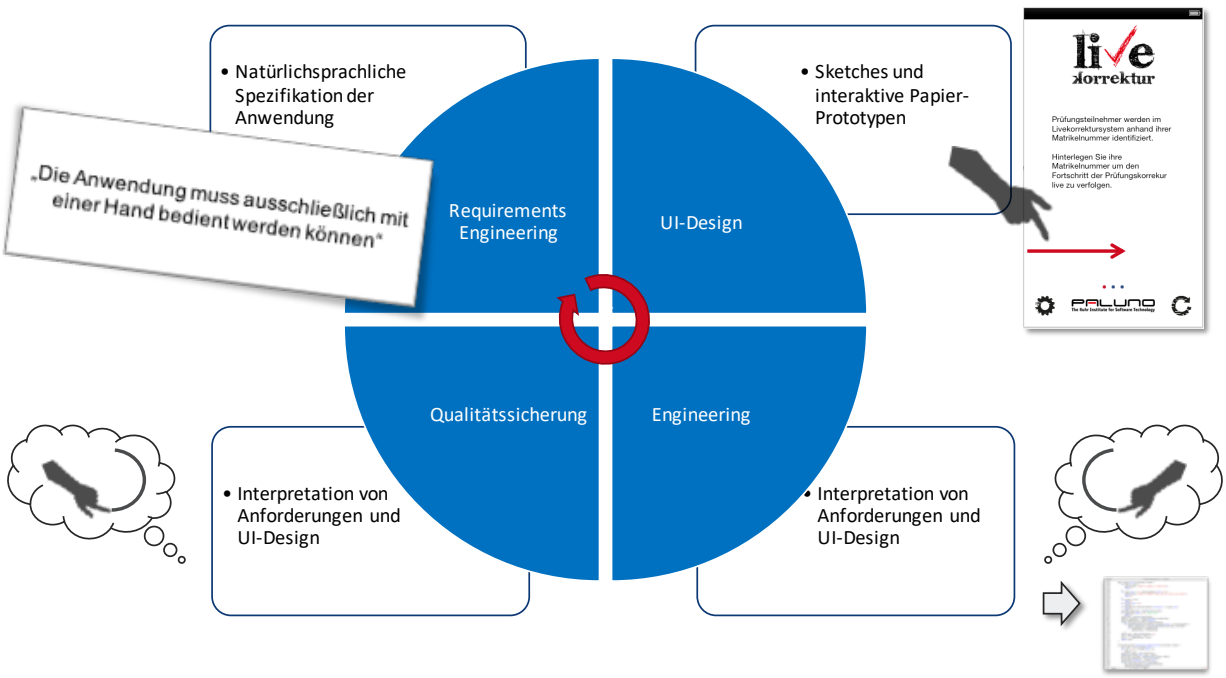


Abbildung 2.2: Gesten im SE-Zyklus

Diese Unterschiede erfordern neue technische Ansätze, sind aber durch Auswahl der passenden Komponenten und korrekte Implementierung lösbar (vgl. Kap. 4). Darüber hinaus stellen sich aber methodische Herausforderungen, die die gestiegene Bedeutung von Interaktion und UX mit sich bringen. Software-Entwickler haben darauf mit der Einführung agiler und iterativer Entwicklungsprozesse reagiert, in die sich die notwendigen Schritte zur Entwicklung einer guten UX integrieren lassen (vgl. Kap. 1.2.2). Dabei wird insbesondere auf eine frühzeitige Einbindung der angepeilten Nutzergruppe Wert gelegt. SE ist heute aber eine vorwiegend im Team ausgeführte Tätigkeit, da für die zahlreichen Aufgaben, die hier anfallen, unterschiedliches Fachwissen notwendig ist. Daraus folgt zwangsläufig, dass die Aufteilung der Aufgaben koordiniert und insbesondere Informationen zwischen den Teammitgliedern ausgetauscht werden müssen.

Abb. 2.2 zeigt einen heute üblichen SE-Zyklus. Im ersten Schritt werden Anforderungen an die Anwendung erfasst. Dabei werden bevorzugt Gespräche mit Vertretern der Nutzer geführt und vorwiegend in natürlicher Sprache dokumentiert. Der Einbezug technisch unbedarfter, aber in der jeweiligen Anwendungsdomäne fachlich versierter Nutzer steht dabei vor verschiedenen Herausforderungen. Die wesentliche ist, eine gemeinsame Sprache zu finden, in der die unterschiedlichen Anwendungsszenarien für alle Beteiligten verständlich formuliert werden können. Im zweiten Schritt wird darauf aufbauend ein UI entworfen, um nicht nur das Aussehen der Anwendung zu definieren, sondern – idealerweise – auch ein Interaktionskonzept festzulegen, das definiert, wie die Interaktion mit der Anwendung umgesetzt werden soll. An dieser Stelle müssen der Kontext, in dem die Anwendung bedient werden soll, und die Möglichkeiten der Benutzer berücksichtigt werden, die sich z. T. aus den erfassten Anforderungen ergeben. Klassische UI-Entwürfe tendieren jedoch oft dazu, eher statische Beispiele zu sein, was konträr zur heutigen

Art der Interaktion steht – gerade Gestensteuerung ist dynamisch und enthält sowohl zeitliche als auch räumliche Komponenten. Im modernen UI-Entwurf hingegen setzen Designer häufig auf einfache Prototypen z. B. aus Papier oder anderen gängigen Büromaterialien, die zwar nicht programmiert werden und somit keine Logik enthalten, jedoch einen ersten, günstigen Blick auf die Anwendung und ihre Bedienung ermöglichen. Die Grundlage hierfür ist industrielles Produktdesign (Buxton, 2007). Aufbauend auf erfassten Anforderungen und erstellten UI-Entwürfen können Software-Ingenieure dann die Anwendung implementieren, die anschließend noch von den Verantwortlichen für Qualitätssicherung getestet wird.

Die in frühen Phasen der Software-Entwicklung getroffenen Entscheidungen nehmen aber mit den zahlreichen neuen Möglichkeiten, mit Computern zu interagieren, und der steigenden Bedeutung von UX eine zentrale Rolle ein. Aus den erfassten Anforderungen müssen die Rahmenbedingungen für die Interaktion abgeleitet werden, aus denen sich dann verschiedene potentielle UIs ergeben. Diese wiederum dienen als Entscheidungsgrundlage für die in der Implementierung eingesetzten Komponenten: Zum einen sind passende Sensoren nötig, die die Daten zur Identifikation der unterschiedlichen Eigenschaften von Gesten zuverlässig lieferen, zum anderen müssen passende Algorithmen ausgewählt und implementiert werden, die die gelieferten Daten analysieren und die ausgeführten Gesten letztlich erkennen. Gestenbasierte UIs unterscheiden sich wie oben dargestellt von klassischen WIMP-UIs insbesondere durch die Verarbeitung, die unterschiedlichen Eigenschaften, die eine weiterführende Bedeutung für die Anwendung haben können, sowie die Ungenauigkeiten, mit denen Nutzer Gesten ausführen. Hier zeigt sich somit ein wesentlicher Unterschied zu klassischen WIMP-UIs: Während die technische Basis von WIMP-UIs weitgehend gleich ist und lediglich Komponenten der Oberfläche ausgetauscht werden, hängt Gestensteuerung z. T. direkt von der verwendeten Hardware und den eingesetzten Erkenneralgorithmen ab. Spätere Änderungen an ebendiesen Entscheidungen haben dann u. U. weitreichende Folgen und gehen mit größeren Anpassungen am bestehenden System einher, die wiederum mehr Aufwand bedeuten.

Der dargestellte SE-Zyklus und die damit verbundene Vorgehensweise machen aber deutlich, dass sich gerade an der Grenze vom UI-Design hin zur Implementierung eine kritische Stelle findet. Software-Ingenieure müssen die Anforderungen an die Anwendung verstehen, die natürlichsprachliche Definition korrekt interpretieren und anschließend eine Verbindung zu den von Designern erarbeiteten UI-Entwürfen und Prototypen herstellen. Mit diesen Informationen müssen sie dann die technische Basis ableiten und eine Anwendung implementieren, die den gesetzten Erwartungen entspricht. Anschließend muss die Anwendung im Rahmen der Qualitätssicherung getestet werden. Basis für die Definition der Testfälle sind wieder die in frühen Phasen erarbeiteten Artefakte, entsprechend müssen auch die Tester diese verstehen und korrekt interpretieren. Die dargestellten Unterschiede zwischen moderner Gestensteuerung und klassischen WIMP-UIs machen aber deutlich, dass die Interaktion explizit betrachtet und insbesondere zwischen den Beteiligten kommuniziert werden muss. Davon ausgehend, dass jedes Teammitglied eine spezialisierte Ausbildung erhalten und jede der Entwicklungsphasen einen ei-

Phase	Artefakte	Probleme
Requirements Engineering	Oft in natürlicher Sprache und als Fließtext verfasste Dokumente	Mehrdeutigkeit natürlicher Sprache, exakte Beschreibung der Interaktion nur schwer möglich
UI-Design	UI-Mock-Ups, graphische Vorlagen, Click-Dummys, Papierprototypen	Oft statisch, moderne Interaktion ist jedoch dynamisch; häufig nur grobe Richtlinie, keine feste Vorgabe
Engineering	Quell-Code, Applikation	Interpretation und Umsetzung der o. g. Artefakte
Qualitätssicherung	Testfälle	Interpretation der o. g. Artefakte zur Sicherstellung, dass die programmierte Anwendung der Spezifikation entspricht

Tabelle 2.1: HCI-Probleme in den Phasen der Software-Entwicklung

genen Fokus hat, ergeben sich unterschiedliche Anforderungen und Interessen an die im Rahmen des Entwicklungsfortschritts erzeugten Dokumente und Artefakte. Darüber hinaus ergeben sich unterschiedliche Herangehensweisen, um die in den jeweiligen Phasen wesentlichen Informationen zu erarbeiten.

Damit prallen verschiedene Interessen aufeinander, die sich aus den verschiedenen Perspektiven und Arbeitsweisen der in den jeweiligen Phasen verantwortlichen Akteure ergeben. Tabelle 2.1 fasst einige der entstehenden Probleme mit HCI-Bezug zusammen. Im Kern handelt es sich um Schwierigkeiten in der Kommunikation zwischen Projektbeteiligten, die u. a. durch die Übersetzung natürlichsprachlicher Dokumente in Anwendungen und Testfälle entstehen, aber auch durch die implizite Definition unterschiedlicher Aspekte einer Anwendung in UI-Prototypen. Der vorhandene Interpretationsspielraum in derartig definierten Interaktionskonzepten führt zu Missverständnissen zwischen allen Projektbeteiligten und eröffnet Raum für Anwendungsdefekte. Kommunikationsschwierigkeiten sind aber ein inhärentes Problem zwischenmenschlicher Beziehungen und die erfolgreiche Arbeit in einem Projektteam wird dadurch beeinflusst, dass alle Beteiligten in der Lage sind, ihr spezielles Wissen im Sinne des gemeinsamen Projektziels einzubringen, darzustellen und für alle Beteiligten verständlich zu formulieren (Steinheider et al., 2009). In der Software-Entwicklung sind Artefakte wie z. B. Skizzen, Modelle oder Diagramme zentrale Hilfsmittel, um den gemeinsamen Verständigungsprozess zwischen allen Beteiligten erfolgreich zu gestalten.

2.1.2 Interaktionsdesign

Davon ausgehend, dass sich das eigentliche Problem an der Schnittstelle von Design und Implementierung auftut, lohnt ein Blick in die Arbeit von Designern, die die Grundlage für die nachgelagerten Tätigkeiten bildet. Natürlich ist modernes SE nicht durch eine strikte Trennung aller Phasen, Arbeitsschritte und beteiliger Personen geprägt – in agilen Methoden, die ein iteratives Vorgehen propagieren, ist der Einbezug aller Teammitglieder essentieller Bestandteil. Dieses Vorgehen ermöglicht Zweierlei: Zum einen können die unterschiedlichen Facetten des Projekts von spezialisierten Fachkräften er-

arbeitet werden, zum anderen können die anderen Teammitglieder in allen Phasen partizipieren und so Einblick in alle Aspekte der Anwendung gewinnen. Folglich haben die unterschiedlichen Mitglieder des Entwicklungsteams so jederzeit Einblick in den Gesamtfortschritt und können ihre Sichtweise bzw. die Konsequenzen für ihre fachlichen Schwerpunkte vertreten. Gleichzeitig bleibt aber die Verantwortung für die Arbeiten und die erstellten Artefakte der einzelnen Phasen bei den jeweiligen Spezialisten.

Buxton (2007) beschreibt den Designprozess als *Designtrichter*[1] (engl. *Design Funnel*). Das finale Produkt – oder vielmehr, der finale Prototyp, der zum Produkt werden wird – wird in einem iterativen Prozess entwickelt, bei dem verschiedene Ideen erarbeitet und evaluiert werden. Zu Beginn existieren viele vage, unterschiedliche Ideen, aus denen dann in mehreren Versuchen – idealerweise mit Nutzern – einige gestrichen und andere weiterentwickelt werden, bis letztlich eine übrig bleibt. Aus einem breiten Spektrum potentieller Lösungen verdichtet sich das Konzept, das letztlich zum Produkt wird, immer weiter, daher die Analogie zum Trichter. Anders ausgedrückt ist der Designprozess ein mehrfach verzweigter Baum (vgl. Abb. 2.3), dessen Zweige unterschiedliche Ideen darstellen, die teils parallel entwickelt werden und aus denen sich dann der interessanteste Zweig herauskristallisiert (Greenberg et al., 2011). Im Rahmen dieses Prozesses werden verschiedene Techniken eingesetzt (vgl. Kap. 3), die in einen oder mehrere finale Prototypen münden, von denen letztlich einer zur Implementierung ausgewählt wird.

[1] Greenberg et al. (2011) offenbaren allerdings, dass die Grundidee auf Pugh (1991) zurückgeht.

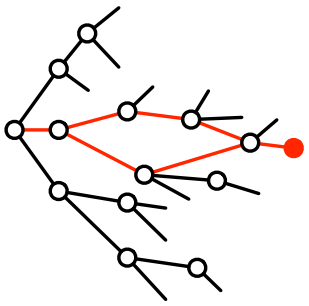


Abbildung 2.3: Verschiedene Konzepte, die beim UI-Design erarbeitet, verworfen und zum finalen Prototyp weiterentwickelt werden; in Rot der finale Prototyp sowie der Weg dorthin (Quelle: Eigene Darstellung nach Greenberg et al. (2011))

Während sein Fokus klar auf der Arbeit von Designern liegt, betont Buxton (2007) auch explizit, dass alle in die Produktentwicklung involvierten Abteilungen in den Designprozess einbezogen werden sollten, aber die Verantwortlichkeiten klar geregelt sein müssen. Abb. 2.4 stellt die einzelnen Phasen nach Buxton (2007) dar und verdeutlicht, dass an der Schnittstelle zwischen Design und Engineering ein wesentlicher Schritt im gesamten Entwicklungsprozess erreicht wird und hier die Verantwortung an ein anderes Teammitglied bzw. eine andere Abteilung abgegeben wird. Dass sowohl das Management als auch die Abteilungen für Vertrieb und Marketing in den Gesamtprozess einbezogen werden müssen, ist im Rahmen seiner Philosophie nur konsequent, spielt aber für die Zielstellung dieser Arbeit keine Rolle. Auf

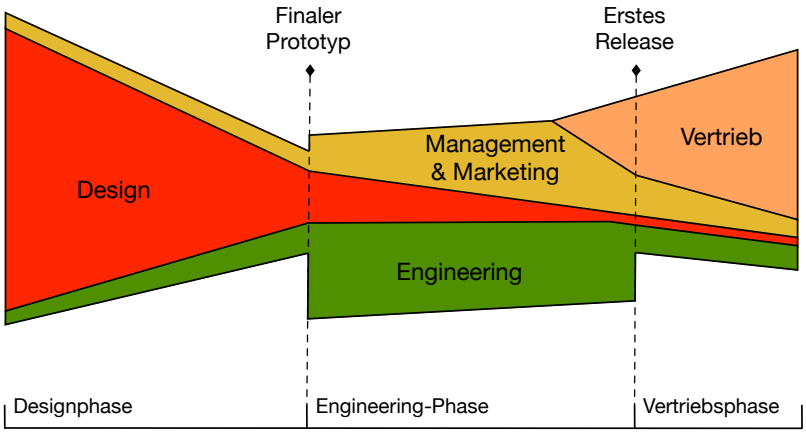


Abbildung 2.4: Der Design Funnel nach Buxton (2007) (Quelle: Eigene Darstellung nach Greenberg et al. (2011))

eine detaillierte Darstellung wird daher an dieser Stelle verzichtet und der Fokus auf Design und Engineering gelegt. Eine strikte Trennung der einzelnen Phasen und insbesondere der Ausführenden voneinander ist also nicht mehr gegeben – stattdessen steht fortwährende Kooperation im Vordergrund, die aber wie bereits dargestellt mit mehr Kommunikation einhergeht.

In der Designphase ragen zwei wesentliche Arten von Artefakten heraus: *Sketches* (dt. Skizzen) und *Prototypen*. Der Übergang vom Sketch zum Prototyp ist u. U. fließend und an dieser Stelle soll es bei folgender, etwas groben Unterscheidung belassen werden: Sketches dienen vor allem zur Präsentation und Kommunikation von Ideen, Prototypen hingegen zur Evaluierung der Benutzbarkeit.[2] Jener Prototyp, der bei der Evaluierung die besten Ergebnisse erzielt, ist folgerichtig derjenige, der ausgewählt wird, um als finale Anwendung implementiert zu werden. Er definiert, was das Engineering-Team implementiert, und die wesentliche Aufgabe dieses Teams ist es, festzulegen, wie das Produkt implementiert wird. Ausgehend vom finalen Prototypen wählen Software-Ingenieure die notwendigen Komponenten aus und bauen die technische Basis auf. Ein wesentlicher Aspekt der Benutzbarkeit einer Anwendung ist aber natürlich die Interaktion, die implizit im Prototypen definiert ist. Software-Ingenieure müssen daraus die Anforderungen an z. B. notwendige Sensoren und Algorithmen ableiten und entscheiden, welche konkreten Komponenten sie verwenden müssen. Ihre Beteiligung bereits in frühen Phasen der Entwicklung ist also sinnvoll, um mögliche Einschränkungen, die sich durch Hard- und Software ergeben, frühzeitig erkennen zu können.

[2] In Kap. 3 wird im Detail auf die Unterschiede eingegangen sowie typische Werkzeuge vorgestellt.

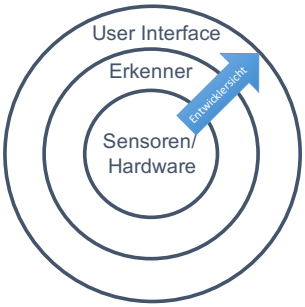


Abbildung 2.5: Typische Entwicklersicht auf eine Anwendung

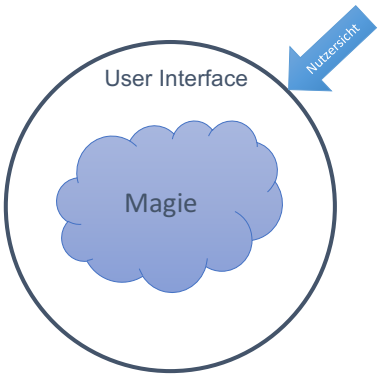


Abbildung 2.6: Typische Benutzersicht auf eine Anwendung

Die unterschiedlichen Schwerpunkte, die Designer und Software-Ingenieure in den Phasen setzen, die sie verantworten, gehen naturgemäß mit unterschiedlichen Methoden und Lösungsstrategien einher. So berichten Wigdor & Wixon (2011, S. 123), dass Software-Ingenieure eher *bottom-up* arbeiten (vgl. Abb. 2.5), also die Möglichkeiten der Hardware und die Fähigkeiten der Nutzer als Basis für die Anwendung nehmen. Ebenso muss dabei auch die Zuverlässigkeit der Algorithmen zur Gestenerkennung eine wesentliche Rolle spielen. Designer hingegen arbeiten eher *top-down* und betrachten die Nutzungsszenarien und den Kontext der Nutzer stärker. Im Gegensatz zu Software-Ingenieuren betrachten sie das System also eher aus Sicht der potentiellen Nutzer, die aber – im wahrsten Sinne des Wortes – nur eine oberflächliche Sicht auf die Anwendung und keine Kenntnisse der Verarbeitungsdetails haben (vgl. Abb. 2.6). Somit spielt keine Rolle, wie die technische Basis funktioniert, wodurch mehr gestalterischer Spielraum möglich ist. Software-Ingenieure entwerfen folglich eher Systeme, die sich an bestehenden orientieren (also den Nutzer dem System anpassen), während Designer eher neue Ideen generieren (also das System dem Nutzer anpassen). Nicht nur im SE, sondern auch in anderen Fachbereichen beeinflussen Faktoren wie eine eher ingenieurs- oder designorientierte Ausbildung die Lösungen, die vorgeschlagen werden (Agogué et al., 2015). Dazu kommt, dass sich viele Aspekte interaktiver Applikationen – insbesondere schwammiger Faktoren wie UX – sich nicht mit klassischen Ingenieursaktivitäten messen lassen. Iteratives Vorgehen und konstantes Testen sind entsprechend wichtig (Blandford, 2013), gehen jedoch mit gesteigertem Aufwand einher, der sich mit den Mitteln und Werkzeugen, die Designer typischerweise einsetzen, minimieren lässt.

Die unterschiedliche Herangehensweise von Software-Ingenieuren und Designern hat für die Definition der Interaktion mit der Anwendung Folgen. So unterscheiden analog dazu auch Nielsen et al. (2003, 2004) zwei Ansätze, um passende Gesten für eine Anwendung zu identifizieren: technologie- und nutzerbasiert. Im technologiebasierten Ansatz werden mögliche Gesten in Abhängigkeit von vorhandener Hardware, Sensoren oder Algorithmen zur Erkennung definiert. Die Möglichkeiten von Gestensteuerung und die Menge der potentiellen Gesten werden so eingeschränkt, ohne dass die Anwendung oder ihre Nutzung als solche betrachtet wurde. Im nutzerbasierten[3] Ansatz hingegen werden Gesten durch ein experimentelles Vorgehen mit potentiellen Anwendern und Prototypen getestet. Die Autoren berichten, dass ein technologiebasierter Ansatz zwar einfacher ist, ein nutzerbasierter aber die für den Nutzer und den jeweiligen Anwendungsfall passenderen Gesten identifiziert – was aber natürlich nicht bedeutet, dass sich diese dann auch leichter implementieren lassen. Im Gegenteil, den Anforderungen der Nutzer, die durch den jeweiligen Anwendungskontext geprägt sind, steht dann der Aufwand der technischen Implementierung gegenüber. Nielsen et al. (2004) kommen zu dem Schluss, dass nur ein experimentelles Vorgehen zur direkten Evaluierung der Gestensteuerung Sinn macht. Sie gehen jedoch nicht auf Details ein und lassen z. B. eine Beschreibung vermissen, wie sie Gesten erfassen und für die weitere Verwendung spezifizieren[4] oder wie sie konkret bei den unterschiedlichen Schritten in ihrem Prozess vorgehen.

[3] Aigner et al. (2012) weisen darauf hin, dass es bei einem nutzerbasierten Vorgehen zwei grundverschiedene Ansätze gibt, die eingesetzt werden können: Gesten können aus der Beobachtung von Benutzern abgeleitet werden (*Eliciting*) oder unter Berücksichtigung verschiedener Faktoren dediziert entwickelt werden (*Designing*).

[4] Die Beispiele der Autoren lassen den Schluss zu, dass sie Gesten natürlichsprachlich beschreiben.

Dass letztlich ein technologiegetriebener Ansatz auch kommerziell nicht erfolgreich sein kann, erläutert Bosch (2016). Neue Technologie wird heute am Markt immer schneller verfügbar gemacht, wodurch eine Differenzierung gegenüber Mitbewerbern anhand technologischer Merkmale immer schwerer fällt. Es setzen sich eher Anbieter durch, die die Anforderungen ihrer Kunden bzw. Nutzer besser verstehen und umsetzen. Daraus ergibt sich zwangsläufig, dass genaueres Wissen über die Zielgruppe ein Wettbewerbsvorteil ist. Dieses Wissen lässt sich durch ein experimentelles Vorgehen[5] akquirieren, wie es Designer vorleben. Demgegenüber steht, dass ein gemischtes Vorgehen – bei dem ein technologiegetriebener Bottom-Up-Ansatz und ein nutzerorientierter Top-Down-Ansatz parallel eingesetzt werden – wohl die besten Ergebnisse bringt, da sie sich gegenseitig vor potentiellen Fallstricken bewahren können (Wigdor & Wixon, 2011).

[5] Auch Bosch (2016) geht allerdings nicht im Detail darauf ein, wie ein derartiges Vorgehen umgesetzt werden soll.

Damit schließt sich der Kreis – ein expliziter Designprozess, der von Fachkräften verantwortet wird, ist für den Erfolg einer Anwendung notwendig, jedoch müssen auch Software-Ingenieure frühzeitig ihre Schwerpunkte einbringen können. Gleichzeitig ermöglicht dieses Vorgehen aber auch, unterschiedliche Ideen und Ansätze mit (potentiellen) Benutzern evaluieren zu können. Der Design Funnel zeigt somit drei wesentliche Dinge:

- Design und Engineering sind durchgehend beteiligt, übernehmen aber in verschiedenen Phasen die Verantwortung: Designer definieren, *was* implementiert wird, und Ingenieure, *wie* implementiert wird

- Es gibt einen Zeitpunkt, an dem entschieden wird, dass ein bestimmter Prototyp zu einer Implementierung wird, und die Verantwortung für das weitere Vorgehen vom Design ans Engineering übergeben wird

- Der Prototyp, der letztlich zur Implementierung ausgewählt wird, stellt eine Art Vertrag zwischen Design und Engineering dar

Dieser Prototyp muss alle notwendigen Aspekte definieren, damit die verantwortlichen Ingenieure die Entscheidungen für die technische Basis treffen und das Produkt implementieren können. Dazu gehört insbesondere, wie die Interaktion mit der Anwendung gestaltet ist, denn gerade hierfür müssen geeignete Hard- und Software-Komponenten ausgewählt werden. Experimentelle Prototypen bleiben bei der Definition der Interaktion und insbesondere von Gestensteuerung aber eher schwammig – die notwendigen Informationen sind implizit im Prototyp festgelegt, werden aber nicht explizit dokumentiert. Zwar wird die Notwendigkeit einer Dokumentation der Gesten von Designern und Entwicklern allgemein anerkannt, in der Praxis jedoch weitgehend ignoriert (Altakrouri & Schrader, 2015) – vielleicht auch, weil passende Standards fehlen. Generell sind die Ergebnisse früher Phasen der Entwicklung, insbesondere die Arbeit von Designern, oft (gewollt) ungenau, wodurch für die in späteren Phasen notwendigen Entscheidungen Fragen offen bleiben. Diese Ungenauigkeit ist aber Teil des kreativen Prozesses (Buxton, 2007).

2.1.3 Spezifikation von Interaktion

Letztlich ist eine exakte Spezifikation der Anwendung spätestens zum dem Zeitpunkt notwendig, an dem ein Prototyp zu einem Produkt wird, um eine korrekte Implementierung zu gewährleisten. Auch für eine weitergehende Diskussion, insbesondere über mehrere Workshops o. ä. Formate hinaus, ist eine genaue Definition der Anwendung und ihrer Komponenten hilfreich. Die Interaktion mit der Anwendung im Allgemeinen und Gestensteuerung im Speziellen bilden keine Ausnahme. Dabei können zahlreiche Mittel eingesetzt werden, um Gesten zu beschreiben: Textuelle Beschreibungen, graphische Notationen/Bilder und hybride Formen ermöglichen unterschiedliche Ansätze, die mehr oder weniger gut auf die Anforderungen verschiedener Projektbeteiligter eingehen (vgl. Kap. 6). Eine starke Formalisierung, wie sie in vielen Modellierungssprachen und generell in eher ingenieursgetriebenen Phasen üblich ist, stört aber den kreativen Fluss und ist in Designphasen eher hinderlich (Shipman & Marshall, 1999).

Demgegenüber steht, dass eine schwache Formalisierung mit einfachen Regeln nicht hinderlich sein muss und vor allem für die Kommunikation zwischen Projektbeteiligten mit unterschiedlichem technischen Hintergrund genutzt wird. So folgen beispielsweise User Stories, die ein weitverbreitetes Mittel in der modernen Software-Entwicklung sind, einem genau definierten Aufbau und beschreiben zwar die einzelnen Funktionalitäten aus Sicht der Anwender (Cohn, 2004), lassen jedoch die Details der Interaktion außen vor. Die Domänenspezifische Sprache (engl. *Domain Specific Language* (DSL)) Gherkin ermöglicht die Erstellung von Testszenarien mit einer pseudo-natürlichen

Syntax, die ebenfalls einem genau definierten Aufbau mit passenden Schlüsselwörtern folgt, es aber dennoch ermöglicht, dass Projektbeteiligte ohne technisches Hintergrundwissen sie verstehen und formulieren können (Wynne & Hellesoy, 2012). Unterschiedliche Frameworks setzen zur Formulierung und automatischen Ausführung von Testfällen auf Gherkin, fokussieren aber in der Regel WIMP-UIs und bieten keine oder nur sehr eingeschränkte Möglichkeiten, Gesten zu beschreiben, selbst wenn sie auf Geräte ausgerichtet sind, für die Gestensteuerung elementar ist, wie z. B. für mobile Geräte (Hesenius et al., 2014a). Interessant ist, dass beide Ansätze auf natürlicher Sprache aufbauen, um sie für technische Laien verständlich zu machen.

Sprachen zur Spezifikation interaktiver Anwendungen dienen verschiedenen Zwecken. Nach Dumas et al. (2010) können Spezifikationssprachen[6] in das in Abb. 2.7 abgebildete Kontinuum eingeordnet werden. Je weiter sie auf der rechten Seite zu finden sind, desto mehr verlieren ihre formale Striktheit. Aktuelle Ansätze zur Modellierung und Spezifikation der Interaktion sind aber stark auf Entwickler ausgerichtet und finden sich in erster Linie auf der linken Seite von Abb. 2.7. Sie werden eingesetzt, um den Entwicklungsaufwand zu minimieren, indem passende Gestenerkenner einfach konfiguriert werden können, und nicht zur Kommunikation der Interaktion mit anderen Projektbeteiligten. Sie müssen folglich über eine hohe Ausdrucksstärke verfügen und maschinenlesbar sein, während bei Ansätzen auf der rechten Seite der Fokus eher auf der Benutzbarkeit durch Menschen liegt.

[6] Die Autoren fokussieren Spezifikationssprachen für multimodale Anwendungen, jedoch lässt sich die Aussage für jede Art interaktiver Anwendung generalisieren

Aus der Notwendigkeit einer genauen Definition von Gesten zur Kommunikation zwischen Projektbeteiligten ergibt sich auch, dass klar sein muss, was eigentlich über Gesten kommuniziert wird. Kap. 1.1.2 veranschaulicht, dass Gestensteuerung vielfältig ist und je nach Anwendungsfall unterschiedliche Eigenschaften relevant sind. Eben die Definition dieser Eigenschaften und ihrer möglichen Werte ist für die Diskussion und weitere Verarbeitung der Gesten notwendig. Es ist davon auszugehen, dass eine Geste umso genauer spezifiziert wird, je mehr ihrer Eigenschaften festgelegt werden.

Forschung zu UI-Spezifikation im Allgemeinen bewegt sich seit einigen Jahren immer stärker in eine modellgetriebene oder deklarative Richtung (Mijailovic & Milicev, 2013). Dabei fokussieren bestehende Ansätze zur Beschreibung von Gesten im Speziellen entweder rein graphische oder rein textuelle Ansätze (vgl. Kap. 6) und lassen mögliche Vorteile hybrider Lösungen außer Acht – es ist denkbar, dass manche Eigenschaften von Gesten vielleicht besser graphisch dargestellt werden als textuell und andere besser kurz und knapp in einfachen Worten notiert werden. Der bereits dargestellte Fokus auf eher ingenieursgetriebene Phasen der Entwicklung erschwert den Einsatz bestehender Notationen zusätzlich.

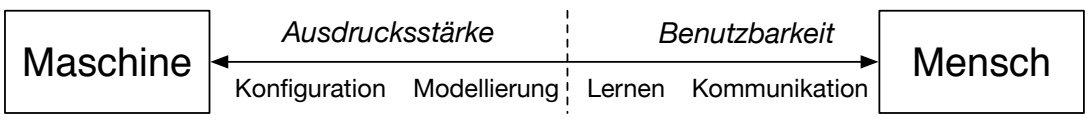


Abbildung 2.7: Zweck von Spezifikationssprachen für interaktive Anwendungen (Bildquelle: Eigene Darstellung nach Dumas et al. (2010))

2.2 Zielsetzung

Aus der ursprünglich simplen Verknüpfung von Interaktion und Funktionsaufruf in WIMP-UIs ist heute ein kompliziertes Zusammenspiel aus verschiedenen Informationen geworden, deren Beschreibung in natürlicher Sprache schwierig ist und schnell in Mehrdeutigkeiten mündet, die wiederum zu Fehlinterpretationen führen. Daraus resultieren Missverständnisse, die im weiteren Verlauf zu fehlerhaften Implementierungen, unterschiedlichen Erwartungen und letztendlich zu Frust im gesamten Projektteam führen. Demnach ist eine gemeinsame Sprache aller Projektbeteiligten notwendig, um über Gesten zu sprechen und diese festlegen zu können. Allerdings haben die einzelnen Projektbeteiligten unterschiedliche Schwerpunkte und Interessen, die sich in der Arbeitsweise und somit auch in den verwendeten Methoden und Werkzeugen wiederfinden. Aktuelle Lösungen haben oftmals einen starken Einschlag in Richtung Engineering, da sie ihren Fokus in der Konfiguration von Gestenerkennern setzen. Für die Kommunikation von Gesten sind sie somit eher ungeeignet. Ein entsprechender Mittelweg ist nötig, der auf der einen Seite die Lesbarkeit für alle Projektbeteiligten garantiert, auf der anderen Seite aber auch die Interaktion detailgenau und exakt wiedergeben kann.

Software-Entwickler müssen zur Implementierung der Interaktion verschiedene Entscheidungen treffen. So hängen von den definierten Gesten die zu verwendenden Sensoren ab, die die grundlegenden Daten erfassen, und die notwendigen Algorithmen, die zur Erkennung der Gesten eingesetzt werden. Die in Kap. 2.1 beschriebenen Probleme sind aber oft weniger auf technische Einschränkungen, sondern auf die Kommunikation zwischen den einzelnen Beteiligten im Projektteam zurückzuführen. Zur erfolgreichen Umsetzung einer gestenbasierten Anwendung müssen die verwendeten Gesten folglich genau spezifiziert und in passender Form niedergeschrieben werden können. Nötig ist eine Gestendefinitionssprache, die die Lücke zwischen Definition und Implementierung der Interaktion schließt.

Der Arbeit liegt die Annahme zugrunde, dass Gesten aus Teilgesten zusammengesetzt werden. Einzelne Teilgesten, die nicht weiter aufgeteilt werden können, werden im Folgenden auch als *atomare Gesten* bezeichnet. Eine zusammengesetzte Geste ist dabei mehr als die Summe ihrer Teile – sie setzt diese in Beziehung zueinander und fügt ggf. weitere Eigenschaften hinzu. Eine adäquate Gestendefinitionssprache – im Sinne dieser Arbeit – muss folgende Möglichkeiten bieten:

- Sie erlaubt die Definition atomarer Gesten und ihrer Eigenschaften
- Sie ermöglicht die Komposition von Gesten durch Definition der zeitlichen und räumlichen Beziehung zwischen atomaren Bestandteilen
- Sie lässt die Erweiterung der Komposition um eigene Eigenschaften zu, die nicht schon durch atomare Gesten vorgegeben sind

Zur Lösung der beschriebenen Probleme werden in dieser Arbeit *GestureCards* vorgestellt, eine Notation zur Spezifikation von Gesten. Alle Projektbeteiligten sollen mithilfe von GestureCards gemeinsam Gesten formulieren

und festlegen können, um die Basis für folgende Entwicklungsschritte zu legen. Im Gegensatz zu verwandten Arbeiten, die im Wesentlichen auf rein textuellen oder rein graphischen Ansätzen aufbauen (vgl. Kap. 6), soll im Rahmen dieser Arbeit ein hybrider Ansatz verwendet werden, der graphische und textuelle Elemente in einer Notation kombiniert. Dadurch sollen unterschiedliche Eigenschaften von Gesten mit der jeweils optimalen Art und Weise beschrieben werden, um die Spezifikation und Entwicklung gestenbasierter UIs zu vereinfachen. Der Einsatz der Notation wird in mehreren Studien validiert, die zeigen, dass GestureCards Vorteile gegenüber den getesteten textuellen und graphischen Notationen haben und auch im praktischen Einsatz von Entwicklern gewinnbringend verwendet werden können.

2.3 Aufbau der Arbeit

Die Arbeit ist in fünf Teile gegliedert. Teil I begann mit einem Überblick über HCI, Gestensteuerung und UX und wird mit diesem Kapitel, das die Problem- und Zielstellung definiert, abgeschlossen.

In Teil II werden die Grundlagen der Entwicklung gestenbasierter Anwendungen analysiert. Dabei wird zunächst auf die Arbeitsweise in der Designphase eingegangen und typische Werkzeuge und Methoden näher betrachtet (Kap. 3). Anschließend wird die Implementierung gestenbasierter Anwendungen beleuchtet und verschiedene Entscheidungen, die Entwickler treffen müssen, diskutiert (Kap. 4). Die unterschiedlichen Eigenschaften von Gestensteuerung werden anschließend analysiert (Kap. 5), um daraufhin ihre Umsetzung und Integration in verwandten Arbeiten, die sich mit der Spezifikation von Gesten auseinandersetzen, näher zu betrachten (Kap. 6).

Der im Rahmen der Arbeit entwickelte Lösungsansatz wird in Teil III beschrieben. Zunächst werden GestureCards, die vorgeschlagene Notation zur Spezifikation von Gesten, im Detail vorgestellt (Kap. 7). Es werden die grundlegende Struktur, notwendige Elemente und der genaue Aufbau definiert. Die Notation wird anschließend in drei Studien evaluiert. In einer ersten Studie wird die im Rahmen dieser Arbeit entwickelte hybride Notation zur Beschreibung von Gesten mit einer textuellen Notation in Bezug auf Lesbarkeit und Verständlichkeit verglichen (Kap. 8). Anschließend werden in einer weiteren Studie die Prägnanz und Genauigkeit textueller, graphischer und hybrider Notationen gegenübergestellt (Kap. 9). In einem letzten Schritt wird validiert, wie GestureCards im praktischen Einsatz angenommen werden (Kap. 10).

In Teil IV werden verschiedene Optionen zur graphischen Repräsentation räumlicher Gesten vorgeschlagen. Zunächst werden der Einsatz von animierten Handmodellen und Pfeilen diskutiert (Kap. 11). Anschließend werden die verschiedenen Ansätze in einer Studie verglichen (Kap. 12).

Mit Teil V wird die Arbeit abgeschlossen. Zuerst werden mögliche weitere Anwendungsgebiete für die vorgeschlagene Notation diskutiert (Kap. 13). Nach einem Fazit wird dann ein Ausblick auf weiterführende, interessante Fragestellungen gegeben, die Raum für zukünftige Arbeiten bieten (Kap. 14).

Teil II

Grundlagen und verwandte Arbeiten

3 Sketching und Prototyping

Sketching (engl. *skizzieren*) ist eine wesentliche Aktivität in zahlreichen Fachbereichen, bei der Zusammenhänge und Konzepte auf mehr oder weniger einfache Art und Weise in der Regel zeichnerisch dargestellt werden. Auf ersten Ergebnissen aufbauend kann anschließend das angedachte Produkt Stück für Stück verfeinert und detaillierter definiert werden, ehe es an die eigentliche Produktion oder – im Sinne der Software-Entwicklung – Implementierung geht. Damit ist auch ersichtlich, dass Software-Entwicklung keine Ausnahme darstellt – im Gegenteil, Sketching wird auch im SE für unterschiedliche Tätigkeiten und Ziele eingesetzt. Baltes & Diehl (2014) zeigen, dass Software-Entwickler täglich mit selbst sowie von anderen angefertigten Sketches arbeiten, dass Sketches typischerweise eine Lebensdauer von mehreren Tagen haben und dabei häufiger überarbeitet werden, insbesondere wenn sie zur Dokumentation eingesetzt werden. Aus ihrer Studie werden auch die wesentlichen Zwecke von Sketches im SE deutlich: Entwerfen, Erläutern und Verstehen. So ist auch nicht weiter verwunderlich, dass dem Sketching in frühen Phasen des SE eine weitergehende Bedeutung zukommt: Das gemeinsame Sketching von z. B. Prozessmodellen hat wesentlichen Einfluss auf die Entstehung eines gemeinsamen Verständnisses vor allem in fachlich gemischten Teams. Dabei verzichten gerade Software-Entwickler hier in höchstem Maße auf eine starke Formalisierung, die eher als hinderlich angesehen wird (Shipman & Marshall, 1999). Stattdessen greifen sie zu informellen Modellen, die vor allem auf analogen Medien wie Papier und Whiteboard gezeichnet werden, auch wenn insbesondere Elemente, die aus der Unified Modeling Language (UML) stammen, gerne wiederverwendet werden (Baltes & Diehl, 2014). Ungeachtet der augenscheinlichen Präferenz analoger Medien sind in den vergangenen Jahren zahlreiche Anwendungen entwickelt worden, um Designer und Entwickler beim (informellen) Sketching mit digitalen/technischen Hilfsmitteln zu unterstützen (z. B. Calico (Mangano et al., 2008) oder FlexiSketch (Wüest et al., 2013, 2015)), die auch offensichtliche Vorteile haben (z. B. Unterstützung bei der Navigation durch unterschiedliche Sketches und die Begradigung gezeichneter Formen).

Auch im UI-Design hat sich frühzeitig die Erkenntnis durchgesetzt, dass einfache Sketches sehr viel effizienter sind als direkt programmierte Entwürfe. Die *Unfertigkeit* von Sketches sorgt dafür, dass der Eindruck einer vermeintlich vollständigen Lösung vermieden wird und das Entwicklungsteam sich nicht in Details verliert, die insbesondere in frühen Phasen noch keine Rolle spielen sollten, und sich auch nicht zu früh auf eine Lösung festlegt, sondern Alternativen berücksichtigt (Wong, 1992). Das Layout eines UIs lässt sich somit problemlos auf einfache Art und Weise entwerfen: Sketches zeigen dann einen bestimmten Zeitpunkt im Lebenszyklus der Anwendung und definieren das Aussehen zu ebendiesem Zeitpunkt. Allerdings ist das Layout nur ein Faktor des UI, denn auch die Interaktion mit der Anwendung spielt – wie

in Kap. 2 gezeigt – beim Design bereits eine zentrale Rolle und muss daher beim Sketching berücksichtigt werden können.

Vom Sketching ist der Weg zum Prototyping in der Regel nicht weit, insbesondere mit Blick auf die Interaktion, denn Prototypen ermöglichen die Evaluierung von Konzepten, die mithilfe von Sketching erstellt wurden. Dabei verschwimmen allerdings die Begrifflichkeiten: Während verschiedene Arten von Prototypen unterschieden werden, die sich sowohl in Darstellung als auch Detailgenauigkeit und eingesetzter Technik immer mehr der eigentlichen Anwendung nähern, wird gerne auch jede Art von Sketching als Prototyping bezeichnet. Coyette et al. (2007) unterteilen z. B. in NoFi-, LoFi-, MeFi- und HiFi-Prototypen,[1] die sich in erster Linie in ihrer Genauigkeit unterscheiden. Während der Design-Phase kommt vorwiegend LoFi-Prototyping zum Einsatz, da hierfür die gleichen Werkzeuge wie fürs Sketching eingesetzt werden. Eine Abgrenzung beider Aktivitäten erscheint daher sinnvoll. Im Folgenden ist mit Prototyping die Erstellung von LoFi-Prototypen gemeint.

[1] Die Kürzel stehen für verschiedene Abstufungen der *Wiedergabetreue* (engl. *fidelity*): *no*, *low*, *medium* und *high fidelity*.

Sketches und Prototypen sind wichtige Hilfsmittel beim Entwurf und bei der ersten Evaluierung einer Anwendung. Sie müssen somit mindestens als Teil der Spezifikation einer Anwendung verstanden werden. Gleichzeitig stellt sich die Frage, wie Interaktion, vor allem Gestensteuerung, hier dargestellt werden kann und wo ggf. die Grenzen liegen, die eine genauere Spezifikation mit einer passenden Notation wie GestureCards notwendig machen. Denkbar sind zwei Einschränkungen: Die Darstellung dynamischer Komponenten von Gesten in einer statischen Darstellung und die genaue Definition aller Eigenschaften einer Geste.

Im Folgenden soll zunächst zwischen Sketches und Prototypen unterschieden werden, um ihre Bedeutung im gesamten Entwicklungszyklus herauszustellen (Kap. 3.1). Anschließend wird ein Überblick über typische Werkzeuge und Methoden gegeben (Kap. 3.2), mit denen in frühen Phasen der Entwicklung potentielle Anwendungen entworfen und evaluiert werden können. Der Plural ist an dieser Stelle bewusst gewählt, denn das Ziel früher Phasen ist insbesondere die Generierung unterschiedlicher Lösungsansätze. Daraufhin wird zusammengefasst, wie sich die Interaktion mit der Anwendung in Sketches und Prototypen wiederfindet und wie Interaktion erfasst, definiert und evaluiert werden kann (Kap. 3.3).

3.1 Begriffsklärung und Abgrenzung

Die Begriffe – sowohl Sketching/Sketch als auch Prototyping/Prototyp – werden unterschiedlich verwendet und sind in der Literatur mindestens verwandt, wenn nicht sogar – je nach Autor – synonym. Manche verstehen darunter Prozesse, andere Artefakte (Johnson et al., 2009). Gerade im UI-Design sind aber beides wesentliche Elemente, deshalb wird im Folgenden eine Abgrenzung beschrieben, die im Rahmen dieser Arbeit Verwendung findet.

3.1.1 Sketching

Fish & Scrivener (1990) führen aus, dass sich die Grundlagen des Sketchings seit dem Aufkommen im Mittelalter kaum verändert haben. Im Fokus steht die *unordentliche* Abbildung, um Ideen zu visualisieren[2] (Fish & Scrivener, 1990) – eine logische Konsequenz aus der Trennung von Entwurf und Herstellung im Mittelalter (Buxton, 2007), die nun die Darstellung und Kommunikation von Ideen und grundlegender Konzepte notwendig machte. Design-Prozesse starten demnach mit einer Menge vager, unvollständiger Ideen, die als Basis für die eigentliche Produktumsetzung dienen und durch weiteres Sketching verfeinert und durch neue ergänzt werden (Fish & Scrivener, 1990) – im Wesentlichen der Vorgang, den Buxton (2007) als Design Funnel beschreibt (vgl. Kap. 2.1.2). Sketching wird allerdings häufig auf *Stift und Papier* reduziert und sicher spielen Zeichnungen eine große Rolle, jedoch trifft das nicht den Kern der Sache: Vielmehr geht es um das grundlegende Konzept, dass das zu entwickelnde Produkt in unterschiedlichen Variationen mit einfachen Mitteln nachgestellt wird, um neue Ideen zu generieren (Buxton, 2007). Die Wahl des konkreten Werkzeugs ist dabei unerheblich und sollte sich an Zielstellung, Stand im Projekt und Art der Anwendung orientieren. Kap. 3.2.1 stellt unterschiedliche Möglichkeiten vor.

[2] „Sketching, or the production of untidy images to assist in the development of visual ideas [...].“ (Fish & Scrivener, 1990)

Die Aufgaben von Sketches innerhalb eines Entwicklungsprozesses sind somit recht schnell zusammengefasst: Sie kommunizieren und dokumentieren Ideen, entweder zum eigenen Gebrauch als Gedankenstütze oder als Basis zur Diskussion im Team. Darüber hinaus eignen sie sich als Spielwiese, um neue Ideen zu entwickeln und Konzepte losgelöst von technischen Restriktionen auf einer rein konzeptuellen Ebene durchzuspielen. Durch die konstante Verfeinerung unterschiedlicher Details ergeben sich somit aus zahlreichen Sketches eine Menge von Entscheidungen, die letztlich zum finalen Produkt führen. Ebendiese (Design-)Entscheidungen sind für nachgelagerte Phasen besonders interessant, denn sie beeinflussen notwendige Elemente und den technischen Aufbau der Lösung.

Sketches dienen also einem recht genau definiertem Zweck und ordnen sich ebendiesem in Aufbau und Struktur unter. Somit haben Sketches auch einige grundlegende Eigenschaften und wesentliche Merkmale, die Buxton (2007, S. 111f) wie im Folgenden dargelegt beschreibt:

- Sketches sind schnell (*quick*) erstellt oder wirken zumindest so. Baltes & Diehl (2014) zeigen, dass im SE Sketches oftmals in weniger als einer Stunde erstellt werden und bevorzugt analoge Medien (Papier oder Whiteboards) zum Einsatz kommen. Dadurch entsteht ein Fokus auf das eigentliche Problem und für die aktuelle Projektphase unwesentliche Faktoren können ausgeblendet werden. Durch die unsaubere Zeichnung sind Sketches eindeutig als solche erkennbar – eine deutliche und klare Sprache (*clear vocabulary*), die aussagt, dass es sich hier keineswegs um eine Implementierung, sondern eine konzeptuelle Visualisierung einer potentiellen Lösung handelt. Damit sind vor allem Ungenauigkeiten in der Ausführung gemeint, die diesen Aspekt unterstreichen[3].

[3] Selbst gängige digitale Tools zum Erstellen von Wireframes orientieren sich hier an Handzeichnungen, obwohl technische Hilfsmittel problemlos perfekte Sketches ermöglichen (vgl. Kap. 3.2.1.3)

- Mit der Geschwindigkeit gehen weitere Eigenschaften einher. Sketches können genau dann erstellt werden, wenn sie gebraucht werden. Sie sind rechtzeitig fertig (*timely*) und beinhalten auch genau die Informationen und Detailtiefe, die zu diesem Zeitpunkt innerhalb der Entwicklung notwendig sind. Darüber hinaus sind sie günstig (*inexpensive*) herzustellen – ein Sketch kostet nicht viel. Überhaupt sollten Kosten kein Hindernis dafür sein, ein denkbares Konzept der Anwendung zu durchdenken, insbesondere in sehr frühen Phasen des Designs. Das schließt eher aufwendige Arbeiten, die schon auf evtl. wiederverwendbaren technischen Umsetzungen basieren, aus. Generell sollten Sketches keine Implementierung sein, sondern lediglich ein wegwerfbares (*disposable*) Konzept darstellen, was wiederum erzwingt, dass Sketches – wie bereits dargelegt – günstig herzustellen sein müssen. Im Umkehrschluss bedeutet das aber nicht, dass Sketches wertlos sind. Ihr Wert besteht vielmehr darin, dass sie weggeworfen werden können, ohne weiteren Schaden anzurichten – ihr Gewinn liegt im Prozess sowie im durchdachten Konzept und weniger im Artefakt als solchem.

- Sketches existieren selten isoliert voneinander. Im Rahmen eines Projekts entstehen typischerweise reichlich viele (*plentiful*) Sketches, die unterschiedliche Aspekte beleuchten und je nach Projektstand in ihrer Detailtiefe variieren (*appropiate degree of refinement*). Die Bedeutung einzelner Sketches ist dabei in der Regel im Kontext mit den anderen Sketches zu sehen, sodass sich das im Sketch kommunizierte Konzept eingrenzen lässt. Generell sind Sketches ungenau und geben so die Freiheit, den Inhalt unterschiedlich zu interpretieren (*distinct gesture*) – ganz im Gegensatz zu z. B. präzisen technischen Zeichnungen. Sie sind minimalistisch (*minimal detail*) und werden auf die Informationen reduziert, die es zu kommunizieren gilt. Somit machen einzelne Sketches nur im Kontext des gesamten Projektes Sinn.

- Die unterschiedliche Detailtiefe von Sketches geht mit einer gewissen Mehrdeutigkeit (*ambiguity*) einher, die durch den im Team bekannten Projektkontext aber vernachlässigt werden kann. Die Mehrdeutigkeit von Sketches bringt aber auch essentiellen Mehrwert, da sie den notwendigen Interpretationsspielraum erlaubt, um Diskussionen anzuregen und so neue Konzepte zu besprechen und bestehende Ideen zu überarbeiten. Somit schlagen Sketches eher Lösungen vor anstatt sie zu definieren und sollen zum Erforschen alternativer Lösungswege animieren (*suggest and explore rather than confirm*).

Baltes & Diehl (2014) zeigen, dass Sketching eine wesentliche Tätigkeit innerhalb des SE ist, was jedoch insbesondere mit Blick auf die angesprochene Mehrdeutigkeit von Sketches nicht direkt intuitiv erscheint. Gerade die exakte Erfassung und Planung sind ja grundlegende Eigenschaften sauberen Engineerings. Auch Buxton (2007) kommt zu dem Schluss, dass der Vorteil von Mehrdeutigkeit nicht offensichtlich ist, obschon er Sketching eher aus einer Designperspektive betrachtet. Tatsächlich stellt er es so dar, dass die Mehrdeutigkeit und die daraus entstehenden Lücken erst die kreativen Impulse ermöglichen, die zur Entwicklung neuer Ideen und Lösungsansätze notwendig

sind. Sie fordern unterschiedliche Interpretationen heraus und ebnen somit den Weg für Diskussion, Kommunikation und Kreativität. Letztlich ist hier der Wert zu finden, den Sketching ausmacht. Damit einher geht aber auch die notwendige Fähigkeit, mit der resultierenden Unsicherheit umzugehen und Sketches im Kontext des Projekts korrekt zu interpretieren.

Die Mehrdeutigkeiten in Sketches werden im Laufe eines Projekts immer weiter eingegrenzt, sodass mit der Zeit ein Punkt erreicht wird, an dem ein Übertrag in ein formales Modell als finale Dokumentation sinnvoll ist. So wird aus einem einfach skizzierten Geschäftsprozess ein vollständiges Diagramm in einer Notation wie Business Process Modelling Notation (BPMN) oder aus einer groben Architektur ein Komponentendiagramm der UML. Dasselbe gilt im Kern auch für UI-Design, bricht jedoch oftmals an genau dem Schritt, an dem in anderen Bereichen des SE aus Sketches formale, vollständige Modelle werden, ab – oftmals bleibt es bei reinen Sketches, die dann nur als Vorlage und (grobe) Richtungsanweisung dienen. Folgt man aber den Ausführungen von Buxton (2007), so gehört zum UI-Design weit mehr als die Definition des Layouts. Weitere Aspekte müssen berücksichtigt werden, die jedoch in einfachen Sketches oftmals lediglich implizit definiert werden können. Dazu zählt insbesondere die Interaktion, die aber in modernen Anwendungen dynamisch und nicht statisch ist – ganz im Gegensatz zu einfachen Sketches.

3.1.2 Prototyping

Die Dynamik, die sich immanent in der Interaktion mit einer Anwendung verbirgt, geht in einfachen Sketches auf den ersten Blick verloren, kann aber unter Einsatz der richtigen Mittel integriert werden (vgl. Kap. 3.2). Die Integration von Interaktion erweitert Sketches dahingehend, dass das Verhalten der Anwendung über Zeit nachgestellt und validiert werden kann. So wird aus einem Sketch ein Prototyp. Ein Prototyp ist im Grunde genommen eine Art Vorschau der finalen Anwendung, die bereits die notwendigen Mechanismen zur Interaktion enthält oder zumindest so weit nachstellt, dass sie möglichst nah am endgültigen Produkt sind. Die Integration von Interaktionsmöglichkeiten erlaubt auch, dass ein Prototyp mit Benutzern validiert wird, während Sketches eigentlich nur gezeigt und diskutiert werden können.

Sketches und Prototypen unterscheiden sich also in ihrem Einsatzzweck: Während Sketches in erster Linie ein Werkzeug zur Kommunikation und Weiterentwicklung von Ideen sind, werden Prototypen eingesetzt, um UI-Konzepte zu evaluieren (Buxton, 2007). Aus einem validierten Prototypen erwächst dann über mehrere Iterationen eine Vorstellung, wie die eigentliche Anwendung implementiert werden soll. Somit hat ein Prototyp eine wesentlich größere Verbindlichkeit als Sketches und ist – wie in Kap. 2.1.2 gesehen – das entscheidende Artefakt beim Übergang von Design zum Engineering.

Mit dem unterschiedlichen Einsatzzweck gehen eine Menge verschiedener Eigenschaften einher, an denen Sketches und Prototypen abgegrenzt werden können. Buxton (2007) unterscheidet die in Abb. 3.1 genannten. Ohne auf einzelne im Detail einzugehen, ist eine deutliche Tendenz zu erkennen. Sketches sprechen eher vorläufige Empfehlungen aus und machen Vorschläge,

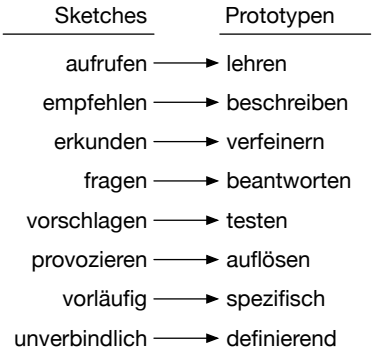


Abbildung 3.1: Abgrenzung von Sketches und Prototypen (Bildquelle: Eigene Darstellung nach Buxton (2007)). Im Vergleich zur Quelle wurde der letzte Punkt bei Prototypen geändert: Buxton (2007) verwendet hier den englischen Begriff *depiction*, der soviel wie *Bildliche Darstellung* bedeutet, aber im Deutschen nicht so kontrastreich zum Begriff *unverbindlich* steht.

können aber daher auch provokant sein. Sie bewegen sich also auf einer eher unverbindlichen Ebene – oder wie Johnson et al. (2009) es ausdrücken: Ein Sketch ist kein Vertrag, sondern ein Vorschlag, der angepasst, gelöscht und als Grundlage dienen kann.[4] Prototypen hingegen beantworten Fragen, beschreiben eine Lösung und sind zur Validierung spezifischer Anforderungen geeignet. Aus ihnen wird sehr wohl eine Art Vertrag, denn eine erfolgreiche Validierung definiert in der Regel eine zu implementierende Funktionalität.

[4] „A sketch is not a contract: it is a proposal that can be modified, erased, built upon." (Johnson et al., 2009)

Hervorzuheben ist allerdings, dass in der gängigen Literatur kein expliziter Unterschied im gewählten Material gemacht, sondern lediglich darauf hingewiesen wird, dass mit steigender Nähe zur finalen Anwendung der Aufwand zur Erstellung von Prototypen wächst. Im Gegenteil: Verschiedene Autoren, z. B. Wong (1992), weisen explizit darauf hin, dass bei sehr exakten Prototypen die Gefahr besteht, dass sie mit der Umsetzung der Anwendung verwechselt werden und ein falscher Eindruck über den Projektfortschritt entsteht. Prototyping hat gegenüber der Arbeit mit reinen Sketches außerdem den Vorteil, dass die Integration von Benutzern in den Entwicklungsprozess, die mit eher abstrakten Sketches keine gute Vorstellung entwickeln können, effizienter gestaltet wird (Nielsen, 1994).

Buxton (2007) führt weiter aus, dass damit auch eine Evolution von Sketches hin zu Prototypen im Laufe des Projekts einhergeht, jedoch eher von einem Kontinuum denn einer scharfen Trennung gesprochen werden muss. Mit fortschreitender Verfeinerung der Sketches in den zahlreichen Iterationen des Designprozesses wird es immer wichtiger, potentielle Fallstricke zu vermeiden. Evaluierung mit Benutzern ist also notwendig und Prototypen haben sich als probates Mittel erwiesen. Abb. 3.2 verdeutlicht, wie zu Beginn des Design Funnels (vgl. Kap. 2.1.2) viele Sketches existieren, die im Laufe der Entwicklung in mehreren Prototypen zusammengefasst und evaluiert werden. Sketches sind jedoch handgezeichnete, unsaubere Skizzen (vgl. Kap. 3.1.1) und das wesentliche Merkmal von Prototypen hingegen ist die Möglichkeit der Interaktion.

Davon ausgehend, dass der Unterschied zwischen Sketches und Prototypen nicht im verwendeten Werkzeug, sondern im Einsatzzweck liegt, muss auch eine Interaktion mit einfachen Prototypen möglich sein. Wenn ein Prototyp aber – wie weiter oben angedeutet – Teil der Spezifikation ist und eine Art Vertrag darstellt, dann müssen alle notwendigen Elemente eindeutig und explizit definiert werden können. Das ist gerade im Falle von Gestensteuerung in Sketches und Prototypen nicht möglich, wie im Folgenden gezeigt wird.

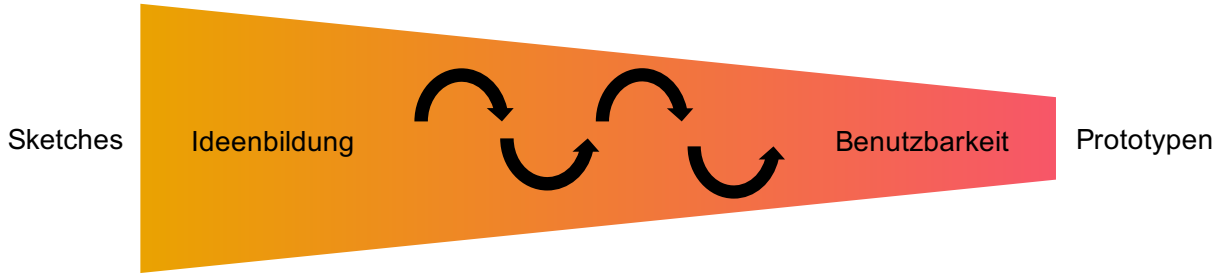


Abbildung 3.2: Entwicklung von Sketches zu Prototypen im Laufe des Design Funnels (Bildquelle: Eigene Darstellung nach Greenberg et al. (2011))

3.2 Werkzeuge für Sketching und Prototyping

Im SE werden Unterschiedliche Werkzeuge und Methoden zum Sketching und zur Erstellung von Prototypen eingesetzt. Im Folgenden sollen zwei grundlegende Werkzeuge und unterschiedliche Ausprägungen näher betrachtet werden, die besondere Bedeutung für das UI-Design haben. Während die grundlegenden Konzepte des Sketchings bereits zu Beginn dieses Kapitels eingeführt wurden, sollen an dieser Stelle konkrete Methoden vorgestellt werden. Anschließend werden verschiedene Arten von Storyboards vorgestellt, die Sketches zu einem Navigationsfluss kombinieren.

3.2.1 Sketches

Sketches lassen sich mit unterschiedlichen Werkzeugen erstellen. Im Folgenden sollen *Scribble Sketches* als grundlegende Variante betrachtet werden. Anschließend werden Sketches mit *Büromaterial* betrachtet, die die interessante Eigenschaft haben, dass sie interaktiv verändert werden können und somit eine weiterführende Dynamik erlauben. Außerdem wird eine digital unterstützte Variante des Sketchings betrachtet: *Wireframing* erlaubt das schnelle Zusammenstellen von UI-Sketches aus vorgefertigten Komponenten.

3.2.1.1 Scribble Sketches

Ideen haben die – mehr oder weniger angenehme – Eigenschaft, teils unerwartet zu kommen. Vielleicht aus einer Diskussion in einem Meeting, in dem ein völlig anderes Projekt besprochen wird, aber interessante Ideen für neue Konzepte entstehen. Manchmal auch einfach durch externe Einflüsse wie z. B. Kinofilme,[5] Bücher oder sonstige Medien. Letztlich ist der konkrete Kontext nicht ausschlaggebend – wesentlich ist, dass die Idee festgehalten werden muss, um sie später wieder aufgreifen zu können.

[5] Marcus (2013) und Figueiredo et al. (2015) listen z. B. verschiedene Filme, die interessante, futuristische HCI-Konzepte zeigen, die mit heutiger Technik z. T. bereits realisiert werden können.

Greenberg et al. (2011) empfehlen *Scribble Sketching* als Methode, um Ideen überall schnell festzuhalten. Mit Blick auf UI-Design bedeutet das insbesondere, die Essenz neuer UI-Ideen festzuhalten und für spätere Verfeinerung quasi zu speichern. Abb. 3.3 zeigt ein Beispiel für eine mobile Anwendung. Auch ohne den Kontext der Anwendung – eine Anwendung zur Bearbeitung von Bildern – zu kennen, werden schnell Elemente deutlich: oben eine Leiste mit unterschiedlichen Elementen; in der Mitte wohl eines, das gerade ausgewählt wurde; und unten eine Leiste mit unterschiedlichen Menus. Scribble Sketches sind also per Definition recht rudimentär, besonders zu Beginn, entwickeln sich aber schnell zu einem grundlegenden Konzept für eine potentielle Anwendung.

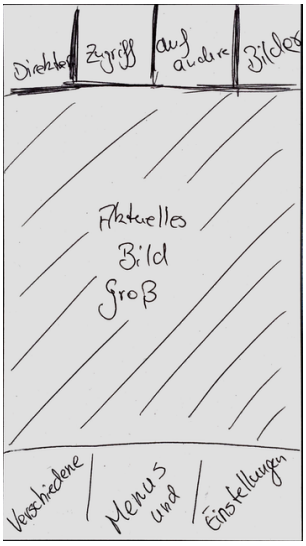


Abbildung 3.3: Ein Scribble Sketch für eine mobile Anwendung

Scribble Sketches fokussieren Geschwindigkeit – Greenberg et al. (2011) empfehlen hier, auf Details weitgehend zu verzichten und möglichst weit zu abstrahieren. Somit wird immer das Detail hervorgehoben, das gerade im Fokus steht. So hebt das Beispiel in Abb. 3.3 die Struktur der einzelnen Elemente – oben eine Übersicht, in der Mitte eine zentrale große Ansicht und unten eine Leiste mit Menüpunkten – hervor, während Abb. 3.4 konkrete

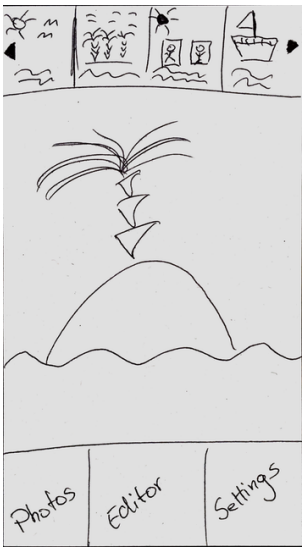


Abbildung 3.4: Eine Verfeinerungsstufe des *Scribble Sketch* in Abb. 3.3

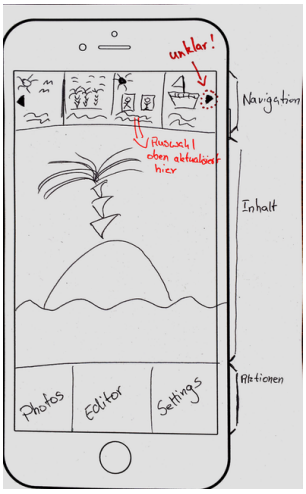


Abbildung 3.5: Der Sketch aus Abb. 3.4 mit einigen Annotationen

Inhalte zeigt: Unten sind die verfügbaren Menüs zu sehen und oben wird durch die Dreiecke links und rechts angedeutet, dass irgendwie – die eigentliche Interaktion bleibt hier verborgen – zu anderen Bildern navigiert werden kann.

Neben der Sammlung von Scribble Sketches für den persönlichen Gebrauch können sie auch während Diskussionen in Meetings nützlich sein. So können z. B. unterschiedliche Versionen der Sketches in Abb. 3.3 und Abb. 3.4 gegenübergestellt oder von einzelnen Teammitgliedern entwickelt werden, um die Auswirkungen unterschiedlicher Darstellungen zu diskutieren oder z. B. Stärken unterschiedlicher Ansätze zu vereinigen und Schwächen zu eliminieren. Dieses Vorgehen entspricht genau dem im Design Funnel nach Buxton (2007) skizzierten Prozess (vgl. Kap. 2.1.2).

Darüber hinaus zeigen Buxton (2007) und Greenberg et al. (2011), wie Scribble Sketches mit zusätzlichen Informationen versehen werden können. Sie beschreiben *Annotationen*: Markierungen, Namen oder kurze Erläuterungen, die in einem räumlichen Verhältnis zu einem Teil des Sketches stehen. Abb. 3.5 veranschaulicht, wie das Beispiel aus Abb. 3.4 mit Annotationen erweitert werden kann. Auf der rechten Seite werden Bereiche der Anwendung beschrieben. Durch die (optionale) Klammer werden die Teile des Bildschirms räumlich definiert und es wird ihnen eine Rolle zugewiesen, die ihre Funktion im Rahmen des Sketches einschränkt. Eine andere Variante zeigt im Prinzip schon Abb. 3.3, nur dass die Annotationen hier schon direkt in den Sketch als Beschriftung der einzelnen Regionen integriert sind. Im direkten Vergleich erscheint diese Version aber durch die fehlenden Inhalte wesentlich abstrakter.

Außerdem wurde der Sketch in Abb. 3.5 mit zwei Pfeilen annotiert. Pfeile sind ein wesentliches Hilfsmittel für Annotationen und können zahlreiche Informationen übermitteln: Betonung eines Elements, Bezüge zu anderen Elementen, Bewegung, Richtungsangabe, Interaktion (Greenberg et al., 2011). Die Beschriftung *unklar!* z. B. betont, dass an dieser Stelle wohl ein Verständnisproblem beim markierten Element besteht. Die Annotation könnte von einem Kollegen eingefügt worden sein und z. B. ein Hinweis auf Diskussionsbedarf beim nächsten Meeting sein. Generell bieten sich Annotationen an, um auch weiterführende Konzepte, die nicht direkt im Sketch zu sehen sind und daher Raum für Interpretationen bieten, genauer darzustellen. Der andere Pfeil, der auf die Auswahl im oberen und die Aktualisierung im mittleren Teil hinweist, deutet eine Möglichkeit zur Interaktion an, bleibt aber weitgehend unspezifisch, wie diese ausgestaltet werden soll. Tatsächlich bieten sich Pfeile auch an, um Dynamik innerhalb des Sketches darzustellen. So könnte mit einem solchen Pfeil dargestellt werden, dass das kleine Bild oben mit einer Geste nach unten gezogen werden kann, um den Bereich der Inhalte zu aktualisieren. Für einfache Gesten mag diese Form der Annotation auch funktionieren, jedoch wird sie bei erweiterten Informationen schnell an ihre Grenzen geraten, z. B. wenn Druck auf den Bildschirm ausgeübt werden soll.

Im Gegensatz zu Annotationen sind Notizen nach Greenberg et al. (2011) zusätzliche Informationen, die kein räumliches Verhältnis zu einem der dargestellten UI-Elemente haben. Die Autoren listen unterschiedliche Möglichkeiten, Notizen einzusetzen: zusätzliche Ideen zu Elementen, die im aktuellen Sketch nicht zu sehen sind (z. B. zusätzliche Hardware); alternative Designop-

tionen; eventuelle Probleme; weitergehende Erläuterungen oder offene Fragen. Notizen dienen auch dazu, lose Gedanken zum Sketch aufzuschreiben.

In Abb. 3.5 wird noch eine weitere Information eingefügt: den Formfaktor des verwendeten Geräts. Der Sketch ist so leichter in seinen Kontext zu integrieren, insbesondere wenn eine Anwendung auf mehreren Geräten laufen soll und dabei unterschiedliche Einschränkungen zu beachten sind (z. B. die gleiche Anwendung auf einem Smartphone, Tablet oder einer Smartwatch). Die Verwendung von Vorlagen bietet sich an, um wiederkehrende Informationen nicht jedesmal neu zeichnen zu müssen (Greenberg et al., 2011).

In den letzten Jahren wurden darüber hinaus zahlreiche Anwendungen entwickelt, um Sketching und insbesondere Scribble Sketching elektronisch zu unterstützen. Dabei werden die grundlegenden Konzepte übernommen und im Kern nur das Medium Papier oder Whiteboard gegen eine elektronische Variante – z. B. Tablet oder digitales Whiteboard – ausgetauscht. Elektronische Sketching-Tools bieten zahlreiche Vorteile beim Sketching im Allgemeinen, aber auch für UI-Sketching im Besonderen. So können gleiche Informationen in unterschiedlichen Sketches erkannt und verknüpft werden, z. B. Entitäten aus Prozessmodellen, die sich in UI-Sketches wiederfinden, um so ein ganzheitliches Bild des Projekts in frühen Phasen zu geben und alle notwendigen Perspektiven zu verknüpfen (Kleffmann et al., 2015a). Besonders interessant sind elektronische Sketching-Tools aber zur Integration moderner HCI-Elemente wie Gestensteuerung (vgl. Kap. 3.3), denn so ließe sich ein UI, das nur als Sketch vorliegt, mit entsprechenden Möglichkeiten zur Bedienung ausstatten. Gleichzeitig sind elektronische Sketching-Tools mit zahlreichen Herausforderungen konfrontiert, die den Fokus dieser Arbeit übersteigen und daher nicht detailliert betrachtet werden. Johnson et al. (2009) bieten hierzu einen weitreichenden Überblick.

Gerade für mobile Anwendungen, die auch von der aktuellen Umgebung abhängig sein können, ist die Integration des aktuellen Kontexts in einen Sketch interessant. Diesen zu zeichnen erfordert allerdings u. U. weitreichende künstlerische Fähigkeiten. Greenberg et al. (2011) stellen dafür eine einfache Alternative vor: die Bearbeitung von Fotos. Dazu werden die notwendigen Information – z. B. das in Abb. 3.4 gezeigte UI – nachträglich in ein Foto einer bestimmten Situation eingearbeitet. Der – scheinbar – höhere Aufwand wird durch unterschiedliche Faktoren wieder ausgeglichen: Fotos sind einfacher gemacht als eine handschriftliche Zeichnung des Kontexts (aufgrund der Verbreitung von Smartphones hat auch jeder in der Regel eine Kamera dabei) und sie lassen sich wiederverwenden, um z. B. unterschiedliche Alternativen des UI zu demonstrieren. Auf diese Art und Weise können Scribble Sketches auch zur Entwicklung von AR-Anwendungen[6] eingesetzt werden.

Scribble Sketches ermöglichen die schnelle Erstellung von UI-Designs. Annotationen und Notizen erlauben die Erweiterung um zusätzliche Informationen. Pfeile können eingesetzt werden, um auf bestimmte Elemente zu verweisen oder dynamische Aspekte anzudeuten. Und hier liegt auch die wesentliche Schwachstelle von Scribble Sketches: Dynamik kann nur angedeutet werden. Interaktion – und insbesondere Gestensteuerung – ist aber zwangsläufig

[6] Ein Beispiel ist bei Hesenius et al. (2018) zu finden.

dynamisch, sodass hier eine Lücke entsteht: Scribble Sketches können kein direktes Feedback geben und somit nicht auf Interaktion reagieren.

Sketches können dennoch auch als Prototyp eingesetzt werden, obwohl Prototypen grundsätzlich interaktiv sein müssen. Die Interaktion kann nachgestellt werden, indem z. B. die verwendeten Sketches ausgetauscht oder mündliche Erläuterungen dazu gemacht werden. Darüber hinaus sind weitere Nachteile zu nennen, bei denen der Aufwand, der zum Scribble Sketching betrieben werden muss, nicht mehr im Verhältnis zu den Vorteilen steht. Scribble Sketches können nur durch Zeichnen neuer oder Löschung bestehender Elemente verändert werden. Dadurch ist eine kurzfristige Änderung nicht möglich – jede Art des *Ausprobierens* sorgt sofort für zusätzliche Arbeitsschritte oder gar völlige Neuerstellung des Sketches. Die Bedienung vieler moderner Geräte, z. B. Wearables, ist außerdem von einer gewissen Haptik abhängig, die mit gezeichneten Sketches nicht nachgestellt werden kann. Dadurch wird insbesondere die Validierung erschwert, bei der die UX auch mit Prototpyen möglichst genau imitiert werden soll.

Abbildung 3.6: Ein Sketch mit Büromaterial

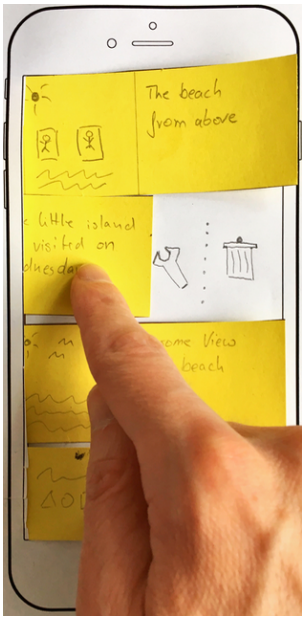


Abbildung 3.7: Interaktion in einem Sketch mit Büromaterial

3.2.1.2 Büromaterial und Bastelbedarf

Zeichnungen kommen also dann an ihre Grenzen, wenn die Interaktion mit einer Anwendung stärker in den Fokus rückt. Greenberg et al. (2011) schlagen als praktische Alternative für Sketching und Prototyping einfaches Büromaterial vor. Insbesondere Klebenotizen, die auf ein Blatt Papier oder ein Flipchart geklebt werden, lassen sich sehr gut verwenden, um schnell Elemente auszutauschen und so unterschiedliche Varianten auszuprobieren, die man z. B. zur Archivierung fotografiert. Abb. 3.6 zeigt, wie der Sketch aus Abb. 3.4 mit Büromaterial erstellt werden kann.

Das Beispiel zeigt, wie Elemente nun einfach umpositioniert werden können: So können die obere und untere Leiste ohne viel Aufwand vertauscht werden. Auf den ersten Blick nicht ersichtlich ist aber, dass nun auch direkt mit dem Sketch interagiert werden kann. Ein *Klick* auf einen nachgestellten Button kann mit einer Konsequenz verbunden sein, z. B. dem Entfernen oder Hinzufügen eines UI-Elements. Elemente können auch direkt mit dem Finger verschoben werden, um z. B. Drag'n'Drop mit virtuellen Objekten zu simulieren. Abb. 3.7 zeigt, wie ein nachgestelltes UI-Element durch eine Geste in einem Sketch mit Büromaterial manipuliert werden kann: Das Listenelement kann zur Seite geschoben werden und darunter kommen verschiedene Buttons zum Vorschein. Die Transition vom Sketch zum Prototypen fällt somit sehr viel kürzer aus als bei Scribble Sketches. Demgegenüber steht, dass die Arbeit mit Büromaterial aufwendiger ist und somit mehr Zeit für die Erstellung eingeplant werden muss. Außerdem offenbart das Beispiel noch einen weiteren Nachteil: Die Interaktion ist an physische Grenzen gebunden, jedoch sind in einem (rein virtuellem) UI durchaus Interaktionen möglich, die in der realen Welt nicht machbar sind. Deshalb lassen sich nicht alle Möglichkeiten, die UIs heute bieten, nachstellen. So ist es z. B. nicht ohne Weiteres möglich, in einem Prototypen aus Büromaterial den Druck auf das nachgestellte Display

abzufragen und mit feingranularen Änderungen (z. B. graduelle Anpassung der Hintergrundfarbe an der berührten Stelle) darauf zu reagieren.

Das Beispiel in Abb. 3.7 verdeutlicht aber auch einen weiteren Nachteil: Es wird zwar eine Anwendung dargestellt, die auf einem Smartphone läuft, jedoch ist die verwendete Interaktion (mit dem Zeigefinger einer Hand) für dieses Gerät eher unüblich – Smartphones werden vorwiegend mit dem Daumen der Hand bedient, in der das Gerät gehalten wird. Das setzt aber zwangsläufig voraus, dass das verwendete Gerät – oder hier vielmehr der Prototyp – auch in die Hand genommen werden kann. Ebenso wie Scribble Sketches basieren Sketches mit Büromaterial in erster Linie auf Papier, das aber nicht die notwendige Stabilität aufweist. Damit lässt sich zwar das UI grundlegend darstellen und verwenden, jedoch kann die Haptik nicht oder nur bedingt nachgestellt werden. Diese ist vor allem bei Wearables, bei denen die Haptik einen wesentlichen Teil der UX ausmacht, wichtig. Zwar können Vorlagen verwendet werden, um zumindest den Formfaktor deutlich zu machen (vgl. Abb. 3.5), jedoch können auch diese das Gefühl, das Gerät in der Hand zu halten oder am Körper zu tragen, nicht nachstellen.

Einfacher Bastelbedarf – z. B. Leichtschaumpappe – kann hier Abhilfe schaffen (Greenberg et al., 2011). Abb. 3.8 zeigt, wie eine Smartwatch mit Bastelbedarf nachgebaut werden kann, um die Haptik zumindest grundlegend nachzustellen. Das nachgebaute Gehäuse kann wie eine echte Smartwatch am Handgelenk getragen werden. Das UI wird mit Scribble Sketches dargestellt, die im Beispiel diverse Bilder anzeigen. Mit dem Prototypen kann über Gestensteuerung interagiert werden: Mit einer einfachen Wischgeste kann zum nächsten Bild gewechselt werden. Dieser Wechsel wird durch einen Papierstreifen implementiert, der durch das Gehäuse des Prototypen gezogen wird. Das Konzept lässt sich auf vielerlei neuartiger Geräte übertragen: So erläutert z. B. Chi (2012), wie er mit einer ähnlichen Vorgehensweise erfolgreich und in kurzer Zeit die UX einer AR-Brille testen konnte.

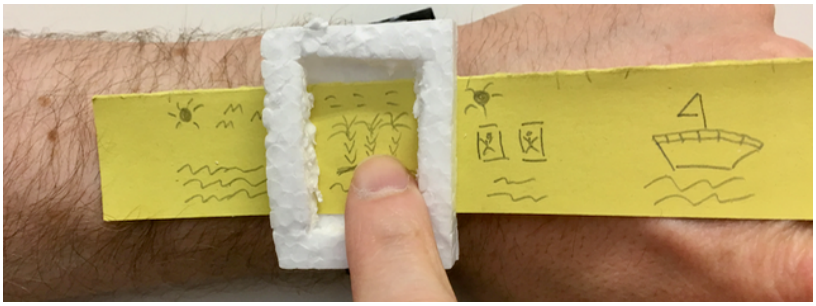


Abbildung 3.8: Prototyp für eine Anwendung auf einer Smartwatch

Büromaterial und Bastelbedarf punkten also insbesondere mit drei Eigenschaften: Sie ermöglichen den partiellen Austausch eines Sketches; sie erlauben eine gewisse Dynamik in der Interaktion, indem Elemente z. B. tatsächlich verschoben werden können; und sie erlauben die Nachbildung von Geräten zur Darstellung der Haptik. Gerade Letzteres kann mit Scribble Sketches nicht erreicht werden. Demgegenüber steht, dass sie aufwendiger in der Erstellung sind und außerdem der Umgang mit dem Material sowie die notwendigen

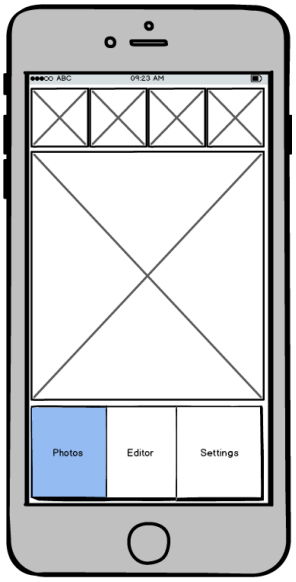


Abbildung 3.9: Ein Wireframe für eine mobile Anwendung (erstellt mit *Balsamiq*)

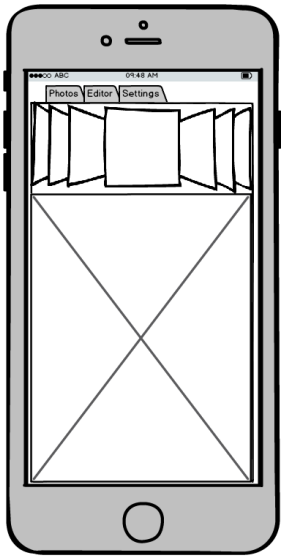


Abbildung 3.10: Ein alternativer Wireframe für eine mobile Anwendung (erstellt mit *Balsamiq*)

Kniffe zur Nachbildung der Interaktion mehr Übung verlangen. Gerade die Interaktion ist auch an physikalische Grenzen gebunden, die bei der tatsächlichen Anwendung nicht existieren. Wigdor & Wixon (2011) bezeichnen diese Art der Interaktion als *superreal*: Interaktion, die in der realen Welt nicht möglich ist, aber in einer (von physikalischen Grenzen befreiten) virtuellen Anwendung problemlos umgesetzt werden kann. Daher können u. U. Möglichkeiten zur Interaktion nicht vollständig dargestellt, sondern genau wie bei Scribble Sketches nur angedeutet werden.

3.2.1.3 Wireframing

Wireframing bezeichnet die Erstellung von Wireframes (engl. für *Drahtgittermodell*) und unterstützt Designer beim Aufbau der grundlegenden Struktur einer Anwendung. Ursprünglich bevorzugt im Bereich des Web-Designs eingesetzt, wird es heute auch gerne zur Entwicklung anderer Anwendungen verwendet und ist oftmals ein wesentlicher Schritt auf dem Weg zum finalen Design. Die grundlegende Idee ist, dass der Aufbau der Anwendung definiert wird, indem bestimmten Bereichen eine Funktion gegeben wird, z. B. wo sich das Menu befindet oder Text dargestellt wird. Der Fokus liegt also stark auf der Positionierung von Inhalten. Typischerweise wird dabei auf bestehende UI-Elemente zurückgegriffen, wodurch die Anwendung aus einer Art Baukasten zusammengesetzt werden kann. Damit entsteht ein Grundgerüst der Anwendung, das über mehrere Iterationen verfeinert werden kann.

Beim Wireframing werden vorwiegend digitale, spezialisierte Tools eingesetzt, allerdings können Wireframes z. B. auch mit Büromaterial erstellt werden – letztlich fokussiert Wireframing die Darstellung eines Grundgerüsts aus vorgefertigten Komponenten. Auch bringen viele Anwendungen, die zur Erstellung von Visualisierungen eingesetzt werden, Komponenten zur Erstellung von Wireframes mit. Darüber hinaus wurden in den letzten Jahren zahlreiche Anwendungen entwickelt, die Designer beim Wireframing unterstützen und unterschiedliche Genauigkeit ermöglichen. Dabei erlauben viele Anwendungen die Integration grundlegender Interaktionen und somit auch die Erstellung von Prototypen, die evaluiert werden können. Abb. 3.9 zeigt ein Beispiel der Anwendung, die bereits in den vorigen Abschnitten eingesetzt wurde, als Wireframe. Anstatt der Bilder wurden – analog zu Abb. 3.3 – Platzhalter verwendet, die in weiteren Iterationen gegen passende Beispiele ausgetauscht werden können. Dennoch wird das grundlegende Prinzip des Wireframings deutlich: Es wird eine Struktur erstellt, die den grundlegenden Aufbau der Anwendung zeigt.

Interessant ist an dieser Stelle, dass das für die Beispiele verwendete Tool auf eine Optik setzt, die einen unfertigen und handgezeichneten Eindruck macht. Dadurch soll unterstrichen werden, dass es sich eben nicht um eine fertige Anwendung, sondern eben nur um ein Konzept handelt, das kommuniziert werden soll. Abb. 3.10 macht dies deutlich und zeigt darüber hinaus einen weiteren Vorteil von Wireframing: Die Entwicklung von Alternativen wird einfacher, da unterschiedliche Ansätze aus den vorgefertigten schnell zusammengestellt werden können. So werden im vorliegenden Beispiel gleich zwei

alternative Konzepte gezeigt: Zum einen wird die bisher unten dargestellte Leiste mit Buttons jetzt nach oben in eine Tab-Bar verlegt, zum anderen wird statt einer Reihe von Vorschaubildern jetzt ein Karussell eingesetzt, um zwischen Bildern zu navigieren.

Wireframing bietet noch weitere Vorteile. So können Wireframes mit digitaler Unterstützung schnell zusammengestellt werden. Die Verwendung standardisierter Elemente fördert dies und hat insbesondere einen gewissen Wiedererkennungswert, da Elemente gleichen Typs stets identisch dargestellt werden. In Kombination mit einem Storyboard (vgl. Kap. 3.2.2) können Wireframes auch eingesetzt werden, um einen Überblick über die Navigationsflüsse innerhalb der Anwendung zu bekommen. Gleichzeitig erlauben einige Tools zur Erstellung von Wireframes auch die Ausführung als *Click-Dummy*, d. h. dass rudimentäre Navigation zwischen UIs möglich wird, analog zum Austausch von Sketches. Somit kann einfache Interaktion schnell dargestellt werden. Ein weiterer interessanter Vorteil, der insbesondere durch die Unterstützung mit digitalen Tools entsteht, ist, dass Wireframes auch auf das Gerät, auf dem die jeweilige Anwendung laufen soll, exportiert werden können. Insbesondere für die Entwicklung von Anwendungen für mobile Geräte oder Wearables hat dies den Vorteil, dass die Wireframes auch auf dem Zielgerät betrachtet und somit die damit einhergehende Haptik dargestellt werden kann.

Den Vorteilen stehen ein paar Nachteile gegenüber. Aus der Wiederverwendbarkeit der standardisierten Elemente ergibt sich zwangsläufig eine gewisse Gleichförmigkeit, sodass der – zumindest in den verwendeten Beispielen – vorgetäuschte Effekt der handgezeichneten Ungenauigkeit verloren geht. Darüber hinaus lassen die vorgegebenen Elemente wenig Spielraum für Kreativität und Designer werden schnell in die Welt klassischer UI-Elemente versetzt: Man greift auf einen vorgefertigten Werkzeugkasten zurück, der lediglich im Rahmen der gegebenen Möglichkeiten erweitert werden kann. Darüber hinaus erlauben Wireframes das schnelle Hinzufügen von Details, die vielleicht das Abstraktionsniveau des aktuellen Projektstands überschreiten.

Die Verwendung digitaler Tools zum Wireframing wirft auch unweigerlich die Frage nach der Integration von Interaktion und somit der Evolution von Wireframes hin zu Prototypen auf. Die Meinungen gehen auseinander: Während manche Hersteller bewusst auf die Integration von Interaktion verzichten und sich auf reines Sketching mit Wireframes konzentrieren (vgl. z. B. Balsamiq Blog (2011)), integrieren andere sowohl die Interaktion mit klassischen UI-Elementen (vgl. z. B. Justinmind Support (oJ[a])) als auch eine Menge vorgefertigter Gesten (vgl. z. B. Justinmind Support (oJ[b])) und erlauben so die Erstellung von Prototypen. Während der Kern der Diskussion für diese Arbeit unerheblich ist, folgt daraus aber zwangsläufig die Frage, wie die Integration individueller Gesten, die nicht durch die vorgegebene Menge Gesten abgedeckt werden, in derartigen Tools umgesetzt werden kann. Eine Möglichkeit wäre die Spezifikation von Gesten mit einer Notation wie GestureCards. Ungeachtet dessen fokussieren Tools zum Wireframing weniger die Interaktion, sondern eher das Layout und den Aufbau der Anwendung.

3.2.2 Storyboards

Die Interaktion mit einer Anwendung hat immer eine zeitliche Komponente, die einen wesentlichen Teil der UX ausmacht (Greenberg et al., 2011). Die grundlegende Idee findet sich schon in einem der wesentlichen Artefakte der Anforderungserfassung, die dem Design der Anwendung vorausgeht, wieder: Anwendungsfälle werden oft als *User Stories* erfasst, also als *Geschichten*, die die Ziele des Benutzers beschreiben. Geschichten beschreiben aber eine Entwicklung über einen gewissen Zeitraum. Anwendungsfälle bestehen somit typischerweise nicht aus einzelnen UIs, die wie in Kap. 3.2.1 dargestellt auch mit einfachen Mitteln erstellt werden können, sondern aus einer ganzen Kette mehrerer UIs, in denen unterschiedliche Dinge getan werden, die letztlich in ihrer Gesamtheit zum Ziel führen. Die daraus folgende Dynamik kann – wenn auch eingeschränkt – in einigen der bisher vorgestellten Beispiele eingefügt werden, z. B. in der Nachbildung einer Smartwatch mit Bastelbedarf (vgl. Abb. 3.8). Einzelne Sketches sind aber im Wesentlichen statisch, denn sie zeigen das UI zu einem bestimmten Zeitpunkt im gesamten Lebenszyklus der Anwendung; hintereinander ergeben sie jedoch – wie im Beispiel der Smartwatch – eine Sequenz mehrerer UIs, die eben der User Story[7] entspricht. Es gibt also auch eine Zeit zwischen zwei Sketches, die für die Anwendung von Belang ist. Tatsächlich zeigen die Beispiele aus Abb. 3.7 und Abb. 3.8, dass insbesondere für Gestensteuerung auch die Zeit der eigentlichen Ausführung relevant ist. Kap. 3.3 betrachtet, was die Integration der Interaktion mit einer Anwendung für Sketching und Prototyping bedeutet.

[7] In diesem Beispiel könnte die zugehörige User Story lauten: „Als Benutzer möchte ich meine Bilder der Reihe nach betrachten können."

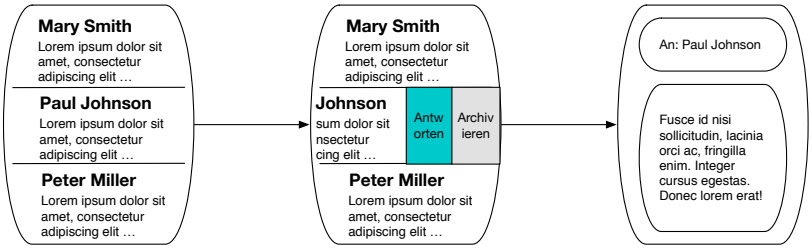


Abbildung 3.11: Ein Storyboard für eine Anwendung auf einer Smartwatch

Der zeitliche Ablauf zwischen mehreren Sketches einer Anwendung wird typischerweise mit Storyboards dargestellt. Greenberg et al. (2011) stellen dabei unterschiedliche Möglichkeiten vor. Für einfache Storyboards kann eine Sequenz von Sketches ausreichen, die einen Anwendungsfall repräsentieren. Abb. 3.11 zeigt als Beispiel das Storyboard für eine Anwendung auf einer Smartwatch. Die wesentlichen Elemente eines Storyboards werden hier sichtbar: einzelne UIs und Pfeile, die den zeitlichen Fluss darstellen. Der Nutzer navigiert in diesem Szenario von einem der dargestellten UIs zum nächsten. Das Beispiel verdeutlicht aber auch das wesentliche Problem eines Storyboards: Es ist zwar klar, dass von einem UI zum nächsten navigiert wird, aber nicht wie. Der Aspekt der Interaktion wird nur abstrakt dargestellt und die Details komplett weggelassen – die Leerstellen zu füllen bleibt der Vorstellungskraft des Betrachters überlassen und kann somit zu unterschiedlichen Interpretationen führen (Greenberg et al., 2011).

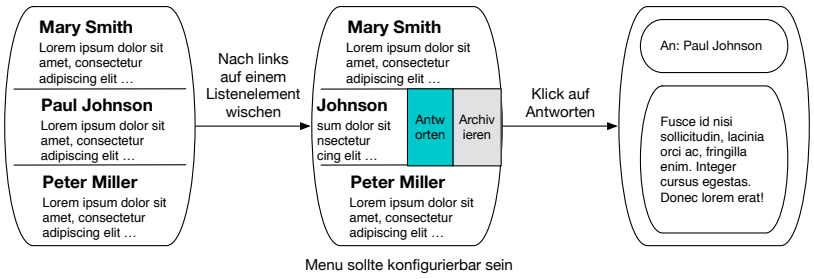


Abbildung 3.12: Ein Storyboard mit Annotationen für eine Anwendung auf einer Smartwatch

Greenberg et al. (2011) beschreiben, dass ein Storyboard als Zustandsdiagramm betrachtet werden kann, bei dem die Sketches des UIs die Zustände sind und die Übergänge, die durch Interaktion mit der Anwendung ausgelöst werden, die Transitionen. Dabei demonstrieren die Autoren, wie unterschiedliche Detailtiefe durch die Einbindung von Sketches, Annotationen und Notizen – die analog zur ihrer Verwendung in Sketches eingesetzt werden – erreicht werden kann. Abb. 3.12 erweitert das Storyboard und verwendet zusätzlich Annotationen und Notizen. Die Transitionen werden dazu mit der zugrundeliegenden Interaktion annotiert und die Gesten, die hier verwendet werden sollen, namentlich genannt.

Sequentielle Storyboards sind verhältnismäßig einfach und finden sich mutmaßlich eher am Anfang des Designs (Greenberg et al., 2011). Typischerweise können Benutzer auf einem UI aber unterschiedliche Entscheidungen treffen, wodurch sich mehrere Navigationspfade ergeben. Im Laufe der Entwicklung werden also im Storyboard Punkte identifiziert, an denen der Nutzer abbiegt und einem anderen Pfad folgen kann. Dafür werden verzweigende (engl. *branching*) Storyboards eingesetzt (Greenberg et al., 2011). Von jedem im Storyboard dargestelltem UI können mehrere Transitionen ausgehen, die die unterschiedlichen Möglichkeiten repräsentieren. Abb. 3.13 zeigt ein Beispiel.

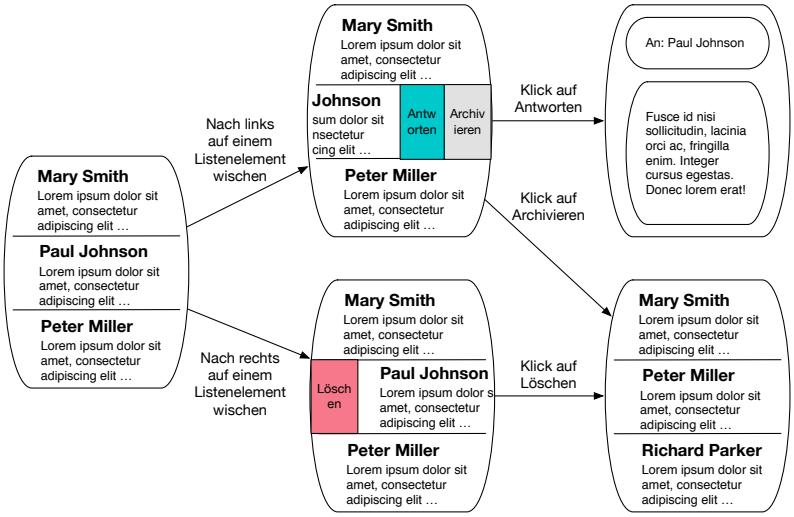


Abbildung 3.13: Ein verzweigendes Storyboard

Bei der Erstellung eines Storyboards sind die gleichen Regeln zu beachten wie bei Sketches. So muss nicht jede einzelne Änderung am UI im Storyboard zu finden sein, jedoch sollten die wesentlichen Zustände – Greenberg et al.

(2011) sprechen hier von *Schlüsselszenen* (engl. *Key Frames*) – dargestellt werden, um den Navigationsfluss der Anwendung deutlichen zu machen. Die zentrale Frage ist für die Autoren, ob die Zielgruppe, der das Storyboard gezeigt wird, die fehlenden Teile mental hinzufügen kann. Natürlich können – wie in Abb. 3.12 gezeigt – Annotationen und Notizen ebenso verwendet werden, um die dargestellten Informationen zu erweitern und anzureichern. Die verschiedenen Arten von Storyboards unterscheiden sich wie gezeigt in der Feingranularität. Somit werden Storyboards analog zu Sketches verwendet und je nach Zweck mit unterschiedlichen Informationen ausgestattet.

Storyboards zeigen einen Überblick über die Anwendung und verknüpfen mehrere Sketches zu einer gesamten Geschichte. Sie zeigen die Entwicklung einer Anwendung in Abhängigkeit von der Interaktion des Benutzers und welche Navigationspfade möglich sind. Außerdem können Storyboards auch Grundlage für animierte Präsentationen sein, durch die schon früh gezeigt werden kann, wie sich die Anwendung verhält (Greenberg et al., 2011).

Buxton (2007, S. 293) kritisiert jedoch, dass Storyboards, wie sie typischerweise verwendet werden, einen starken Fokus auf die einzelnen Zustände des UIs legen, die in den verwendeten Sketches dargestellt werden. Die Details der Interaktion mit einer Anwendung verbergen sich jedoch in den Transitionen von einem Zustand zum nächsten. Zusammengefasst kommt Buxton (2007) zu dem Schluss, dass Storyboards zeigen, was man von wo erreichen kann, aber die für die Interaktion wesentliche Information – *wie* man den nächsten Zustand erreicht – fehlt. Zwar zeigen Abb. 3.12 und Abb. 3.13, wie die Interaktion als Annotation hinzugefügt werden kann, daraus ergeben sich aber unterschiedliche Probleme:

- Die Bedeutung von *wischen* muss allen im Team klar sein. Gerade die unterschiedlichen Möglichkeiten, die Gestensteuerung bietet, lassen sich mit textuellen Annotationen nur schwer exakt wiedergeben.
- Die Reaktion der Anwendung auf Details der Interaktion bleibt unklar, z. B. wenn auf Details der Ausführung reagiert werden soll.
- Typischerweise können unterschiedliche Pfade beschritten werden, z. B. durch unterschiedliche Gesten, oder mit mehreren UI-Elementen interagiert werden. Daraus ergibt sich zwangsläufig ein (z. T. undurchsichtiges) Darstellungsproblem.

Insbesondere der letzte Punkt ist nicht auf moderne UIs beschränkt, die auf Gestensteuerung basieren, sondern bereits in klassischen WIMP-UIs zu finden. Es ist schon hier schwer darzustellen, auf welches Element wie reagiert werden soll, da eine direkte Zuordnung notwendig ist. Landay & Myers (1996) z. B. verknüpfen Transitionen direkt mit UI-Elementen, zeichnen also den Startpunkt direkt in den dargestellten Sketch und versuchen so, den initialen Punkt der Interaktion zu verdeutlichen. Dadurch wird zum einen der Sketch unübersichtlich und mit weiteren Informationen überlagert, zum anderen ist diese Variante für Gestensteuerung nur schwer umzusetzen. Eine Notation zur Definition von Gesten wie GestureCards kann hier Abhilfe schaffen. Darüber hinaus fehlt in den dargestellten Beispielen die Umgebung, in der sich der

Benutzer befindet, die je nach Anwendung ebenfalls dargestellt werden muss, wenn sie denn Einfluss auf die jeweilige User Story hat. Greenberg et al. (2011) stellen als praktisches Werkzeug hierfür das narrative Storyboard vor, das lediglich eine Variante des sequenziellen Storyboards ist und den Benutzer sowie die Anwendung in der passenden Umgebung zeigt.

3.3 Interaktion und Gestensteuerung in Sketches und Prototypen

Die vorigen Abschnitte zeigen einen Überblick über gängige Techniken zum Sketching und Prototyping, die zum Entwurf und zur Validierung unterschiedlicher Lösungsansätze verwendet werden können. Sie verdeutlichen die in Kap. 2 postulierte These, dass Interaktion vorwiegend implizit in Sketches und Prototypen definiert wird und daher explizit spezifiziert werden muss. Dies zeigt sich an unterschiedlichen Stellen:

- In Scribble Sketches können bekannte UI-Elemente aufgezeichnet werden. Die Interaktion mit ihnen muss meist nicht expliziert definiert werden, da sie z. T. aus weit verbreiteten Systemen bekannt sind und sie recht wenig Variation erlauben. Die Interaktion ist auf einfache Aktionen beschränkt und wenig dynamisch. Dynamische Interaktion wie Gestensteuerung kann mit Annotationen – insbesondere Pfeilen – angedeutet, aber nicht deutlich beschrieben werden. Saffer (2008, S. 103) geht noch einen Schritt weiter und beschreibt explizit, dass textuelle Annotationen ein probates Mittel sind, um einfache Gesten zu beschreiben. Allerdings beschränkt er sich auf Gesten, die allgemein bekannt sind und geht nicht auf individuelle Gesten ein.

- Durch den Einsatz von Büromaterial und Bastelbedarf kann Interaktion besser dargestellt werden. Insbesondere kann mehr Dynamik integriert werden, wodurch eben auch Gestensteuerung nachgestellt werden kann. Dadurch wird aber besonders die implizite Definition von Interaktion unterstützt – sie ist im Prototypen enthalten und kann verwendet werden, ist aber nicht exakt spezifiziert, sondern einfach nur da. Gleichzeitig sind der Interaktion nun auch physikalische Grenzen gesetzt, die bei der finalen (virtuellen) Anwendung nicht existieren.

- Wireframing steht vor einem ähnlichen Dilemma: Zwar kann dank technischer Unterstützung zügig eine grundlegende Struktur der Anwendung entwickelt werden, die auch – je nach eingesetztem Wireframing-Tool – ausführbar und somit validierbar wird, die Integration von Interaktion beschränkt sich aber oftmals auf die klassischen UI-Elemente. Zwar unterstützen einige Tools eine Menge vordefinierter Gesten, die Definition individueller Gestensteuerung, wie sie u. U. notwendig sein kann, ist aber nicht ohne Weiteres möglich.

- Storyboards zeigen zwar, wie sich Nutzer in einer Anwendung bewegen, legen ihren Fokus aber sehr stark auf die wesentlichen Zustände, also die Schlüsselszenen. Interaktion findet aber zwischen Zuständen statt,

sodass hier Informationen fehlen. Buxton (2007) geht sogar so weit, dass die Transitionen für die UX wichtiger sind als die Zustände.

Hier liegt auch ein wesentlicher Unterschied zwischen UI-Design und klassischem industriellen Produktdesign: Zeit und Verhalten der Anwendung sowie des Nutzers spielen eine wesentliche Rolle und müssen explizit entworfen werden (Buxton, 2007). Gerade beim Einsatz von Gestensteuerung ist dies auch notwendig, da Gesten offensichtlich eine zeitliche Komponente enthalten, die aber mit gängigen Werkzeugen nur größtenteils implizit definiert werden kann. Die Interaktion ist da, aber sie wird nicht detailliert beschrieben, sodass sie aus Sketches und Prototypen abgeleitet werden muss. Insbesondere wenn individuelle Gesten eingesetzt werden sollen, geraten gängige Werkzeuge schnell an ihre Grenze. Interessant ist dabei auch, dass Anwendungen, die Sketching und Prototyping elektronisch unterstützen, häufig Gestensteuerung zum Zeichnen einsetzen und anschließend mithilfe von Formenerkennung automatisch UI-Elemente ableiten. Oftmals bieten diese Anwendungen jedoch keine eingebaute Unterstützung zur Definition von Gestensteuerung. Ausnahmen sind z. B. de Souza Alcantara et al. (2013) und Hesenius et al. (2016), die die explizite Definition von Gesten in Sketches erlauben und damit ausführbare Prototypen mit frei definierbaren Gesten ermöglichen.

Sowohl Buxton (2007) als auch Greenberg et al. (2011) schlagen als Alternative den Einsatz von Animationen vor, um dynamische Aspekte der Interaktion darzustellen. Dabei gehen beide Autoren insbesondere auf Gestensteuerung als Beispiel ein, schlagen aber unterschiedliche Techniken vor. Buxton (2007) schlägt eine Art Daumenkino vor, bei dem mehrere einzelne Sketches den Verlauf der Animation darstellen und schnell hintereinander gezeigt werden. Dadurch stellt sich der vom Daumenkino bekannte Effekt einer – scheinbar – flüssigen Bewegung ein. Greenberg et al. (2011) gehen noch einen Schritt weiter und setzen auf elektronische Unterstützung mithilfe passender Software, die aber – in den Beispielen der Autoren – keine feingranulare Definition der Interaktion ermöglicht. In Kap. 10, Kap. 11 und Kap. 12 wird diese Idee aufgegriffen und gezeigt, welche Optionen der Einsatz von Animationen in einer Notation zur Beschreibung von Gesten zur genauen Darstellung bietet.

Die in Sketches und Prototypen festgelegte Interaktion muss sowohl von UI-Designern als auch Entwicklern verstanden und zwischen den Beteiligten kommuniziert werden, denn sie ist die Grundlage für die Auswahl notwendiger technischer Komponenten, die im Folgenden diskutiert werden. Daher muss die Interaktion explizit definiert und genauestens beschrieben werden können. Eine Notation wie GestureCards, die in Kap. 7 vorgestellt wird, ist der Schlüssel für die Spezifikation von Gesten.

4 Implementierung gestenbasierter Anwendungen

Klassische WIMP-UIs werden im Wesentlichen aus vorgefertigten Software-Komponenten zusammengestellt, die oftmals direkt vom Betriebssystem zur Verfügung gestellt werden. Bei Gestensteuerung hingegen spielt die Zusammensetzung und Abstimmung von Hard- und Software-Komponenten eine wichtige Rolle. Zur Implementierung einer gestenbasierten Anwendung müssen eine Reihe von Entscheidungen getroffen werden, um die Komponenten auszuwählen, die zur Erzeugung, Übertragung und Verarbeitung der anfallenden Daten notwendig sind. Dabei müssen Entwickler Antworten auf verschiedene Fragen finden, die auf Informationen aufbauen, die u. U. von anderen Teammitgliedern in vorigen Phasen der Entwicklung erarbeitet wurden:

- Welche Hardware-Komponenten müssen zur Darstellung der Anwendung eingesetzt werden?
- Welche Informationen über die Interaktion müssen geliefert werden?
- Welche Sensoren müssen zur Erfassung der Interaktion eingesetzt werden?
- Welche Muster in den Informationen müssen als gültige Interaktion erkannt werden?
- Welche Software-Komponenten müssen zur Verarbeitung der Interaktion eingesetzt werden?

Die Antworten auf diese Fragen sind direkt mit der Spezifikation von Gestensteuerung verknüpft. Die Spezifikation dient an dieser Stelle nicht nur als Grundlage für Entscheidungen und Auswahl, sondern auch zur Nachvollziehbarkeit. Bestimmte Gesten setzen bestimmte technische Komponenten voraus und ggf. muss im weiteren Verlauf der Entwicklung gerechtfertigt werden werden, warum eine bestimmte Komponente ausgewählt wurde oder eine bestimmte Geste nicht implementiert werden konnte. Eine Notation wie GestureCards kann diese Grundlage liefern und die notwendigen Informationen über die verwendeten Gesten zur Verfügung stellen. Folglich ist eine Betrachtung der unterschiedlichen Entscheidungen, die bei der Entwicklung einer gestenbasierten Anwendung getroffen werden müssen, sinnvoll.

Im Rahmen des Designs entwickelte Prototypen und Designkonzepte (vgl. Kap. 3) beantworten die genannten Fragen zu einem gewissen Grad oder liefern zumindest einige Informationen, die für die entsprechenden Entscheidungen notwendig sind. So wird zwar vielleicht eine Geräteklasse (z. B. Smartphone oder Smartwatch) festgelegt, aber nicht die konkrete Ausprägung. Die Auswahl eines passenden Gerätes obliegt dann den Entwicklern, die die o. g. Fragestellungen im Blick haben müssen. Oder der umgekehrte Fall

tritt ein: Das konkrete Gerät ist bereits vorgegeben und das entwickelte Designkonzept muss mit den Fähigkeiten des Gerätes abgeglichen werden, um festzustellen, ob die entworfene Anwendung überhaupt mit den gegebenen Mitteln umgesetzt werden kann. Zwangsläufig folgt daraus die Entscheidung, ob das Design überarbeitet oder ein anderes Gerät ausgewählt wird. In beiden Szenarien müssen jedoch Entscheidungen getroffen werden, die den weiteren Verlauf der Entwicklung nachhaltig beeinflussen können.

Eine weitere Entscheidung betrifft die verwendeten Sensoren und – damit verbunden – die Daten über die Interaktion, die diese bereitstellen müssen. Einerseits kann auf im Gerät bereits vorhandene Sensoren zurückgegriffen werden: Mobile Geräte sind z. B. oft mit einem Accelerometer ausgestattet, das zur Erkennung von räumlichen Gesten mit dem Gerät eingesetzt werden kann, und haben in der Regel einen berührungsempfindlichen Bildschirm, der für Oberflächengesten verwendet wird. Andererseits kann es notwendig sein, zusätzliche Sensoren auszuwählen und in das System zu integrieren. Ein separater Tiefensensor kann z. B. eingesetzt werden, um die Position der Hände zu bestimmen, und ermöglicht so die Integration räumlicher Gesten in Anwendungen, die auf Geräten ohne entsprechende Möglichkeiten laufen. Die Sensoren stellen letztlich die notwendigen Informationen zur Verfügung, um Gesten zu erkennen und mit dem Gerät interagieren zu können.

Auf die Auswahl passender Geräte und Sensoren folgen aber oft weitere Entscheidungen, Einschränkungen und ggf. Folgekosten, die nicht zwingend monetärer Natur sind, sondern eher mit erhöhtem Aufwand im Betrieb und fortschreitendem Lebenszyklus der Anwendung einhergehen – also bei Änderung, Anpassung und Erweiterung um neue Funktionalitäten. Verfügbare Hardware ist oft an eine Plattform bestimmter Hersteller gebunden und Entwickler sind gezwungen, auf die zur Verfügung gestellten Bibliotheken und Möglichkeiten zurückzugreifen. Daraus folgen in der Regel auch Einschränkungen bei der Auswahl der Komponenten und es wird gleichzeitig eine technische Umgebungen vorgegeben, auf der u. U. notwendige weitere Komponenten, z. B. die optimale Bibliothek zur Erkennung ausgeführter Gesten, nicht verfügbar sind. Über mehrere Entscheidungen kaskadierende Konsequenzen sind also möglich und zu vermeiden, weshalb eine genaue Analyse der Interaktion notwendig ist. Gleichzeitig bringen die verschiedenen Plattformen oft schon einige Möglichkeiten mit, Gestensteuerung zu integrieren, sodass eine gezielte Betrachtung der vorhandenen Werkzeuge sinnvoll ist.

Zu guter Letzt setzt Gestensteuerung noch die Integration einer wesentlichen Software-Komponente voraus: des Gestenerkenners. Der Gestenerkenner interpretiert ausgeführte Gesten aus der Menge der Eingabedaten und ist somit eine Kernkomponente einer gestenbasierten Anwendung, die sorgsam ausgewählt werden muss. In den letzten Jahren wurden zahlreiche Verfahren entwickelt, die nicht nur unterschiedliche Vor- und Nachteile haben, sondern auch auf bestimmte Anwendungsfälle zugeschnitten sind. Die für die Auswahl notwendigen Faktoren hängen im Wesentlichen von den zu implementierenden Gesten und den von den Sensoren zur Verfügung gestellten Daten ab.

Im Folgenden wird auf die unterschiedlichen Fragestellungen und Entscheidungen eingegangen, die die Entwicklung gestenbasierter Anwendungen für Software-Ingenieure mit sich bringt. Zunächst sollen die unterschiedlichen Arten von Gesten betrachtet werden (Kap. 4.1). Anschließend wird erläutert, welche Möglichkeiten gängige Frameworks für die Integration von Gestensteuerung bereits mitbringen (Kap. 4.2), die sich auf wenige vorgefertigte Gesten reduzieren lassen. Auf die unterschiedlichen technischen Entscheidungen, die Entwickler bei der Implementierung einer gestenbasierten Anwendung treffen müssen, wird daraufhin näher eingegangen (Kap. 4.3). Im Fokus steht dabei die Auswahl der Hardware, der Sensoren und der notwendigen Algorithmen.

4.1 Arten von Gesten

Grundsätzlich sind zwei Arten von Gesten zu unterscheiden: Gesten auf zweidimensionalen Oberflächen und Gesten im dreidimensionalen Raum. Erstere werden auch als *Oberflächengesten* oder *Multi-Touch-Gesten* bezeichnet, obwohl diese Bezeichnung eigentlich irreführend ist: Multi-Touch-Gesten stellen nur eine Teilmenge der Oberflächengesten dar (vgl. Kap. 5.2.1). Gesten im dreidimensionalen Raum hingegen werden auch als *räumliche Gesten* oder *In-Air-Gesten* bezeichnet. Beide Arten haben eine Fülle an Gemeinsamkeiten, aber auch einige wesentliche Unterschiede, die für die Implementierung gestenbasierter Anwendungen relevant sind. Insbesondere können räumliche Gesten nicht einfach als Erweiterung von Oberflächengesten um eine weitere Dimension betrachtet werden.

Die Bezeichnung *Oberflächengesten* impliziert bereits, dass eine Oberfläche vorhanden sein muss, die berührt und auf der die Geste ausgeführt wird. Allgemein verbreitet sind berührungsempfindliche Bildschirme, aber auch andere Oberflächen sind denkbar: Harrison & Hudson (2008) zeigen, wie unterschiedliche Oberflächen (z. B. Wände) für Gestensteuerung eingesetzt werden können. Auch schon eine Annäherung an die Oberfläche kann – passende Sensoren vorausgesetzt – für die Interaktion mit einer Anwendung interessant sein (Hinckley et al., 2016); darüber hinaus können auch Gesten, die knapp über der Oberfläche ausgeführt werden, erkannt werden (vgl. Kap. 5.1.6).

Räumliche Gesten hingegen sind nicht von einer Oberfläche abhängig. Dieser – scheinbar triviale – Unterschied geht aber mit Herausforderungen einher, die sowohl technischer als auch konzeptioneller Natur sind. Oberflächengesten sind durch den notwendigen Bezug zu einer Oberfläche vor allem einfacher zu segmentieren als räumliche Gesten, d. h. bei einer Oberflächengeste sind die Teile, die zur Ausführung der Geste gehören, eindeutig durch Anfang und Ende der Oberflächenberührung gekennzeichnet. Bei einer räumlichen Geste hingegen besteht keine derartig eindeutige Trennung, sodass aktiv zwischen gewollter Eingabe und *Rauschen* durch Bewegung des Nutzers unterschieden werden muss. Notwendig ist eine *Kupplung*, mit der zwischen tatsächlicher Interaktion und Rauschen unterschieden werden kann (Wigdor & Wixon, 2011, Kap. 15). Eine einfache Möglichkeit sind z. B. reservierte Aktionen wie das Öffnen und Schließen der Hände, um Start und Ende einer Geste

zu markieren. Die Kupplung muss aber auf die spezifischen Anforderungen der Anwendung und des Anwendungskontexts angepasst sein und erfordert somit zwangsläufig intensive Tests des Interaktionskonzepts. Jerald (2016) bezeichnet das gleiche Konzept als *Push-to-Gesture*, also die Aktivierung eines bestimmten Modus, in dem Gesten erkannt werden.

Ein weiterer Unterschied zwischen Oberflächengesten und räumlichen Gesten ist, dass Oberflächengesten in der Regel auf einer vom Sensor zur Verfügung gestellten Oberfläche,[1] z. B. auf berührungsempfindlichen Bildschirmen, durchgeführt werden. Das heißt, die Eingaben werden in einer für den Sensor gültigen Fläche durchgeführt; die Informationen sind demnach ortsgebunden innerhalb des vom Sensor vorgegebenen Koordinatensystems. Das gleiche Szenario existiert auch für räumliche Gesten: Ein Sensor wird an einer passenden Stelle platziert und überwacht einen Raum, in dem Benutzer Gesten ausführen. Die Gesten sind dann ebenfalls an ein vom Sensor aufgespanntes Koordinatensystem gebunden. Darüber hinaus kann aber der Benutzer den Sensor für räumliche Gesten auch in der Hand halten. Dann ist eine Überwachung des Raumes durch den Sensor und somit eine Ortsbestimmung nicht nötig, denn die Informationen über die Gesten können auch anderweitig, z. B. über Beschleunigungsdaten des Sensors, erfasst werden. Daraus folgen aber andere Eingabedaten, die letztlich eine Grundlage der Entscheidungen für unterschiedliche Komponenten sind, die Entwickler treffen müssen.

[1] Ausnahmen wie Harrison & Hudson (2008), die die Ausführung von Gesten auf gewöhnlichen Oberflächen wie Wänden oder Tischen ermöglichen, bestätigen lediglich die Regel.

Es liegt nahe, dass die Art der zu integrierenden Gesten eine wichtige Grundlage für alle weiteren Entscheidungen ist. Im Folgenden soll daher ein weiterführender Blick mithilfe der vorhandenen wissenschaftlichen Literatur geworfen werden und zunächst einige Taxonomien vorgestellt werden, nach denen verschiedene Autoren Gesten klassifizieren. Anschließend werden Gesten und Posen unterschieden, die beide als Teil von Gestensteuerung wahrgenommen werden. Beide Begriffe werden in der Literatur unterschiedlich und teils synonym verwendet.

4.1.1 Taxonomien von Gesten

Die (nahezu grenzenlose) Vielfalt, die Bewegungen der menschlichen Hände und somit Gestensteuerung für HCI bieten, werfen zwangsläufig die Frage auf, ob und wie sich diese kategorisieren lassen und ob eine derartige Taxonomie hilfreiche Hinweise für die Spezifikation von Gesten enthält. Verschiedene Autoren haben daher versucht, Gesten zu klassifizieren und zu kategorisieren. Im Folgenden soll ein Fokus auf Arbeiten gelegt werden, die einen direkten Bezug zu HCI haben. Eine Betrachtung weiterführender Arbeiten, insbesondere solcher, die Gesten in zwischenmenschlicher Kommunikation fokussieren, ist im Kontext von HCI bzw. der Entwicklung von Software nicht zielführend, da sich die grundlegende Idee von Gesten in beiden Fällen unterscheidet (Wobbrock et al., 2009).

Wobbrock et al. (2009) unterscheiden vier grundlegende Dimensionen, denen Gesten zugeordnet werden können:

- Form (*Form*)

- Natur (*Nature*)
- Bindung (*Binding*)
- Fluss (*Flow*)

Die Autoren unterteilen diese noch weiter in Kategorien, die genauer auf die unterschiedlichen Eigenheiten der jeweiligen Dimension eingehen.

Die *Form* einer Geste beschreibt bei Wobbrock et al. (2009) weniger die geometrische Form, sondern eher die Form der Ausführung. Die Autoren unterteilen weiter in statische und dynamische Posen (*static* und *dynamic Poses*), die beschreiben, ob die Haltung der Hand bestehen bleibt oder sich mit der Zeit ändert, ohne ihre Position zu verlassen. Außerdem fassen sie sowohl Bewegungen als auch einfache Berührungen als Teil dieser Dimension auf, nach denen sich Gesten nochmal genauer unterscheiden lassen. Die Form wird lediglich einer Hand zugeteilt; eine Geste mit beiden Händen hat also zwangsläufig zwei Formen, die auch gemischt werden können.

Die Dimension *Natur* beschreibt im Sinne von Wobbrock et al. (2009), ob die Geste eine Verknüpfung zu bestehenden Gegebenheiten hat. So werden symbolische Gesten, die eine anderweitig bekannte Bedeutung übernehmen (z. B. das Zeigen des ausgestreckten Daumens zur Bestätigung), von abstrakten Gesten unterschieden, deren Bedeutung willkürlich gewählt ist (z. B. Doppeltap mit zwei Fingern zur Detailauswahl). Außerdem werden physische Gesten, die ein Äquivalent in der echten Welt haben (z. B. Drag'n'Drop), von metaphorischen Gesten abgegrenzt, deren eigentliche Bedeutung vom mentalen Modell der Nutzer abhängt (z. B. eine sich wiederholende kreisförmige Bewegung, um das Zurückspulen in der Zeit darzustellen).

Gesten können bei Wobbrock et al. (2009) darüber hinaus eine *Bindung* an virtuelle Objekte haben und dadurch eine andere Bedeutung erhalten. Das Auseinanderziehen beider Finger kann z. B. ein einzelnes Objekt vergrößern, wenn die Geste gebunden ist. Ebenso könnte der Start- oder Endpunkt einer Geste in der virtuellen Welt eine Bedeutung haben: Ein virtuelles Objekt könnte gelöscht werden, wenn es über eine bestimmte Seite des Bildschirms herausgeschoben wird. Demgegenüber steht, dass Gesten auch losgelöst sein können, nur für sich selbst stehen und an jeder beliebigen Stelle innerhalb der Anwendung ausgeführt werden können. Das Auseinanderziehen beider Finger kann so auch verwendet werden, um in die Szenerie hereinzuzoomen. Gesten können also entweder gebunden oder unabhängig sein.

Außerdem können Gesten einem gewählten *Fluss* unterliegen. Die Anwendung kann während der Ausführung oder danach reagieren (vgl. Kap. 5.1.11). Für die Umsetzung der Anwendung ist diese Unterscheidung relevant, da eine Reaktion während der Ausführung einer Geste sofort eine gewisse Unsicherheit über das weitere Vorgehen des Nutzers in die Anwendung einbaut. Folglich muss das Interaktionskonzept so definiert werden, dass Verwechslungen von Gesten während der Ausführung ausgeschlossen sind. Darüber hinaus muss eine Geste, auf die zur Ausführungszeit reagiert werden soll, nicht als atomares Kommando verstanden werden, sondern als kontinuierlich eintreffender Strom von Befehlen. Eintreffende Informationen müssen also sofort

interpretiert und die Ergebnisse der Interpretation ausgeführt werden. Es steht zu vermuten, dass hierfür auch eine angepasste UI-Architektur notwendig ist.

Zhai et al. (2012, S. 107ff) beschreiben fünf Dimensionen, anhand derer Gesten klassifiziert werden können:

- Analoge und abstrakte Gesten (*Analogue vs. Abstract Gestures*)
- Kommandos und Symbole (*Command vs. Symbol Gestures*)
- Ordnung der Komplexität (*Order of Complexity*)
- Visuelle und räumliche Abhängigkeit (*Visual-spatial Dependency*)
- Werkzeug und Sensortyp (*Implement and Sensor Type*)

Analoge Gesten bezeichnen im Sinne von Zhai et al. (2012) Gesten, die Effekte aus der realen Welt nachahmen oder eine Bedeutung aufgrund kultureller Konventionen haben. Damit einher geht demnach häufig auch eine Reaktion der Anwendung auf die Geste während diese ausgeführt wird (vgl. Kap. 5.1.11). Demgegenüber stehen abstrakte Gesten, die die Autoren auch *symbolische* Gesten nennen, deren Ausführung keine direkte Entsprechung in der realen Welt hat (z. B. die lange Berührung eines virtuellen Objekts zur Aktivierung). Bei Wobbrock et al. (2009) findet sich die Unterscheidung in abstrakte und analoge Gesten in der Dimension *Natur*.

Zhai et al. (2012) unterscheiden außerdem zwischen Gesten, die verschiedene Kommandos an ein System definieren, und Symbolen, die zur Texteingabe verwendet werden. Ein Kommando wird dabei direkt mit einer Geste verknüpft und die entsprechende Funktionalität aufgerufen. Mit Texteingabe hingegen ist nicht nur die Eingabe von Buchstaben mithilfe eines Stifts auf einer berührungsempfindlichen Oberfläche gemeint, sondern auch alternative Texteingabesysteme, die auf den gleichen technischen Grundlagen wie Oberflächengesten basieren (vgl. Kap. 4.1.2 für weitere Details). Ein Beispiel für gestenbasierte Texteingabe findet sich u. a. bei Goldberg & Richardson (1993). Allerdings können Zhai et al. (2012) zufolge Gesten für Kommandos und Texteingabe eng verwandt sein. Im gegebenen Beispiel von Goldberg & Richardson (1993) können die Gesten als Kommandos für einzelne Buchstaben betrachtet werden. Bei anderen Texteingabesystemen hingegen – z. B. Sim (2012) – basiert die Eingabe lediglich auf den gleichen Daten wie Gestensteuerung, stellt aber kein Kommando in diesem Sinne dar.

Außerdem unterscheiden Zhai et al. (2012) drei Ordnungen von Gesten. Sie fokussieren zwar Oberflächengesten, die grundlegende Idee kann aber auch auf räumliche Gesten angewandt werden. Unter Gesten der nullten Ordnung (*zero order*) verstehen die Autoren einfache Berührungen der Oberfläche ohne weitere Bewegung. Das Äquivalent für räumliche Gesten wäre, wenn der Benutzer auf eine bestimmte Stelle zeigt. Gesten, die in einem Zug ausgeführt werden, bezeichnen sie als Gesten der ersten Ordnung (*first order*) und Gesten, die in mehreren Zügen ausgeführt werden, als Gesten höherer Ordnung (*higher order*). Beides ist auch bei räumlichen Gesten möglich.

Die visuelle und räumliche Abhängigkeit beschreibt im Sinne von Zhai et al. (2012), ob die Bedeutung einer Geste von dem Ort, an dem sie ausgeführt wird, oder ihrer Größe abhängt. Ein einfaches Beispiel wäre, ob eine Geste abhängig von einem virtuellen Objekt ist oder an einer beliebigen Stelle auf dem Bildschirm ausgeführt werden kann. Bei Wobbrock et al. (2009) ist dies unter der Dimension *Bindung* zu finden. Außerdem kann die Größe, in der die Geste ausgeführt wird, relevant für die Anwendung sein – z. B. kann die Entfernung zwischen den beiden Fingern einer Spread- oder Pinch-Geste bestimmen, wie weit ein Objekt vergrößert oder verkleinert wird.

Abschließend gehen Zhai et al. (2012) noch auf die technische Grundlage ein, die Wobbrock et al. (2009) nicht betrachten. So unterscheiden sie Gesten, die mit unterschiedlichen Werkzeugen (z. B. Stift oder Finger) ausgeführt werden, und verschiedene Sensortypen (vgl. Kap. 4.3.1). Gerade die denkbaren Werkzeuge gehen oftmals mit dem gewählten Sensortyp einher, da nicht alle Sensoren in der Lage sind, die gleiche Geste mit unterschiedlichen Werkzeugen zu erkennen. Bei räumlichen Gesten ist darüber hinaus noch zu berücksichtigen, ob der Sensor an einer bestimmten Stelle steht und einen Raum überwacht oder vom Benutzer in der Hand gehalten wird.

Die betrachteten Taxonomien haben Gemeinsamkeiten und Unterschiede. Einige der genannten Kategorien sind eher für die Konzeption der Interaktion interessant, z. B. ob eine Geste eine Entsprechung in der realen Welt haben muss oder nicht. Die wesentliche Gemeinsamkeit ist aber, dass Gesten nicht nur nach ihrer Art, sondern auch nach verschiedenen Eigenschaften klassifiziert werden. So gehen sowohl Wobbrock et al. (2009) als auch Zhai et al. (2012) darauf ein, dass Gesten abhängig von virtuellen Objekten sein können. Eine ausschließliche Verarbeitung der Eingabedaten reicht nicht aus, sondern sie müssen auch mit dynamischen Aspekten der Anwendung abgeglichen werden. Gleichzeitig ist somit z. B. relevant, wo eine Geste ausgeführt wird (vgl. Kap. 5.1.10). Weitere Eigenschaften wie die Reaktion zur Ausführungszeit (bei Wobbrock et al. (2009) als *Fluss* bezeichnet, vgl. Kap. 5.1.11) oder die zeitliche Relation der Gestenzüge zueinander (bei Zhai et al. (2012) als Gesten erster und höherer Ordnung bezeichnet, vgl. Kap. 5.2.1) spielen ebenfalls eine Rolle. Es liegt daher nahe, dass eine Betrachtung der Gesteneigenschaften sinnvoll ist, um die notwendigen Entscheidungen zu treffen und die von den Sensoren benötigten Informationen über die Interaktion zu definieren. Bei der Definition von GestureCards, der Notation zur Beschreibung von Gesten, die im Rahmen dieser Arbeit entwickelt wurde, haben sich rückblickend die in Kap. 5 vorgestellten Eigenschaften als interessantere Quelle im Vergleich zu den betrachteten Taxonomien herausgestellt.

4.1.2 Gesten und Posen

Unter dem Begriff *Gestensteuerung* werden typischerweise zwei sehr ähnliche Arten der Interaktion zusammengefasst, deren Unterschied sich im Detail verbirgt, der aber für die Entwicklung gestenbasierter Anwendungen durchaus relevant ist. Auch verwischen in der Literatur häufig die Unterschiede, da sie im Kontext zwischenmenschlicher Kommunikation oft synonym verwendet

werden. Im Rahmen dieser Arbeit soll zwischen *Gesten* und *Posen* unterschieden werden. Andere Autoren (z. B. Choi et al. (2014) oder de Souza Alcantara et al. (2012)) verwenden die Begriffe *dynamische* und *statische* Gesten, meinen aber letztlich die gleichen Konzepte.

Bull (1987, Kap. 1) definiert den Unterschied zwischen Gesten und Posen anhand der Bewegung bzw. dem Fehlen von Bewegung, also einer statischen Position. Überträgt man diese Definition in den Bereich von HCI, dann ist eine Geste demnach eine Bewegung von Körperteilen bzw. Werkzeugen durch den Raum bzw. auf einer Oberfläche. Eine Pose hingegen ist eine Positionierung von Körperteilen oder auch Werkzeugen, die zur Interaktion eingesetzt werden, im Raum bzw. auf einer Oberfläche. Diese einfache Definition muss im Sinne dieser Arbeit, und insbesondere im Sinne von HCI, noch um den Aspekt der semantischen Bedeutung für die Anwendung erweitert werden, denn nicht jede Bewegung bzw. Positionierung ist für die Anwendung relevant. Dorau (2011, S. 170) spricht in diesem Zusammenhang von der *kommunikativen Absicht* des Benutzers. Bei der Spezifikation muss also klar werden, wann und wie Benutzer mit der Anwendung kommunizieren. Gesten und Posen sind im Sinne dieser Arbeit daher wie folgt definiert:

Eine Geste ist eine Bewegung des Nutzers mit einem Werkzeug, die eine semantische Bedeutung für die Anwendung hat. Eine Pose ist eine Positionierung von Werkzeugen, die eine semantische Bedeutung für die Anwendung hat.

Eine weitere Unterscheidung, z. B. im Sinne von statischen und dynamischen Posen wie bei Wobbrock et al. (2009), erfolgt nicht. Nach der o. g. Definition sind statische Posen synonym mit Posen. Dynamische Posen hingegen sind – im Sinne der o. g. Definition – als Gesten zu betrachten. Zur Unterscheidung ist heranzuziehen, ob die Bewegung zur Pose hin eine semantische Bedeutung für die Anwendung hat oder nicht. Wenn dem so ist, dann handelt es sich allenfalls um eine Geste, die in eine Pose mündet.

Der Einschluss der semantischen Bedeutung an dieser Stelle bleibt nicht ohne Folgen, denn damit ist nicht jede Anwendung, die z. B. berührungsempfindliche Oberflächen einsetzt, automatisch eine gestenbasierte Anwendung. Andere Autoren, z. B. Zhai et al. (2012), schließen die semantische Bedeutung nicht ein und definieren eine Geste lediglich als Bewegung.[2]

[2] „We broadly define stroke gestures as the movement trajectories of a user's finger or stylus contact points with a touch sensitive surface“ (Zhai et al., 2012).

Ein Beispiel für eine ganze Klasse von Anwendungen, die zwar nach Zhai et al. (2012), aber nicht im Sinne dieser Arbeit gestenbasiert sind, sind *Swipe Keyboards* (z. B. Kristensson & Zhai (2004), Sim (2012) oder Zhai & Kristensson (2012)), wie sie u. a. für Smartphones eingesetzt werden. Vereinfacht ausgedrückt können Anwender bei Swipe Keyboards einfach über die Tastatur von einem Buchstaben zum nächsten Wischen, anstatt jeden Buchstaben einzeln anzutippen. Anhand unterschiedlicher Metriken wird dann mithilfe eines passenden Algorithmus und einem Wörterbuch das eingegebene Wort identifiziert. Die Konsequenz ist aber, dass die Bewegungen, die letztlich zu Worten und somit zu semantisch sinnvollen Eingaben führen, nicht im Vorhinein genau spezifiziert werden können. Außerdem können sie sich ändern, z. B. wenn das Tastaturlayout angepasst wird. Demgegenüber stehen Texteingabesysteme, die auf unterschiedlichen, als Geste ausgeführten Repräsentationen

von Buchstaben aufbauen (z. B. Goldberg & Richardson (1993)). Diese sind tatsächlich gestenbasiert im Sinne der o. g. Definition.

Ungeachtet dessen folgen aus der gegebenen Definition von Gesten und Posen Konsequenzen für die Entwicklung gestenbasierter Anwendungen. Die Möglichkeiten, Posen bei Oberflächengesten einzusetzen, sind begrenzt und reduzieren sich auf einfache Berührungen ohne Bewegung, die oft als *Taps* oder *Touches* bezeichnet werden.[3] Folglich ist bei Oberflächengesten in erster Linie mit tatsächlichen Gesten – also Bewegungen – zu rechnen. Räumliche Gesten hingegen ermöglichen eine Vielzahl unterschiedlicher Posen, z. B. das Zeigen des ausgestreckten Daumens zur Bestätigung einer Aktion. Allerdings gehen Posen hier zwangsläufig mit Bewegung einher, nämlich der Bewegung hin zur und weg von der Pose. Die Herausforderung für Software-Entwickler ist, die für die Anwendung nicht interessante Bewegung herauszufiltern und nur die Pose als gültig zu erkennen (vgl. Kap. 4.3.2).

[3] Dennoch können hierbei zusätzliche, interessante Informationen erfasst und zur Interaktion genutzt werden (vgl. Kap. 5.1.2, Kap. 5.1.5 und Kap. 5.1.13).

4.2 Gestenunterstützung in typischen Frameworks

In den vergangenen Jahren sind einige Frameworks zur Implementierung gestenbasierter Anwendungen veröffentlicht worden, insbesondere für mobile Geräte, für die Gestensteuerung die maßgebliche Interaktionsmodalität ist. Im Folgenden soll anhand einiger kommerziell verfügbarer Beispiele gezeigt werden, wie Gesten in verbreiteten Frameworks implementiert werden.

4.2.1 Frameworks für Oberflächengesten

iOS, das Betriebssystem für mobile Geräte der Firma Apple, bietet im Wesentlichen zwei Möglichkeiten, Gesten zu integrieren. Verschiedene UI-Elemente bringen vorgefertigte Gesten mit: Listen z. B. können ohne Weiteres über eine Geste mit einem Finger hoch- und runtergescrollt werden. Darüber hinaus stellt iOS acht Standardgesten zur Verfügung, denen grundlegende Aktionen zugeordnet werden und die Entwickler möglichst auch wie vom Hersteller vorgesehen verwenden sollten (Apple Developer, 2017a). Neben sieben Oberflächengesten wird bei iOS auch eine räumliche Geste verwendet: Um Eingaben rückgängig zu machen, kann das Gerät geschüttelt werden. Außerdem gibt es einige zusätzliche Gesten, die für bestimmte Aktionen des Betriebssystems reserviert sind – so kann das Kontrollzentrum mit einem Wisch vom unteren Rand nach oben geöffnet werden – und die nur in Ausnahmefällen umdefiniert werden sollten. Auf einigen Geräten mit iOS gibt es außerdem spezifische Gesten, die ausschließlich auf dieser Geräteklasse vorhanden sind. Auf dem iPad kann z. B. im Gegensatz zum iPhone mit vier Fingern zwischen aktiven Apps hin- und hergewischt werden.

Der zentrale Zugriffspunkt zur Integration von Gestensteuerung ist für Entwickler die Klasse `UIGestureRecognizer` (Apple Developer, 2017b). Für die eingangs erwähnten Standardgesten[4], die mit iOS integriert werden können, gibt es eine abgeleitete Klasse, die die jeweilige Geste abbildet. Eigene Gesten können durch Erweiterung von `UIGestureRecognizer`

[4] Mit Ausnahme der räumlichen Geste, um Eingaben rückgängig zu machen; diese wird über eine Eigenschaft der Anwendung gesteuert (Apple Developer, 2015).

Geste	Beschreibung	Aktion
Tap	Berührung mit einem Finger	Aktivierung/Selektion von Objekten/-Elementen
Double Tap	Doppelte Berührung mit einem Finger	Herein-/Herauszoomen
Drag	Führen des Fingers über den Bildschirm	Verschieben von Objekten/Elementen
Flick (Fling)	Schnelle, gradlinige Bewegung in eine beliebige Richtung	Schnelles Navigieren
Swipe	Gradlinige Bewegung in eine beliebige Richtung	Navigieren
Pinch/Spread	Zusammenführen/Auseinanderziehen zweier Finger	Herein-/Herauszoomen
Touch and Hold	Berühren und Halten mit einem Finger	Markierung/Multi-Selektion/Öffnen eines Kontextmenus
Rotate	Kreisförmige Bewegung mit zwei Fingern	Drehen

Tabelle 4.1: De-Facto-Standard für Oberflächengesten

integriert werden. Einzelne Berührungen des Bildschirms werden durch die Klasse `UITouch` (Apple Developer, oJ[b]) repräsentiert und müssen zur Identifikation der Geste verarbeitet werden.

Android, das vorrangig von Google entwickelt wird, bietet ähnliche Möglichkeiten, Gesten in Anwendungen zu verwenden. Auch in diesem Framework sind einige UI-Elemente mit Gestensteuerung versehen, die ohne zusätzliche Arbeiten verwendet werden können. Für Android werden 14 Standardgesten vorgeschlagen (material.io, oJ). Dabei überschneiden sich die Standardgesten mit denen in iOS, fügen aber einige Varianten hinzu, bei denen zwei Finger eingesetzt werden. Eine räumliche Geste wie das Schütteln unter iOS wird nicht vorgeschlagen. Der Leitfaden für Entwickler schlägt auch – im Gegensatz zu iOS – verschiedene potentielle Gesten für dieselbe Aktion vor. So werden z. B. drei unterschiedliche Gesten für die Aktion *Zoom In* empfohlen: *Pinch Open* (Auseinanderziehen zweier Finger), *Double-Touch* (doppelt antippen) und *Double-Touch Drag (down)* (doppelt antippen und nach unten ziehen) (material.io, oJ).

Außerdem gibt es auch unter Android eine Menge von Klassen, die zur Identifikation der Standardgesten eingesetzt werden. Zentral ist dazu die Klasse `GestureDetector` (Android Developers, oJ[e]), die den Einstieg in den integrierten Erkennungsprozess bietet und unterschiedliche Gesten identifizieren kann. Die Klasse `MotionEvent` (Android Developers, oJ[b]) enthält alle Informationen einer Berührung des Bildschirms und muss in den o. g. Klassen verarbeitet werden.

Android liefert darüber hinaus noch die Möglichkeit, Gesten mithilfe der *Gesture Libraries* (Android Developers, oJ[a]) aufzunehmen. Dabei können Gesten mit dem *Gesture Builder*, einer speziellen Anwendung, die auf dem Android Emulator vorinstalliert ist, aufgezeichnet, gespeichert und anschließend in die Anwendung importiert werden. Somit können Muster von Gesten zur Verfügung gestellt werden und die Nutzereingaben dagegen abgeglichen werden. Dieses Vorgehen ist als musterbasierter Ansatz zur Beschreibung und

Identifikation von Gesten bekannt und geht mit unterschiedlichen Vor- und Nachteilen einher (vgl. Kap. 4.3.2 und Kap. 6.1).

Die Betrachtung der beiden meistverbreiteten[5] Frameworks für die Entwicklung mobiler Anwendungen zeigt, dass sich in den letzten Jahren der in Tab. 4.1 dargestellte *De-Facto-Standard* für Oberflächengesten herauskristallisiert hat, der auf beiden Plattformen zu finden ist. Die darin vertretenen Gesten werden mit vorgefertigten Komponenten unterstützt und bereits z. T. für bestimmte Aktionen vorgeschlagen. Entwickler sind gut beraten, den Weisungen der Hersteller hier zu folgen; schließlich ist eine herausragende Eigenschaft von Ökosystemen wie denen mobiler Betriebssysteme, dass sich grundlegende Dinge plattformweit wiederfinden. Allerdings zeigt die Betrachtung auch, dass Entwickler spätestens bei der Verwendung individueller Gesten lediglich die (aufbereiteten) Rohdaten zur Verfügung gestellt bekommen und diese eigenhändig weiterverarbeiten müssen. Dennoch müssen sie zunächst prüfen, ob vorgefertigte Komponenten zum Einsatz kommen können, bevor eigene Komponenten entwickelt werden.

[5] Prognosen erwarten, dass der Markt für mobile Betriebssysteme bis zum Jahr 2021 nahezu vollständig unter den Genannten aufgeteilt ist (IDC, 2017).

4.2.2 Frameworks für räumliche Gesten

Der Leap Motion Controller ist ein kleines Gerät, dass auf den Tisch gestellt wird und mithilfe von zwei Cameras und drei LEDs (Colgan, 2014) die Hände, einzelne Finger und Werkzeuge wie Stifte erkennen kann. Das Software Development Kit (SDK) bietet Unterstützung für drei Gesten (Leap Motion Developer, oJ[d]): eine kreisförmige Bewegung (*Circle*), eine gradlinige Bewegung (*Swipe*) sowie das Antippen (*Tap*). Letztere wird nochmal in zwei unterschiedliche Variationen aufgeteilt: ein Antippen nach unten (*Key Tap*) und nach vorne (*Screen Tap*). Für jede Geste gibt es eine passende Klasse, die zur Erkennung eingesetzt werden kann. Allerdings wurden diese mit der aktuellen Version des SDK als *veraltet* markiert.

Die grundlegende Datenstruktur des Leap Motion Controllers ist die Klasse `Frame` (Leap Motion Developer, oJ[c]), die Informationen über die aktuell in der Sphäre vorhandenen Hände enthält. Hier findet sich auch eine Liste von Gesten, die zum aktuellen Zeitpunkt durchgeführt oder beendet wurden. Diese Datenstruktur enthält die notwendigen Rohdaten, die zur Identifikation individueller Gesten verwendet werden müssen. Es werden Information wie verwendete Finger und Koordinaten im Raum übermittelt.

Die HoloLens, eine Brille, mit der virtuelle Hologramme in das Blickfeld des Trägers eingeblendet werden können, unterstützt zwei wesentliche Gesten (Microsoft Developer, oJ): Antippen mit dem Finger (*Air Tap*) und die Öffnung der Hand (*Bloom*). Während Entwickler Erstere selber einsetzen können, ist Letztere für das System reserviert und wird verwendet, um das Startmenü der HoloLens zu öffnen. Die Geste Air Tap kann außerdem durch ein längeres Halten erweitert werden, um z. B. virtuelle Objekte zu greifen und zu verschieben. Außerdem können virtuelle Objekte durch Bewegungen der Hände manipuliert werden, indem die Bewegung direkt in der Anwendung umgesetzt wird. Darüber hinaus bietet die HoloLens noch UI-Widgets, die analog zu mobilen Plattformen mit Gesten bedient werden können.

Auffällig ist, dass die betrachteten Frameworks für räumliche Gesten weniger Gesten zur Verfügung stellen als Frameworks für mobile Anwendungen. Es ist anzunehmen, dass dafür zwei wesentliche Gründe im Vordergrund stehen: die (noch) geringe kommerzielle Verfügbarkeit passender Geräte und Frameworks sowie der Mangel an standardisierten UI-Elementen, die in Anwendungen, für die räumliche Gesten interessant sind, verwendet werden können. Die Entwicklung steht – im Gegensatz zu mobilen Systemen – in diesem Bereich noch am Anfang und ein ähnlicher, übergreifender De-Facto-Standard konnte sich noch nicht entwickeln. An den Herausforderungen für Entwickler ändert dies hingegen nichts, im Gegenteil: Dadurch, dass weniger vorgefertigte Komponenten seitens der Hersteller zur Verfügung stehen, sind Entwickler gezwungen, alternative Komponenten zu entwickeln.

4.3 Notwendige technische Entscheidungen

Die zur Implementierung gestenbasierter Anwendungen notwendigen Komponenten müssen in Abhängigkeit der eingesetzten Gesten ausgewählt oder entwickelt werden. Die notwendigen Entscheidungen sollen anhand der in Abb. 4.1 dargestellten, typischen Architektur für Anwendungen, die auf Multi-Touch-Gesten basieren (Echtler & Klinker, 2008), veranschaulicht werden. Diese Architektur kann auch auf Anwendungen mit räumlichen Gesten übertragen werden und ist als klassische Schichtarchitektur mit fünf Schichten konzipiert. Daten fließen von der untersten Schicht, auf der sie aufgenommen werden, nach oben und werden von jeder Schicht verarbeitet und mit weiteren Informationen angereichert bzw. generalisiert. Eine ähnliche Architektur beschreiben Laufs & Ruff (2013), jedoch fassen sie die Transformations- und Hardware-Abstraktionsschichten zusammen.

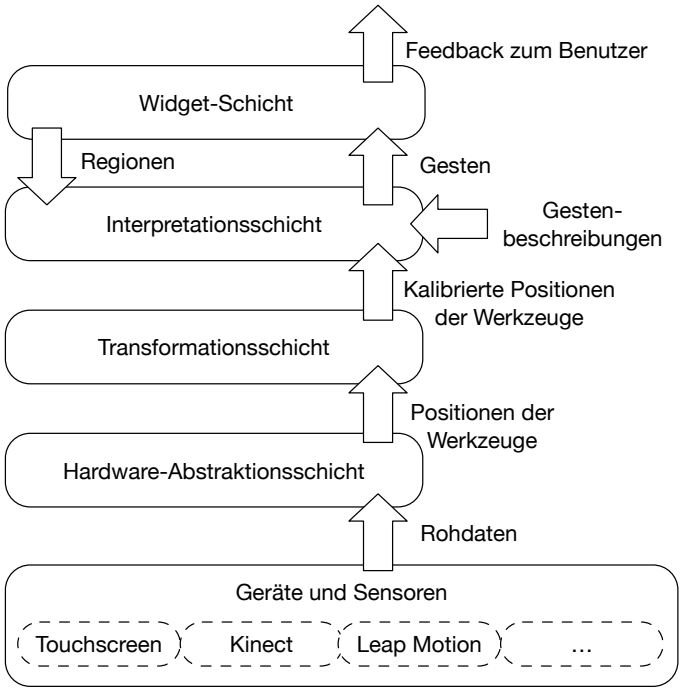


Abbildung 4.1: Grundlegende Architektur gestenbasierter Anwendungen (Bildquelle: Eigene Darstellung nach Echtler & Klinker (2008))

Auf der untersten Ebene sind die verwendeten *Geräte und Sensoren* zu finden. Sie nehmen die Eingaben der Nutzer auf und stellen sie als Rohdaten zur Verfügung. Aufbau und Formatierung der Rohdaten sind hardware-spezifisch und nicht vereinheitlicht. Das ist in der Regel auch nicht notwendig, denn Entwickler müssen typischerweise auch nicht auf dieser Ebene mit den Daten arbeiten, sondern können sich auf die Ergebnisse höherer Schichten stützen.

Die folgende *Hardware-Abstraktionsschicht* nimmt die spezifischen Rohdaten der Geräte auf und wandelt sie in allgemeinere Daten um, die oftmals zunächst von der verwendeten Plattform verarbeitet und dann Entwicklern zum Einsatz in Anwendungen zur Verfügung gestellt werden. Diese Aufgabe wird in der Regel von vorgefertigten Komponenten übernommen, die die Hersteller der Sensoren in Form von Treibern und SDKs ausliefern. Interessant sind an dieser Stelle Protokolle, die versuchen, die gelieferten Informationen zu vereinheitlichen. Ein Beispiel dafür ist TUIO (Kaltenbrunner et al., 2005).

Für manche Anwendungen ist eine weitere Verarbeitung sinnvoll, wenn die Daten noch nicht in der passenden Form vorliegen. Wenn z. B. mithilfe räumlicher Gesten ein klassisches UI bedient werden soll, müssen die Daten der räumlichen Geste erst passend in das zweidimensionale Koordinatensystem der Anwendung umgewandelt werden. Diese Aufgabe übernehmen die Komponenten der *Transformationsschicht*. Je nach Unterstützung durch den Hersteller können hierfür schon vorgegebene Komponenten vorhanden sein, jedoch kann davon nicht ausgegangen werden. Wenn die unterliegende Schicht schon passende Daten liefert, dann kann die Transformationsschicht entfallen bzw. die Daten einfach an die nächste Schicht durchreichen.

Die folgende Schicht, die *Interpretationsschicht*, verarbeitet die transformierten Eingabedaten und versucht, daraus für die Anwendung sinnvolle Eingaben abzuleiten. Im Falle einer gestenbasierten Anwendung werden hier die ausgeführten Gesten identifiziert. Die wesentliche Komponente dafür ist der *Gestenerkenner*, der mit einer Menge gültiger Gesten konfiguriert wird. Der Gestenerkenner sucht im Prinzip nach Gemeinsamkeiten zwischen ihm bekannten und vom Benutzer ausgeführten Gesten, indem er den Strom der Eingabedaten überwacht und kontinuierlich mit seiner Konfiguration abgleicht.

Erfolgreich identifizierte Gesten werden an die letzte Schicht, die *Widget-Schicht*, gemeldet. Typischerweise werden hierfür Ereignisse ausgelöst, auf die die Anwendung dann reagieren und dem Nutzer Rückmeldung geben kann. Die Widget-Schicht kennt aber auch die Gestaltung der Oberfläche und muss daher der Interpretationsschicht zurückmelden, wo Gesten gültig sein können. Dies kann z. B. die aktuelle Position virtueller Objekte sein. Die Widget-Schicht muss also zur Laufzeit der Anwendung die Komponenten der Interpretationsschicht mit neuen Informationen über den aktuellen Aufbau der Oberfläche aktualisieren können.

Der parallele Einsatz mehrerer Sensoren ist an dieser Stelle interessant. Die von den einzelnen Sensoren übermittelten Informationen müssen an geeigneter Stelle zusammengeführt werden, denn sie sind Teil einer einzigen Interaktion. Hierzu bieten sich in der vorgestellten Architektur zwei Stellen an: in

der Transformations- oder der Interpretationsschicht. In der Transformationsschicht könnten die Daten aller Sensoren zu einem Datensatz der Interaktion vereinheitlicht werden. Somit würde in den nachfolgenden Schichten von unterschiedlichen Sensoren weitgehend abstrahiert und die Datenhaltung zentralisiert werden. Ein Protokoll wie das bereits erwähnte TUIO (Kaltenbrunner et al., 2005), das eine einheitliche Schnittstelle zur Übertragung der Eingabedaten gestenbasierter Systeme liefert, ist an dieser Stelle besonders interessant. Die Alternative wäre, die Daten erst in der Interpretationsschicht zusammenzuführen und hier mit mehreren Komponenten zu arbeiten, die für die Analyse der einzelnen Sensordaten verantwortlich sind. Die Daten jedes Sensors würden einzeln analysiert und die Ergebnisse anschließend kombiniert. In beiden Fällen müssen die Daten der verschiedenen Sensoren in einen zeitlichen Kontext gebracht werden.

In der vorgestellten Architektur sind zwei Schichten erkennbar, die bei der Auswahl der Komponenten explizit betrachtet werden müssen: die unterste Schicht mit den verwendeten Geräten und Sensoren sowie die Interpretationsschicht, in der der Gestenerkenner implementiert wird. Die anderen Schichten sind in der Regel abhängig von diesen Entscheidungen; insbesondere die Auswahl der Geräte und Sensoren bringt oftmals die notwendigen Komponenten mit sich. Daher soll im Folgenden ein Blick sowohl auf unterschiedliche Hardware und Sensoren sowie auf die grundlegende Funktionsweise von Gestenerkennern geworfen werden.

4.3.1 Hardware und Sensoren

Gestensteuerung wird insbesondere durch die Verfügbarkeit moderner Sensoren möglich gemacht, die verschiedene Informationen über Gesten und Posen des Nutzers sammeln kann. Diese Informationen werden dann in den verschiedenen Schichten, die in Abb. 4.1 zu sehen sind, aufbereitet und Entwicklern zur Verfügung gestellt. Viele Sensoren werden dabei von den Herstellern mit passenden SDKs ausgeliefert, sodass Software-Entwickler erst auf der Interpretationsschicht mit den Daten der Sensoren arbeiten müssen.

Die Rolle der Sensoren ist also recht einfach zusammengefasst: Sie sind vor allem als Datenlieferant notwendig und stellen die Grundlage für die weitere Verarbeitung (insbesondere der Erkennung der Gesten, vgl. Kap. 4.3.2) zur Verfügung. Bei der Entwicklung gestenbasierter Anwendungen muss somit zwangsläufig eine Auswahl passender Sensoren getroffen werden. Entwickler müssen prüfen, welche Sensoren die notwendigen Informationen in einem passenden Format bereitstellen, und eine entsprechende Entscheidung für oder gegen einen Sensor treffen. Informationen, die von den Sensoren nicht direkt zur Verfügung gestellt wird, können zwar u. U. aus anderen Informationen abgeleitet werden (vgl. z. B. Kap. 5.1.6), jedoch ist auch hier eine entsprechende Prüfung vorab notwendig, um möglichst früh die optimale Entscheidung treffen zu können.

Die Begriffe *Hardware* und *Sensor* sind je nach Anwendung synonym – streng genommen ist ein Sensor auch nur ein Stück Hardware. Allerdings gibt es durchaus Unterschiede, die eine Trennung der Begrifflichkeiten sinnvoll ma-

chen. So kann es sein, dass ein Sensor direkt im verwendeten Gerät verbaut ist (z. B. das Accelerometer in einem mobilen Gerät) oder aber bestehende Hardware um zusätzliche Fähigkeiten erweitert werden muss, die durch externe Sensoren zur Verfügung gestellt werden (z. B. die Erkennung unterschiedlicher Benutzer mithilfe einer Kamera). Das bedeutet, dass eigentlich zwei Entscheidungen getroffen werden müssen: auf welchen Geräten die Anwendung läuft und welche Informationen für die Interaktion relevant sind. Letztere sind die Grundlage zur Auswahl der Sensoren.

Als Beispiel soll im Folgenden auf einen weit verbreiteten Sensor eingegangen werden, der auf den ersten Blick nicht als solcher zu erkennen ist: berührungsempfindliche Bildschirme, auch *Touchscreens* genannt. Diese gibt es in unterschiedlichen Größen und auf Basis verschiedener Technologien. Verbreitet sind *resistive* und *kapazitive* Touchscreens,[6] die sich nicht nur im technischen Aufbau unterscheiden, sondern auch unterschiedliche Vor- und Nachteile haben. Bei einem kapazitiven Touchscreen wird ein elektrisches Feld über das Display gelegt, das sich bei Berührung verändert. Dadurch kann die Position der Berührung gemessen werden. Es muss kein Druck ausgeübt, sondern lediglich Kontakt mit der Oberfläche hergestellt werden. Resistive Touchscreens hingegen benötigen etwas Druck, damit die Position der Berührung ermittelt werden kann. Sie bestehen aus zwei Schichten, zwischen denen sich ebenfalls ein elektrisches Feld befindet. Kapazitive Touchscreens haben den Vorteil, dass mehrere Finger gleichzeitig auf der Oberfläche erkannt werden können, während resistive Touchscreens auf einen Finger beschränkt sind. Demgegenüber steht, dass resistive Touchscreens auch mit Handschuhen oder Gegenständen bedient werden können, während kapazitive Touchscreens nur mit den Fingern oder speziell verarbeiteten Gegenständen/Handschuhen verwendet werden können (Laufs & Ruff, 2013).

[6] Über die beiden genannten hinaus existieren auch noch weitere Verfahren, die z. B. mit optischen oder akustischen Informationen arbeiten; Walker (2012) bietet einen Überblick.

Das Beispiel zeigt, welche Entscheidungen Entwickler an dieser Stelle treffen müssen. Beide Verfahren können zur Interaktion mit Oberflächengesten eingesetzt werden, jedoch ist von den zu integrierenden Gesten abhängig, welches besser geeignet ist: Resistive Touchscreens ermöglichen zwar Druck auf die Oberfläche, schränken aber die Anzahl der parallelen Berührungen ein. Die Hardware und die verwendeten Sensoren müssen somit in Abhängigkeit der Anforderungen ausgewählt werden, die durch die zu integrierenden Gesten vorgegeben sind. Wenn die Hardware schon vorhanden sein sollte, dann muss das angedachte Interaktionskonzept mit den Fähigkeiten der Sensoren und den Informationen, die sie zur Verfügung stellen, abgeglichen werden.

4.3.2 Gestenerkenner

Rubine (1991) definiert den *Gestenerkenner* als die Komponente, die die Eingaben des Benutzers als der Anwendung bekannte und gültige Geste identifiziert. Typischerweise ist der Gestenerkenner in der Interpretationsschicht einer Anwendung zu finden. Ein Gestenerkenner funktioniert grob gefasst in den in Abb. 4.2 dargestellten drei Phasen: Gesten registrieren, Eingabedaten empfangen und Gesten identifizieren. Kin et al. (2012b) weisen darauf hin, dass sich der zur Erkennung von Gesten nötige Quellcode in

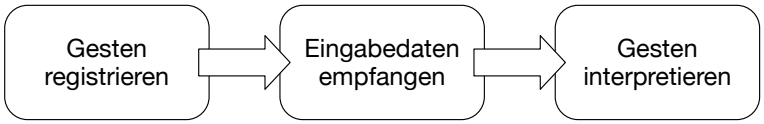


Abbildung 4.2: Phasen der Gestenerkennung

gängigen Frameworks jedoch u. U. auf mehrere Komponenten verteilt und Software-Entwickler somit dazu zwingt, den aktuellen Ausführungszustand zu speichern und zwischen den beteiligten Komponenten hin und her zu kommunizieren. Dazu sind z. B. global gesetzte Zustände zur Verfolgung einzelner Gesten notwendig, die zum richtigen Zeitpunkt zurückgesetzt werden müssen. Kin et al. (2012b) bezeichnen dies treffend als *Callback Soup*; Spano et al. (2013) sprechen in diesem Kontext von *Spaghetti Code*. Daraus folgen insbesondere Probleme für Wartung, Erweiterung und Fehlersuche, denn Entwickler sind gezwungen, für jede mögliche Geste die entsprechenden Informationen über verschiedene Komponenten nachzuverfolgen. Gleichzeitig ist nicht auszuschließen, dass sich die Implementierungen verschiedener Gesten durch Mehrdeutigkeiten sowie ungenaue Ausführung durch Benutzer in die Quere kommen, wodurch es zu Fehlerkennungen kommt.

In der ersten Phase werden dem Gestenerkenner alle Gesten mitgeteilt, die er im Folgenden erkennen soll (in Abb. 4.1 durch den Pfeil rechts der Interpretationsschicht angedeutet). Dies kann auf unterschiedliche Art und Weise geschehen. Die einfachste – und nach Rubine (1991) auch früher häufig praktizierte – Variante ist, die Gesten hart im Erkenner zu kodieren. Tatsächlich verwenden zahlreiche gängige Frameworks, die vorgefertigte Komponenten zur Erkennung bestimmter Gesten bereitstellen (vgl. Kap. 4.2), immer noch diesen Ansatz. Eine Alternative ist, die Beschreibung von Gesten auszulagern und eine generalisierte Komponente zur Erkennung von Gesten zu konfigurieren, um größere Abstraktion zu schaffen und die Wartbarkeit der Anwendung zu verbessern. Heute verbreitet sind musterbasierte Verfahren (vgl. Kap. 6.1) in unterschiedlichen Ausprägungen, denen bekannte Gesten als reproduzierbare Muster zur Verfügung gestellt werden; aber auch textuelle und graphische Notationen (vgl. Kap. 6.2 und Kap. 6.3) können verwendet werden.

In der zweiten Phase wird der Gestenerkenner mit Eingabedaten des Benutzers versorgt, die in den unteren Schichten der in Abb. 4.1 dargestellten Architektur vorverarbeitet und zur Verfügung gestellt werden. In der Regel wird hier ein Strom grundlegender Daten der Interaktion geliefert, die zwar bereits passend aufbereitet wurden, aber lediglich einfache, diskrete Informationen über den aktuellen Zustand zum gegenwärtigen Zeitpunkt liefern. Sie bieten die Grundlage, auf der der Gestenerkenner arbeitet. Je nach verwendeten Sensoren können sich diese Daten im Aufbau und Informationsgehalt unterscheiden, sodass hier bereits Einschränkungen vorliegen bzw. passende Entscheidungen getroffen werden müssen.

In der letzten Phase findet dann die eigentliche Erkennung statt. Die empfangenen Eingabedaten müssen in einen zeitlichen und räumlichen Bezug gebracht und mit den eingangs registrierten Gesten abgeglichen werden. Jede erfolgreiche Übereinstimmung (idealerweise auch versehen mit einem Wert für die Konfidenz) führt zu einer Erkennung, in deren Folge typischerweise

die Schicht, die für die Steuerung des UI verantwortlich ist, informiert wird – in Abb. 4.1 die Widget-Schicht.

Zur Erkennung von Gesten wurden in den letzten Jahren zahlreiche Verfahren mit verschiedenen Vor- und Nachteilen entwickelt.[7] Insbesondere sind Gestenerkenner oft nur für eine bestimmte Art von Gesten geeignet: entweder Oberflächengesten oder räumliche Gesten. Ein passender Gestenerkenner muss auch wichtige Eigenschaften der Interaktion (vgl. Kap. 5) erkennen können. So kann die Richtung der Ausführung für die Anwendung relevant sein, jedoch geht diese bei verschiedenen Verfahren[8] verloren. Außerdem basieren viele Gestenerkenner auf einer Datengrundlage, die nur von bestimmten Sensoren zur Verfügung gestellt wird. So baut z. B. der Gestenerkenner von Kratz & Rohs (2010) auf den Daten eines Accelerometers auf. Accelerometer übermitteln aber Beschleunigungsdaten im Raum, während andere Sensoren nur Positionen in einem zwei- oder dreidimensionalen Koordinatensystem weitergeben. Die Folge ist, dass Erkennerverfahren meist nur für bestimmte Eingaben gut geeignet sind.

[7] Einen Überblick liefern z. B. Mitra & Acharya (2007), Cirelli & Nakamura (2014) und D'Orazio et al. (2016).

[8] Insbesondere die populären Vertreter der $-Familie (vgl. Kap. 6.1) unterliegen hier verschiedenen Einschränkungen.

Software-Entwickler müssen folglich für die zu implementierende Anwendung eine passende Auswahl unter den zahlreichen Verfahren treffen, um für die unterschiedlichen Anforderungen, die sich aus den zu verwendenden Gesten ergeben, die optimale Lösung zu finden. Dabei müssen unterschiedliche Faktoren in Betracht gezogen werden. Zu berücksichtigen sind dabei:

- Die verwendeten Sensoren, da sie letztlich mitbestimmen, in welchem Format die zugrundeliegenden Daten vorliegen und auch unterschiedliche Anforderungen an die Erkennung stellen (vgl. Kap. 4.3.1)
- Die Art der Gesten (vgl. Kap. 4.1); Gestenerkenner sind oftmals auf Oberflächengesten oder räumliche Gesten spezialisiert
- Die unterschiedlichen Eigenschaften, die Gesten haben können (vgl. Kap. 5); insbesondere die zeitliche Relation (vgl. Kap. 5.2.1)
- Die Genauigkeit, die vom Erkenner erwartet wird
- Die Robustheit gegenüber ungenau ausgeführten Gesten

Diese fünf Punkte werden in erster Linie durch die für die Anwendung vorgesehen Gesten definiert. Somit hängt auch die Auswahl des Gestenerkenners direkt an den definierten Gesten. Entwickler müssen also ein grundlegendes Verständnis der Gesten haben, um die bestmögliche Auswahl zu ermöglichen. Es ist davon auszugehen, dass eine Spezifikation der Gesten in einer passenden Notation dieses Verständnis unterstützen kann. Dazu müssen alle Eigenschaften, die die Gesten ausmachen, exakt beschrieben werden können.

5 Eigenschaften gestenbasierter Interaktion

Zur Spezifikation von Gesten muss identifiziert werden, welche Informationen beschrieben werden müssen, um eine Geste mit allen notwendigen Merkmalen definieren zu können. Im Folgenden sollen die unterschiedlichen Eigenschaften von Gestensteuerung vorgestellt werden, die die Grundlage und Rahmenbedingungen für die in Kap. 7 beschriebene Notation liefern. Für jede identifizierte Eigenschaft werden die Informationen definiert, die spezifiziert werden müssen, und welchen besonderen Regeln sie unterliegen.

Buxton (2014) fasst zusammen, dass insbesondere Multi-Touch-Sensoren oftmals lediglich den Ort der Berührung erkennen, jedoch wesentlich mehr Möglichkeiten für eine reichhaltigere Interaktion und UX bestehen, wenn weitere Eigenschaften der ausgeführten Geste von Anwendungen aufgegriffen und verwendet werden.[1] Er geht insbesondere auf die Möglichkeit ein, Kräfte zu messen, die auf die Oberfläche einwirken, oder die Ausmaße der berührten Fläche festzustellen. Es ist davon auszugehen, dass auch bei räumlichen Gesten weit mehr als die bloße Position von Händen und Fingern im Raum eine Rolle spielen kann. Auch Saffer (2008) stellt heraus, dass die Gesten unterschiedliche Eigenschaften haben, auf deren Basis die Interaktion mit der Anwendung entworfen werden kann. Er definiert Präsenz (also die reine Anwesenheit des Benutzers[2]), Dauer, Position,[3] Bewegung, Druck auf die Oberfläche, Auflagefläche, Ausrichtung, physische Objekte, Anzahl Berührungen, Gestensequenzen und Anzahl beteiligter Benutzer. Diese Eigenschaften finden sich auch im Rahmen der folgenden Recherche wieder, müssen jedoch um einige weitere ergänzt werden.

Wang & Ren (2009) untersuchen verschiedene Eigenschaften von Fingern in Bezug auf interaktive Oberflächen und unterscheiden vier Kategorien: Position, Bewegung (*Motion*), Physisch (*Physical*) und Ereignis (*Event*). Unter der Position verstehen die Autoren allerdings eine Abstraktion hin zu einer einzelnen x-y-Koordinate und gehen nicht wie Saffer (2008) auf den Ort der Ausführung der ganzen Geste ein, die aus einer Menge an x-y-Koordinaten besteht. Sie verweisen darauf, dass aus Kompatibilitätsgründen mit klassischen PnC-Anwendungen die Berührung abstrahiert wird und dass durch die Berücksichtigung weiterer Eigenschaften eine reichhaltigere Interaktion möglich ist. In der Kategorie Bewegung unterscheiden Wang & Ren (2009) zwischen Geschwindigkeit und Beschleunigung der Finger auf der Oberfläche und in der Kategorie Physisch zwischen verschiedenen Aspekten, die mit dem Kontakt auf der Oberfläche zu tun haben. So unterscheiden sie die Größe und die Form der Auflagefläche, deren Ausrichtung und den ausgeübten Druck. In der letzten Kategorie – Ereignisse – unterscheiden die Autoren zwischen einfachen Berührungen ohne Bewegungen und schnellem Wischen.

[1] „There is more to touch-sensing than contact and position: [...] other aspects of touch [...] have the potential to enrich the user experience“ (Buxton, 2014).

[2] Ein Konzept, das z. B. Schmidt et al. (2005) auch als implizite Mensch-Maschine-Interaktion (Implicit Human-Computer-Interaction (iHCI)) bezeichnen.

[3] Wobei Saffer (2008) hiermit den Ort der Ausführung meint und nicht die Positionierung von Teilgesten zueinander.

Die genannten Arbeiten zeigen zwar interessante Aspekte von Gestensteuerung auf, es bleibt jedoch fraglich, ob diese Eigenschaften für eine Spezifikation von Gesten ausreichen: Die Autoren betrachten verschiedene interessante Optionen wie z. B. die wiederholte Ausführung einer Geste nicht und lassen außer Acht, dass neben dem Ort der Ausführung auch spezielle Ausprägungen wie Start- und Endpunkt für eine genaue Definition notwendig sein können. Initial ist daher notwendig, mögliche Gesteneigenschaften zu identifizieren und zu analysieren. Darauf aufbauend kann definiert werden, welche möglichen Informationen dann wie spezifiziert werden müssen. Dabei kann davon ausgegangen werden, dass manche Eigenschaften auf verschiedene Art beschrieben werden können bzw. unterschiedliche Ausprägungen existieren, die sich je nach Anwendungsfall und verwendeter Sensoren unterscheiden können. Zum Beispiel kann für die Spezifikation relevant sein, ob eine Geste am Benutzer oder am Sensor ausgerichtet wird: Während bei Smartphones eine Ausrichtung am Sensor sinnvoll ist (da sich das dargestellte Bild meist der Haltung des Geräts im Hoch- oder Querformat anpasst), kann bei einem berührungsempfindlichen Tisch, um den Benutzer herumgehen können, eine Ausrichtung an der Position des Benutzers interessant sein. Denkbar ist des Weiteren, dass manche Eigenschaften ausschließlich für Oberflächengesten oder räumliche Gesten beschrieben werden können. So ermöglicht der Umstand, dass bei Oberflächengesten Kontakt zu einer Oberfläche besteht, die Messung des Drucks (vgl. Kap. 5.1.6) oder der Ausmaße der Auflagefläche (vgl. Kap. 5.1.2). Gleichzeitig muss die Beschreibung räumlicher Gesten die Definition von Bewegungen im dreidimensionalen Raum ermöglichen.

Die vorliegenden Erkenntnisse basieren vorwiegend auf der Auseinandersetzung mit bestehenden Spezifikationsansätzen und wissenschaftlicher Literatur, die entweder die unterschiedlichen Eigenschaften von Gestensteuerung analysiert oder neue Arten von HCI erforscht. Die unterschiedlichen Arbeiten betrachten dabei einige der von Saffer (2008) und Buxton (2014) angesprochenen Eigenschaften sowie einige weitere und demonstrieren, wie sie in Anwendungen integriert werden können. Außerdem wurden gängige Frameworks zur Entwicklung gestenbasierter Anwendungen betrachtet und ihre Umsetzung und Integration verschiedener Eigenschaften analysiert.

Gesten lassen sich in Teilgesten zerlegen, die wiederum zu unterschiedlichen Gesten zusammengesetzt werden können. Jede Teilgeste kann als eigenständige Geste separat betrachtet und definiert werden. Sie hat somit eigene Eigenschaften, die sich lediglich auf sie selbst beziehen. Diese Eigenschaften sind demnach atomar: Sie gehören ausschließlich zu einer Geste oder Teilgeste und sind auch nur für ebendiese definiert. Ihre Definition hat keine weiteren Konsequenzen für andere Gesten. Setzt man einfache Gesten jedoch zu komplexeren Gesten zusammen, ergibt sich ein Zusammenspiel der einzelnen Teilgesten, das sich auch in zu definierenden Eigenschaften wiederfindet. Demnach müssen Eigenschaften definiert werden, die die einzelnen Teilgesten in einen Bezug zueinander setzen. Daraus folgt zwangsläufig, dass in diesem Falle immer zwei Gesten bei der Definition der Eigenschaft berücksichtigt werden müssen. Somit werden im Folgenden auch zwei Arten von Eigenschaften unterschieden und näher beschrieben: Atomare Eigenschaften (Kap. 5.1) und Beziehungen zwischen Teilgesten (Kap. 5.2).

5.1 Atomare Eigenschaften

Atomare Eigenschaften beziehen sich direkt auf eine ausgeführte Geste oder Teilgeste und stehen in keinem Bezug zu anderen Gesten. Die Eigenschaften der Gesamtgeste sind die Summe aller atomaren Eigenschaften der Teilgesten.

5.1.1 Anzahl der Werkzeuge

Eine grundlegende Eigenschaft moderner Sensoren ist die Möglichkeit, verschiedene Werkzeuge[4] einzusetzen, mit denen Gesten ausgeführt werden können. Diese können u. U. gleichzeitig erkannt werden, was letztlich die parallele Ausführung von Gestenzügen ermöglicht. Die damit verbundenen Möglichkeiten haben das Spektrum für Interaktion mit Maschinen erweitert und sorgten erst dafür, dass sich HCI über das bloße Anklicken von Buttons hinaus entwickelt hat (Buxton, 2014). Dabei beschränkt sich diese Eigenschaft zunächst auf die reine Verwendung mehrerer Werkzeuge, ohne diese spezifisch zu benennen – die konkrete Betrachtung unterschiedlicher Werkzeuge erfolgt in Kap. 5.1.13. Auch Saffer (2008) berücksichtigt in seinen Gesteneigenschaften, dass mehrere Werkzeuge verwendet werden können.

So kann z. B. auf einer berührungsempfindlichen Oberfläche eine Geste mit zwei Fingern durchgeführt werden; im Rahmen dieser Arbeit entspricht dies zwei Werkzeugen, berücksichtigt jedoch vorerst nicht, welche Finger konkret eingesetzt werden. Ein typisches Beispiel ist die bekannte Spread-Geste, bei der zwei Finger auseinandergezogen werden und die häufig zum Vergrößern eingesetzt wird. Sie gehört zum klassischen Repertoire vorhandener Gesten-Frameworks (vgl. Kap. 4.2) und findet sich hier in verschiedenen Ausprägungen wieder. Inwieweit einzelne Finger identifiziert werden können, ist jedoch vom verwendeten Sensor abhängig. Oftmals ist es nicht möglich zu ermitteln, welcher Finger konkret (z. B. Mittel- oder Zeigefinger) eingesetzt wird, sondern es kann lediglich festgestellt werden, dass es sich um unterschiedliche Finger handelt. Ähnliches gilt für räumliche Gesten. Moderne Sensoren können Hände, einzelne Finger oder Werkzeuge wie Stifte im freien Raum unterscheiden und ihre Bewegungen nachvollziehen (vgl. z. B. die Analyse der Finger mit dem Leap Motion Controller (Leap Motion Developer, oJ[b])). Analog zur Spread-Geste auf berührungsempfindlichen Oberflächen können im dreidimensionalen Raum die Hände auseinandergezogen werden.

Über die Anzahl eingesetzter Werkzeuge kann die gleiche Geste unterschiedlich verwendet werden.[5] So ermöglicht macOS die Unterscheidung verschiedener Aktionen anhand der Finger auf dem Touchpad (Apple Support, 2016):

- 1 Finger hoch/runter: Bewegen des Cursors auf dem Bildschirm
- 2 Finger hoch/runter: Scrollen
- 3 Finger hoch/runter: Markieren
- 4 Finger hoch/runter: Mission Control öffnen/schließen

[4] In verwandter wissenschaftlicher Literatur findet sich auch der Begriff *Zeiger* (oder englisch *Pointer*), der jedoch eine starke Nähe zu klassischen PnC-Anwendungen impliziert – im Rahmen dieser Arbeit wird daher auf diese Benennung verzichtet.

[5] Verschiedene Autoren (z. B. Wobbrock et al. (2009), Morris et al. (2010) und Rekik et al. (2014)) berichten jedoch, dass der Einsatz mehrerer Werkzeuge für Benutzer schwierig umzusetzen ist und daher genauestens abgewogen werden muss. Allerdings bleibt bei den genannten Arbeiten unklar, ob die Ergebnisse bei steigender Erfahrung der Probanden mit Gestensteuerung nicht anders ausfallen würden und inwiefern fehlende Lerneffekte und bestehendes Wissen über aktuelle WIMP-UIs die Ergebnisse beeinflusst haben.

Die Verwendung mehrerer Werkzeuge kann bei der Spezifikation von Gesten auf verschiedene Art und Weise angegeben werden. Sie kann explizit definiert werden, d. h. es wird gezielt angegeben, von wie vielen Fingern die spezifizierte Geste durchgeführt werden muss. Alternativ kann sie durch Parallelität definiert werden (vgl. Kap. 5.2.1), da diese implizit vorgibt, dass mehrere Werkzeuge gleichzeitig zur Ausführung der Geste eingesetzt werden.

5.1.2 Auflagefläche

Die Auflagefläche beschreibt die Größe der Berührung auf der Oberfläche und kann entsprechend nur für Oberflächengesten beschrieben werden – in räumlichen Gesten bewegen sich Werkzeuge typischerweise ohne auf einer Oberfläche aufzuliegen. Die Beschreibung ist jedoch oftmals direkt abhängig vom verwendeten Werkzeug (vgl. Kap. 5.1.13), da sich dadurch ganz unterschiedliche Optionen ergeben. Beispielsweise lassen sich bei Verwendung der menschlichen Hand u. a. die Fingerspitze und -kuppe sowie der Handballen unterscheiden. Buxton (2014) hält die Größe der Auflagefläche für ebenso wichtig wie die Größe der berührbaren Oberfläche und schon Cao et al. (2008) kritisierten vor einigen Jahren, dass die meisten Anwendungen und Frameworks die Metapher eines Cursors aus klassischer PnC-Interaktion übernehmen und so die reichhaltigen Möglichkeiten außer Acht lassen, die die menschliche Hand bietet. Die Autoren gehen insbesondere darauf ein, dass die Berührungsfläche auf einzelne Koordinaten heruntergerechnet wird und somit die Möglichkeiten ignoriert werden, die sich aus der Verwendung der Auflagefläche ergeben. Auch Saffer (2008) berücksichtigt die Auflagefläche in seinen Gesteneigenschaften, bringt aber als Beispiel die Erkennung unterschiedlicher Teile des Fingers – also unterschiedlicher Werkzeuge – ein. Verschiedene Autoren (z. B. Benko et al. (2006), Smith et al. (2007), Cao et al. (2008)) verwenden die gemessene Auflagefläche, um den auf die Oberfläche ausgeübten Druck zu bestimmen (vgl. Kap. 5.1.6).

Trump & Bullock (2014) demonstrieren, wie Unterschiede in der Auflagefläche sowie ihre Veränderung in einer Anwendung eingesetzt werden können. Ihre App *Orphion*, ein digitales Instrument für Tablets, erzeugt bei Berührung definierter Flächen Töne unterschiedlicher Höhe. Die Auflagefläche bestimmt dabei, welche Art von Tönen gespielt wird. Sowohl Benko et al. (2006)[6] als auch Wang & Ren (2009) beschreiben, wie sie ein Abrollen des Fingers von der Fingerspitze auf die Fingerkuppe als Interaktionstechnik einsetzen. Die Vergrößerung der Auflagefläche wird dann genutzt, um eine beliebige Aktion auszulösen. Boring et al. (2012) demonstrieren, wie mit der gleichen Bewegung ein Zoom auf einer Smartphone-App statt mit einer zweifingrigen Spread-Geste einfach über die veränderte Auflagefläche des Daumens durchgeführt werden kann. Nach den Autoren kann gerade auf Smartphones, die oftmals in einer Hand gehalten und mit dem Daumen ebendieser Hand bedient werden, die Interaktion mit dem Gerät so wesentlich verbessert werden. Mit der steigenden Verbreitung von Wearables, z. B. in Form von Smartwatches, die über einen wesentlich kleineren Bildschirm verfügen, wird eine solche Interaktionstechnik nochmal interessanter.

[6] Obschon die Autoren eigentlich ein anderes Ziel verfolgen, nämlich die Simulation gesteigerten Drucks auf die Oberfläche.

Wang & Ren (2009) analysieren außerdem, wie die unterschiedlichen Möglichkeiten, die die menschliche Hand bietet, für multi-touch-basierte Anwendungen verwendet werden können. Sie schließen, dass neben der reinen Positionierung einer Berührung auf der Oberfläche – was lediglich einen Cursor analog zu klassischen PnC-basierten Oberflächen darstellt – noch weitere Optionen bestehen. Die Autoren unterscheiden zwischen zwei Auflageformen: *vertical touch*, bei dem der Finger gerade auf die Oberfläche aufgesetzt wird und lediglich die Fingerspitze Kontakt hat, und *oblique touch*, bei dem der Finger schräg auf der Oberfläche liegt und somit die gesamte Fingerkuppe Kontakt hat. Dabei gehen sie insbesondere auf die Auflagefläche ein und identifizieren noch weitere, damit einhergehende Aspekte: die Form des Kontakts, die Ausrichtung der Auflagefläche sowie der ausgeübte Druck.

Aus der Ausrichtung der Auflagefläche kann auf die Orientierung des Werkzeugs geschlossen werden, die wiederum für verschiedene Anwendungsfälle genutzt werden kann. Dang & André (2011) liefern interessante Beispiele: Manipulation drehbarer Objekte, Identifikation von durch Hand oder Arm verdeckter Zonen, Auswahl von Objekten sowie genauere Bestimmung von Berührungspunkten. Allerdings ist nach den Autoren die reine Analyse der Auflagefläche nicht immer ausreichend, daher ziehen sie weitere Aspekte zu Rate, um die Orientierung zu erkennen.

Die Verwendung der Auflagefläche kann allerdings auch mit Problemen verbunden sein. Forlines et al. (2007) weisen darauf hin, dass sich die Auflagefläche vergrößert, wenn Benutzer sehr weit über die Oberfläche greifen müssen und ein gerades Aufsetzen des Fingers nicht mehr möglich ist. Dies kann z. B. bei Anwendungen auf Multi-Touch-Tischen der Fall sein, weshalb das Interaktionskonzept der Anwendung dann entsprechend überdacht werden muss. Des Weiteren diskutieren Wang & Ren (2009), ob die Auflagefläche nicht stark benutzerabhängig ist: Ein Kind mit wesentlich kleineren Fingern wird eine kleinere Auflagefläche haben als ein Erwachsener. Sie zeigen jedoch, dass das Verhältnis von Länge zu Breite der Auflagefläche bei allen Menschen in etwa gleich ist und somit von einem Erkenner entsprechend berechnet werden kann. Auch Dang & André (2011) diskutieren dieses potentielle Problem, lösen es jedoch implizit in ihrem entwickelten Erkenner. Dies ist allerdings ein reines Erkennerproblem und hat für die Spezifikation der Geste keine Auswirkung.

Die Analyse der wissenschaftlichen Literatur deutet darauf hin, dass die Auflagefläche insbesondere zur Identifikation anderer Eigenschaften verwendet wird: Der Druck auf die Oberfläche, das verwendete Werkzeug oder dessen Ausrichtung. Damit bleibt für die Spezifikation einer Geste insbesondere die Veränderung interessant, die Aktionen zur Laufzeit (vgl. Kap. 5.1.11) auslösen kann und u. a. von Boring et al. (2012) oder Trump & Bullock (2014) eingesetzt wird. Zu spezifizieren ist daher in erster Linie, ob sich die Auflagefläche vergrößert oder verkleinert.

5.1.3 Ausführender Benutzer

Cirelli & Nakamura (2014) kritisieren an bestehenden Notationen und Gestenerkennern u. a., dass sie nicht mehrbenutzerfähig sind, also nicht explizit Gesten definieren bzw. unterscheiden können, die von unterschiedlichen Benutzern getrennt oder gemeinsam ausgeführt werden. Sie präsentieren zwar im Rahmen einer weiteren Arbeit einen denkbaren Gestenerkenner, der Mehrbenutzergesten erkennen kann (Cirelli et al., 2015), diskutieren jedoch nicht die Auswirkungen auf die Spezifikation von Gesten, obwohl kooperative Gesten auch für ihre geforderten Kombo-Gesten interessant sind (vgl. Kap. 5.2.1).

Die Kritik ist berechtigt, denn mehrbenutzerfähige Sensoren existieren bereits seit mehreren Jahren (z. B. DiamondTouch (Dietz & Leigh, 2001), Carpus (Ramakers et al., 2012) oder Kinect (Microsoft Developer Network, 2016)). Auch aus anderen Eigenschaften lässt sich auf unterschiedliche Benutzer schließen, z. B. identifizieren Zhang et al. (2012) einzelne Nutzer anhand der Ausrichtung der Finger auf einer berührungsempfindlichen Oberfläche. Auch Buxton (2014) hebt die Bedeutung mehrbenutzerfähiger Sensoren für verschiedene Anwendungsfälle hervor, insbesondere in Umgebungen, bei denen mehrere Benutzer gemeinsam auf einer Oberfläche arbeiten können. Saffer (2008) nimmt die Anzahl der Benutzer und ihre Teilhabe an gemeinsamen Gesten ebenfalls in seine Menge denkbarer Gesteneigenschaften auf.

Wigdor & Wixon (2011, Kap. 7) sprechen in diesem Zusammenhang vom *Social Natural User Interface*, von dem es unterschiedliche Ausprägungen geben kann: Nutzer können gemeinsam am gleichen Ort zusammenarbeiten oder über mehrere Standorte verteilt sein. Darüber hinaus benennen sie mehrere Arten der Aufgabenkopplung,[7] durch die die einzelnen Teilaufgaben in Bezug zueinander gesetzt werden:

- Eine *hohe Kopplung* existiert, wenn mehrere Benutzer gemeinsam an derselben Teilaufgabe arbeiten (z. B. zwei Benutzer manipulieren zusammen dasselbe Objekt)
- Eine *schwache Kopplung* existiert, wenn Teilaufgaben zwar parallel ausgeführt werden, aber unabhängig voneinander gelöst werden können (z. B. zwei fachliche Experten arbeiten aus ihrer Perspektive am gemeinsamen Problem im Sinne des Prinzips *Divide & Conquer*)
- *Keine Kopplung* existiert, wenn Benutzer völlig unabhängig voneinander an unterschiedlichen Aufgaben auf dem gleichen UI arbeiten

Bei der Entwicklung einer Anwendung können die unterschiedlichen Arten der Aufgabenkopplung den Autoren nach vor allem die Auswahl der notwendigen Hardware beeinflussen, sodass sie explizit berücksichtigt werden müssen (Wigdor & Wixon, 2011, S. 40).

Morris et al. (2006) gehen dabei noch einen Schritt weiter und haben bereits vor einigen Jahren *kooperative Gesten* erforscht und gezeigt, dass diese sinnvoll einsetzbar sind und – je nach Anwendung – einen Mehrwert für Benutzer bringen, insbesondere für Anwendungen im Bereich Computer Supported Cooperative Work (CSCW). Sie definieren kooperative Gesten als Interaktio-

[7] Die Autoren sprechen in diesem Zusammenhang von *Inter-Task Coupling* (Wigdor & Wixon, 2011, S. 39).

nen, bei denen die Anwendung die Bewegungen mehrerer Benutzer als ein Kommando interpretiert. Sie benennen mehrere Anwendungsbereiche, bei denen die Integration kooperativer Gesten sinnvoll sein kann:

- *Förderung von Partizipation und Kooperation*: Neben den bereits angesprochenen Möglichkeiten im Bereich CSCW sehen Morris et al. (2006) vielversprechende Möglichkeiten für Bildungsanwendungen, insbesondere für jüngere Kinder. Außerdem betonen sie die Nützlichkeit für spezifische Nutzergruppen mit besonderen Anforderungen an soziale Interaktion und Kooperation, beispielsweise als Therapiewerkzeug für Menschen mit eingeschränkten sozialen Fähigkeiten.

- *Betonung wichtiger Handlungen*: Kooperative Gesten können nach Morris et al. (2006) eingesetzt werden, um wichtige Aktionen in Groupware-Anwendungen durchzuführen. Sie verweisen auf eigene Vorarbeiten, die zeigen, dass Handlungen mit weitreichenden Konsequenzen (z. B. viele Elemente löschen oder die Anwendung beenden) kritisch in Groupware-Anwendungen sind, wenn sie von Einzelnen aufgerufen werden können (Morris et al., 2004). Kooperative Gesten können helfen, den versehentlichen Aufruf von Funktionalitäten durch Einzelne zu verhindern.

- *Reichweite*: Insbesondere auf großen Oberflächen wie digitalen Whiteboards können Objekte außerhalb der Reichweite von Benutzern zu finden sein. Morris et al. (2006) schlagen daher vor, dass andere Benutzer Hilfestellung leisten können und ein Benutzer dann das Ziel der Aktion definiert (z. B. ein Objekt auf der Oberfläche) und ein anderer die Funktionalität aufruft.

- *Zugriffskontrolle*: In geteilten Oberflächen kann es sinnvoll sein, den Zugriff auf Objekte nur bestimmten Benutzern zu erlauben. Morris et al. (2006) schlagen vor, dass Besitzer von Objekten die Aktionen anderer Benutzer auf ihren Objekten durch explizite Gesten bestätigen müssen, andernfalls kann die Aktion nicht durchgeführt werden. Eine andere Variante ist eine grundsätzliche Sperre: Nur Gesten, die von einem bestimmten Benutzer ausgeführt werden, werden erfolgreich erkannt und münden im Aufruf der gewünschten Funktionalität.

- *Entertainment*: Synchrone Tätigkeiten sind seit jeher Teil menschlicher Unterhaltung[8] und kooperative Gesten ermöglichen die Entwicklung von Anwendungen, die dem sozialen Naturell entsprechen.

[8] Morris et al. (2006) führen als Beispiel La Ola im Stadion oder gemeinsame Tänze an.

In der wissenschaftlichen Literatur finden sich zahlreiche Beispiele, wie kooperative Gesten eingesetzt werden können. Viele Autoren und Forschungsprojekte haben dabei einen deutlichen Fokus auf therapeutische Zwecke. Piper et al. (2006) veranschaulichen, wie kooperative Gesten in der Behandlung des Asperger-Syndroms verwendet werden können. Battocchi et al. (2010) entwickelten ein gestenbasiertes Puzzle auf einem Multi-Touch-Tisch, das die Fähigkeiten zur Zusammenarbeit durch kooperative Gesten bei autistischen Kindern steigern soll. Giusti et al. (2011) leiten letztlich aus verschiedenen Anwendungen mit kooperativen Gesten Erkenntnisse für die Entwicklung

therapeutischer Software für Kinder mit Autismus ab. Favilla & Pedell (2014) setzen eine kooperative Anwendung zum gemeinsamen Musizieren auf mehreren Tablets ein, um Demenzkranke zu therapieren – ein Ansatz, der ebenfalls interessanten Spielraum für kooperative Gesten bietet. Der Faktor Entertainment steht bei Gutierrez et al. (2013) im Fokus. Sie berichten, wie sie ein Rollenspiel für Multi-Touch-Tische entwickelten, das u. a. kooperative Gesten einsetzt. Dabei beschreiben sie, inwiefern ein Papierprototyp sie bei der Entwicklung des Interaktionskonzepts unterstützt hat und welche Faktoren die Wahl der Gesten beeinflussten. Liu et al. (2017) hingegen adressieren die rein praktischen Probleme, die klassische Multi-Touch-Gesten auf übergroßen Bildschirmen verursachen, denn sie skalieren nicht über die Bildschirmgröße. Entsprechend sind neue Gesten notwendig, um effizientes Arbeiten auf übergroßen Bildschirmen zu ermöglichen. Die Autoren präsentieren kooperative Gesten, die verwendet werden können, damit unterschiedliche Benutzer virtuelle Objekte austauschen, präsentieren oder gruppieren können.

Morris et al. (2006) demonstrieren anhand einer beispielhaften Zeichenanwendung, wie kooperative Gesten eingesetzt werden können. In ihrer Anwendung *CollabDraw* können Benutzer gemeinsam Diagramme zeichnen oder Kollagen aus Bildern erstellen. Eine mögliche Geste in dieser Anwendung wird verwendet, um einzelne Elemente zu löschen. Dazu wird der Handballen auf dem jeweiligen Objekt schnell hin- und herbewegt – führen alle Benutzer gleichzeitig diese Geste aus, dann wird die Bedeutung der Gesten verstärkt und der gesamte Canvas gelehrt. Eine andere kooperative Geste verwenden die Autoren, um Bilder von einem Benutzer zum anderen zu übergeben. Dazu muss Benutzer A einen Swipe mit drei Fingern auf einem Bild ausführen und Benutzer B mit drei Fingern auf der gewünschten Stelle auf dem Multi-Touch-Tisch tippen. Wenn die Bewegungsrichtung des Swipes von Benutzer A in etwa der Position der Geste von Benutzer B entspricht, dann wird das Bild an diese Position gesetzt.

Aus ihren Erfahrungen mit CollabDraw leiten Morris et al. (2006) verschiedene Dimensionen ab, in denen kooperative Gesten gestaltet werden können. Diese können als Grundlage für die Spezifikation herangezogen werden:

- *Symmetrie*: Nutzer können die gleiche Geste (symmetrisch) oder unterschiedliche Gesten (asymmetrisch) Gesten ausführen. Denkbar sind auch Mischformen, wenn mehr als zwei Benutzer an der kooperativen Geste beteiligt sind. Im o. a. Beispiel ist die Geste zum Löschen des Canvas symmetrisch, jedoch die zum Verschieben von Bildern asymmetrisch. Spezifiziert man kooperative Gesten, so findet sich die Symmetrie in der Beschreibung von Teilgesten, aus denen die Gesamtgeste zusammengesetzt ist.

- *Parallelität*: Nutzer können kooperative Gesten in unterschiedlichen zeitlichen Bezügen ausführen, entweder gleichzeitig (parallel) oder nacheinander (seriell). Dies beschreibt die Beziehung zwischen Teilgesten, wie sie in Kap. 5.2.1 vorgestellt wird. Dabei sind parallele kooperative Gesten wie Multi-Touch-Gesten und serielle kooperative Gesten wie Multi-Stroke-Gesten zu betrachten.

- *Nähe*: Hiermit wird die räumliche Distanz zwischen Teilgesten der Benutzer beschrieben. Morris et al. (2006) definieren vier Kategorien der Nähe: *intimate*, bei der sich Benutzer körperlich berühren, z. B. an den Händen fassen; *personal*, bei der Benutzer das gleiche digitale Objekt berühren; *social*, bei der kooperative Gesten auf derselben Oberfläche ausgeführt werden; sowie *public*, bei der zwar kooperative Gesten ausgeführt, aber verschiedene Geräte verwendet werden, z. B. das eigene Smartphone. Mit Ausnahme der ersten Kategorie, die sich nicht auf die Ausführung der Geste selbst, sondern auf den Kontext bezieht, können diese Kategorien genauso wie der Ort der Ausführung (vgl. Kap. 5.1.10) spezifiziert werden.

- *Additivität*: Wenn mehrere Benutzer die gleiche Geste ausführen, dann verstärkt sich ihre semantische Bedeutung. Ein Beispiel ist die bereits angeführte Geste zum Löschen von Objekten: Wird diese von mehreren Benutzern ausgeführt, so wird der gesamte Canvas gelöscht. Bei der Spezifikation von Gesten äußert sich dieser Unterschied dahingehend, die gleiche Geste in Abhängigkeit der beteiligten Benutzer mehrfach definiert werden muss, da es sich durch die unterschiedliche semantische Bedeutung um verschiedene Gesten handelt.

- *Benutzeridentität*: Die Bedeutung bestimmter Gesten kann durch die Identität des jeweiligen Benutzers beeinflusst werden; beispielsweise können alle Gesten, die auf einem Objekt ausgeführt werden, ungültig sein, wenn der ausführende Benutzer nicht als Besitzer des Objekts in der Anwendung registriert ist. Für die Spezifikation von Gesten bedeutet dies insbesondere, dass Platzhalter vorgesehen werden müssen, damit von konkreten Benutzern abstrahiert werden kann.

- *Anzahl Benutzer/Geräte*: Gesten können auf eine feste Anzahl Benutzer ausgelegt sein, also aus einer bestimmten Menge Teilgesten bestehen, die von unterschiedlichen Benutzern ausgeführt werden. Gleichzeitig kann die Anzahl der Benutzer nicht immer exakt festgelegt werden – entsprechend sind Abstraktionen wie *alle Benutzer* oder eine bestimmte Anzahl notwendig. Die Anzahl verwendeter Geräte ist interessant, da es verschiedene, gestenbasierte Anwendungen gibt, die sich auf mehrere Geräte erstrecken (z. B. bei Kleffmann et al. (2015a)), lässt sich jedoch besser durch den Ort der Ausführung beschreiben (vgl. Kap. 5.1.10).

Eine wesentliche Schwierigkeit ist, dass zum Zeitpunkt der Spezifikation die genaue Anzahl der Benutzer u. U. nicht bekannt ist. Wenn sich die Teilgesten der einzelnen Benutzer unterscheiden, dann kann zwar wie bereits erwähnt auf die zeitliche Relation zwischen Teilgesten zurückgegriffen werden, gleichzeitig muss aber deutlich werden, dass es sich um unterschiedliche Benutzer handelt. Zu spezifizieren sind demnach die unterschiedlichen Arten, um Gesten mit mehreren Benutzern zu gestalten, die Morris et al. (2006) zusammenfassen. Diese sind jedoch teilweise bereits implizit durch andere Gesteneigenschaften abgedeckt. Daher muss die Spezifikation insbesondere auf unterschiedliche Benutzer eingehen können und ihnen explizit Teilgesten zuweisen.

5.1.4 Aus- und Bewegungsrichtung

Die Ausrichtung einer Geste und die Richtung der ausgeführten Bewegung hängen direkt miteinander zusammen, beschreiben jedoch unterschiedliche Dinge. Während die Richtung die Bewegungsrichtung eines Gestenzuges definiert – also die Richtung, in die sich z. B. ein Finger auf einer berührungsempfindlichen Oberfläche bewegt –, beschreibt die Ausrichtung, wie die gesamte Geste in Bezug zur Umgebung aufgestellt ist. Die Bewegungsrichtung kann sich also immer nur auf einen einzigen Gestenzug beziehen und sich im nächsten ändern, während sich die Ausrichtung auf die Gesamtgeste bezieht. Entsprechend bedingen sich Bewegungsrichtung von Teilgesten und Ausrichtung der Gesamtgeste.

Die Richtung, in die eine Geste ausgeführt wird, kann unterschiedliche Bedeutung haben. Bei einer Bildergalerie, die in Form eines Karussells dargestellt wird, kann z. B. durch Wischen nach links vorwärts und durch Wischen nach rechts rückwärts geblättert werden. Die Form der Geste bleibt gleich, aber durch die Richtung der Ausführung ändert sich die semantische Bedeutung.

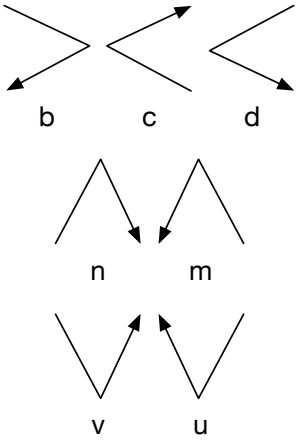


Abbildung 5.1: Einsatz einer Geste in Form eines V für verschiedene Buchstaben nach Goldberg & Richardson (1993) (Bildquelle: Eigene Darstellung nach ibd.)

Goldberg & Richardson (1993) zeigen ein weiteres Beispiel: ein gestenbasiertes Eingabesystem, bei dem das Alphabet zur schnellen Eingabe von Text durch einzelne Gesten dargestellt wird. Sie verwenden dazu aber nur fünf Grundformen, die in unterschiedlichen Ausprägungen eingesetzt werden. In der Folge müssen unterschiedliche Buchstaben zwangsläufig mit der gleichen Geste dargestellt werden. Goldberg & Richardson (1993) setzen dabei u. a. auf die Richtung der Ausführung, z. B. um die Buchstaben U und V voneinander zu unterscheiden. Beide werden mit einer Geste in Form eines V dargestellt, jedoch muss das U von rechts nach links und das V von links nach rechts ausgeführt werden (vgl. Abb. 5.1 unten).

Die Ausrichtung einer Geste kann auch als Erweiterung der Form verstanden werden. Viele Gestenerkenner – insbesondere Template-basierte Erkenner (vgl. Kap. 6.1) – ignorieren die Ausrichtung und erkennen jede gültige Form erfolgreich, ganz gleich wie sie bei der Ausführung ausgerichtet ist. So würde eine Geste in Form eines V erkannt werden, wenn das V auf dem Kopf steht oder nicht. Dieses Verhalten kann gewünscht sein, muss es aber natürlich nicht. Auch hierfür zeigen Goldberg & Richardson (1993) ein Beispiel, in dem die Ausrichtung der Geste eine Bedeutung hat. Die bereits erwähnte Geste in Form eines V wird für unterschiedliche Buchstaben verwendet und hat je nach Ausrichtung der Öffnung eine andere Bedeutung: nach oben V und U, nach unten M und N, nach rechts B und C und nach links D. Außerdem wird innerhalb der Buchstaben auch die Richtung unterschieden. Abb. 5.1 fasst die verschiedenen Möglichkeiten zusammen.

Darüber hinaus kann die Bewegungsrichtung auch abhängig von virtuellen Objekten sein, die erst zur Laufzeit der Anwendung platziert werden. Eine Geste kann sich z. B. von einem Punkt auf ein dynamisch platziertes, virtuelles Objekt zu- oder davon wegbewegen. Folglich kann zum Spezifikationszeitpunkt nicht immer genau definiert werden, in welche Richtung sich ein Gestenzug bewegt.

Allerdings stellen unterschiedliche Geräte auch verschiedene Anforderungen an die Flexibilität der Ausrichtung. Während bei mobilen Geräten die Definition der Ausrichtung aufgrund ihrer geringen Größe und des Fokus auf genau einen Benutzer leichtfällt (egal wie der Benutzer das Gerät hält, oben ergibt sich immer aus ebendieser Haltung), ist dies bei interaktiven Tischen, um die Benutzer herumgehen können, schon wesentlich schwieriger. Darüber hinaus können mehrere Benutzer um den Tisch stehen und die Anwendung bedienen. Kruger et al. (2003) unterscheiden dazu vier Strategien:

- Fixierte Ausrichtung (*Fixed Orientation*): Welche Seiten oben, unten, links und rechts sind, wird fix definiert. Benutzer müssen sich der Anwendung anpassen und entsprechend positionieren.

- Manuelle Ausrichtung (*Manual Orientation*): Überlässt Benutzern das Anpassen virtueller Objekte; sie müssen die Objekte in die passende Ausrichtung drehen und können dann wie gewohnt mit ihnen interagieren. Jeder Benutzer hat somit seine eigene Ausrichtung, auf die die Anwendung eingestellt werden muss.

- Personenbasierte automatische Ausrichtung (*Person-Based Automatic Orientation*): Virtuelle Objekte richten sich automatisch nach Benutzern aus, die mit dem Objekt gerade interagieren. Jeder Benutzer hat seine eigene Ausrichtung, an die sich die Anwendung selbstständig anpasst.

- Umgebungsbasierte automatische Ausrichtung (*Environment-Based Automatic Orientation*): Eine Erweiterung, die bei der automatischen Ausrichtung noch Aspekte der Umgebung hinzuzieht, um eine genauere Lokalisierung des Benutzers vorzunehmen.

Für die Spezifikation von Gesten sind diese Unterschiede insofern von Bedeutung, als dass sich die Bewegungsrichtung und Ausrichtung von Gesten je nach Positionierung der Benutzer um den Tisch herum ändern können. In der Folge müssen passende Strategien frühzeitig ausgewählt und definiert werden, damit Gesten mithilfe dieser Informationen korrekt spezifiziert werden.

Davon ausgehend, dass die Ausrichtung der Gesamtgeste implizit an der Bewegungsrichtung der einzelnen Teilgesten liegt, folgt, dass die Bewegungsrichtung einer Geste exakt spezifiziert werden muss. Gleichzeitig ist aber die Unabhängigkeit von der definierten Ausrichtung zu ermöglichen, denn nicht in allen Fällen soll die Geste bei der Ausführung so ausgerichtet sein wie spezifiziert. Darüberhinaus können Abhängigkeiten zu virtuellen Objekten oder anderen Teilgesten bestehen. Entsprechend muss eine gewisse Dynamik definiert werden können. Die Auswahl der o. g. Strategien hat insofern eine Bedeutung, dass dadurch festgelegt wird, ob Gesten aus Sicht der Anwendung, des Sensors oder des Benutzers spezifiziert werden.

5.1.5 Dauer

Eine Geste kann durch ihre Dauer von anderen Gesten mit gleichem Bewegungsmuster unterschieden werden. Ein klassisches Beispiel ist der *Long*

Press auf berührungsempfindlichen Bildschirmen: Eine längere Berührung eines UI-Elements sorgt für ein spezielles Verhalten, z. B. die Markierung für nachfolgende Aktionen. Dem gegenüber steht eine kurze Berührung, in der Regel ein kurzes Antippen. Im Falle von Oberflächengesten ist der Zeitraum zwischen Aufsetzen des Werkzeugs auf und Abnehmen des Werkzeugs von der Oberfläche als Dauer zu betrachten. Bei räumlichen Gesten ist die Unterscheidung weniger trivial, da eine explizite Segmentierung wie durch das Auf- und Absetzen auf berührungsempfindlichen Oberflächen nicht möglich ist. Letztlich handelt es sich hierbei aber um ein Problem, das aufseiten des Gestenerkenners gelöst werden muss.

Während die Dauer einer Geste bei Experimenten im Rahmen von Gestensteuerung durchaus erfasst wird (so z. B. bei Kray et al. (2010)), wird sie oftmals nicht explizit modelliert, sondern eher als globale Konstante innerhalb des Gestenerkenners angegeben (so z. B. bei Kammer et al. (2015)), um die bereits angesprochenen einfachen und längeren Berührungen zu unterscheiden. Oh & Findlater (2013) zeigen jedoch, dass Benutzer durchaus verschiedene Optionen in der Dauer einer Geste sehen, wenn sie selber Gesten definieren können. Auch in verschiedenen Anwendungen spielt die Dauer der Geste eine Rolle und geht über die einfache Unterscheidung in kurze und lange Berührungen hinaus – so beeinflusst die Dauer der Berührung etwa bei Trump & Bullock (2014) den durch das touch-basierte Instrument erzeugten Klang. Ghomi et al. (2012) verwenden Muster aus unterschiedlich lang dauernden, einfachen Berührungen, um rhythmische Gesten zu definieren. Dorau (2011, S. 128) fügt dem noch das Tempo hinzu, mit dem eine rhythmische Geste ausgeführt wird.

Neben der Dauer der Geste oder Teilgeste als solche – also dem Zeitraum der Ausführung – kann aber auch die Dauer zwischen zwei Teilgesten für die Anwendung von Bedeutung sein (Cirelli & Nakamura, 2014) – die Zeit also, in der Benutzer zwischen Gestenzügen pausieren. Im Falle von Gesten auf berührungsempfindlichen Oberflächen lässt sich der Zeitraum zwischen Teilgesten recht einfach bestimmen: Benutzer berühren in dieser Zeit die Oberfläche nicht. Bei räumlichen Gesten ist diese Unterscheidung nicht mehr ganz so einfach, da die Sensoren die Benutzer typischerweise die ganze Zeit erfassen und somit jede Bewegung aufnehmen. Eine passende Segmentierung in relevante Bewegungen und Pausen muss explizit aufseiten des Gestenerkenners implementiert werden. Die Erfassung der Dauer zwischen Gestenzügen kann aber verwendet werden, um unterschiedliche zeitliche Relationen (vgl. Kap. 5.2.1) genauer zu definieren und explizit spezifizieren zu können. Zu spezifizieren ist demnach neben einem Zeitraum für die Ausführung der Geste u. U. auch ein Zeitraum für eine Pause zwischen zwei Teilgesten. Außerdem können parallel ausgeführte Teilgesten früher oder später starten und früher oder später enden als andere und sich somit in der Dauer unterscheiden.

5.1.6 Druck

Verschiedene berührungsempfindliche Oberflächen erlauben die Messung des Drucks, den der Benutzer auf sie ausübt. Da mit der Möglichkeit, physischen

Druck auszuüben, in der Regel eine Berührung einhergeht, ist diese Eigenschaft vorwiegend der Definition von Oberflächengesten vorbehalten. Bei räumlichen Gesten kann diese Eigenschaft zum Einsatz kommen, wenn diese mit einem Werkzeug ausgeführt werden, auf das Druck ausgeübt werden kann (z. B. ein mobiles Gerät, vgl. Kap. 5.1.13).

Auch Saffer (2008) nimmt den Druck in die Menge der Gesteneigenschaften auf. Cao et al. (2008)[9] zeigen verschiedene Beispiele, welche Funktionalitäten mit der Auswertung des Drucks umgesetzt werden können und insbesondere wie bestehende UI-Widgets (z. B. Buttons) durch die Integration verschiedener Stärken von Druck aufgewertet werden können. Sie unterscheiden zwei Arten von Druck: Druck ohne Bewegung (*Pressing*) und Druck mit Bewegung, wobei letzterer nochmal unterteilt wird in Zusammenstoß (wenn die Geste außerhalb eines virtuellen Objekts startet und dieses berührt (*Colliding*)) und Reibung (wenn die Geste ausschließlich auf einem virtuellen Objekt ausgeführt wird (*Friction*)). Heo & Lee (2011) demonstrieren anhand zweier Beispielapplikationen verschiedene Möglichkeiten, wie Standardgesten durch Druck erweitert und für genauer definierte Funktionen verwendet werden können. Allerdings zeigen sie auch, dass die intensive Integration von Druck schnell zu Ermüdungserscheinungen bei Benutzern führen kann.

[9] Die Autoren befassen sich in erster Linie mit der Auflagefläche, leiten jedoch daraus Informationen über den Druck ab.

Wie viel Druck Nutzer unterscheiden können, wird von verschiedenen Studien allerdings unterschiedlich beantwortet. Bei Rendl et al. (2014) konnten Nutzer nur zwischen zwei Stufen gut unterscheiden. Sie verwendeten unterschiedliche Multi-Touch-Gesten auf einem Trackpad, müssen aber einschränken, dass ihre Versuchspersonen kein Feedback über den Druck bekamen, sodass die eigentliche Ursache nicht eindeutig identifiziert werden kann. Ramos et al. (2004) zeigen hingegen, dass Nutzer sechs unterschiedliche Stufen von Druck erfolgreich einsetzen können. Ihre Ergebnisse basieren allerdings auf der Verwendung eines Stylus, mit dem Nutzer auch viel exakter arbeiten können (vgl. Kap. 5.1.13).

Buxton (2014) verweist auf Herot & Weinzapfel (1978), die bereits vor vielen Jahren einen passenden Sensor vorstellten. Auch gängige mobile Betriebssysteme ermitteln heute den Druck, der auf den Bildschirm ausgeübt wird, und übermitteln ihn an die Anwendung. Der gemessene Druck wird typischerweise genormt verfügbar gemacht und in einen festen Wertebereich übersetzt. iOS verwendet hierzu einen Wert von 1.0 für einen *üblichen* Druck[10] und kleinere bzw. größere Werte für einen geringeren bzw. größeren Druck (Apple Developer, oJ[b]). Android hingegen gibt einen Wert zwischen 0 und 1 zurück, wobei 0 kein Druck und 1 sehr starker Druck auf die Oberfläche bedeutet (Android Developers, oJ[c]). Das Protokoll TUIO definiert in der Version 2.0 einen Wert von 0 bis 1 für unterschiedlich starken Druck und einen negativen Wert für keinen Druck (Kaltenbrunner, 2014). Die Möglichkeit, keinen Druck auf die Oberfläche auszuüben, aber dennoch eine Multi-Touch-Geste auszuführen, klingt zunächst einmal paradox, ist aber mit verschiedenen Geräten möglich (z. B. auf dem Samsung Galaxy S4 (Samsung Tech Life, oJ)). Die Werkzeuge schweben dann knapp über der Oberfläche, ohne sie zu berühren. Wobbrock et al. (2009) beschreiben, dass manche Teilnehmer in ihren

[10] Allerdings bleibt unklar, wie dieser Wert ermittelt wurde und was *üblich* in diesem Kontext bedeutet.

Experimenten Gesten definierten, bei denen die Finger die Oberfläche nicht berührten. Hinckley et al. (2016) zeigen weitere Möglichkeiten.

Zahlreiche Autoren demonstrieren, wie der Druck auch ohne passenden Sensor gemessen werden kann. Wigdor & Wixon (2011) sprechen dann von *Pressure Proxies*, die eingesetzt werden können, um den Druck mithilfe anderer Informationen zu definieren. Gängig ist die Ableitung aus der Auflagefläche des Werkzeugs (z. B. bei Benko et al. (2006), Smith et al. (2007) und Cao et al. (2008)), aber verschiedene Forschergruppen haben auch Alternativen entwickelt. Hwang et al. (2013) demonstrieren, wie der Vibrationsalarm eines Mobilgeräts verwendet werden kann. Drückt der Benutzer auf das Gerät, so absorbiert der Finger Teile der Vibration, die die Autoren messen. Interessanter Nebeneffekt dieses Vorgehens: Druck kann auf dem gesamten Gerät gemessen werden (z. B. durch Zusammendrücken der Hand, in der das Gerät gehalten wird), nicht nur auf der Oberfläche. Dadurch ergeben sich Möglichkeiten, unterschiedlichen Druck in räumliche Gesten zu integrieren, die mit dem Gerät ausgeführt werden. Low et al. (2014) erreichen die gleichen Ergebnisse, wenden jedoch eine andere Strategie an. Sie messen mithilfe der Kamera die Lücke zwischen Mobilgerät und Hand, in der es gehalten wird. Übt der Nutzer Druck auf das Gerät aus, so verringert sich diese Lücke.

Wie der Druck erkannt wird, ist letztlich zwar ein Implementierungsproblem des Erkenners, hat aber Auswirkungen auf die Spezifikation, da andere Eigenschaften u. U. nur eingeschränkt verwendet werden können. Bei vielen Ansätzen aus der wissenschaftlichen Literatur kann z. B. eine Unterscheidung von Fingerspitze und ganzem Finger nur mit dem Nebeneffekt integriert werden, dass mehr Druck erkannt wird. Gleichzeitig zeigen sich hier aber auch die Vorteile einer Spezifikation, denn die notwendigen Sensoren können von ihr abgeleitet und derartige Probleme aufgedeckt werden. Zu spezifizieren sind demnach unterschiedliche Stärken des ausgeübten Drucks.

5.1.7 Form

Die Form einer Geste beschreibt ihre geometrische Struktur. Dabei kann die gleiche Form auch durch unterschiedliche Kombinationen von Gestenzügen erreichen werden: Die Form einer Geste, die wie ein V ausgeführt wird, bleibt gleich, egal welcher der beiden Schenkel zuerst gezeichnet wird. Inwiefern diese Unterschiede für die Anwendung relevant sind, müssen Entwickler anhand des konkreten Anwendungsfalls entscheiden. Interessanterweise gehen weder Saffer (2008) noch Buxton (2014) explizit auf die Form einer Geste ein, obwohl es sich um eine für die Erkennung wesentliche Eigenschaft handelt. Anhand der Form werden letztlich Gesten direkt voneinander unterschieden und viele Erkenner können nahezu ausschließlich diese Eigenschaft einer Geste erkennen (vgl. Kap. 6.1).

Die Freiheiten, die Gestensteuerung zulässt, erlauben alle denkbaren geometrischen Formen, wodurch insbesondere textuelle Ansätze zur Beschreibung von Gesten an ihre Grenzen kommen (vgl. Kap. 6.2), während Ansätze, die auf vorgezeichneten Mustern basieren und somit graphische Elemente ins Spiel bringen, ihre Stärken ausspielen können. Dass gerade die Form einer

Geste insbesondere für Benutzer relevant ist, zeigen Appert & Zhai (2009). Sie demonstrieren, dass Benutzer sich Gesten anhand ihrer Form leichter merken und sie effizienter verwenden können als klassische Tastaturkürzel.

Für die Spezifikation von Gesten ist somit die exakte Definition geometrischer Formen relevant. Um die mögliche Menge denkbarer Gesten für die Anwendung nicht einzuschränken, sollten keine Restriktionen definiert werden.

5.1.8 Geschwindigkeit

Gesten lassen sich in unterschiedlicher Geschwindigkeit ausführen. Saffer (2008) beschreibt die Geschwindigkeit der Ausführung auch als Eigenschaft, allerdings indirekt als Teil der Dauer einer Geste. Er berücksichtigt damit aber nicht, dass das Verhältnis von Dauer zu zurückgelegter Strecke das wesentliche Merkmal der Geschwindigkeit ist.

Durch Integration unterschiedlicher Geschwindigkeit lassen sich z. B. langsam ausgeführte, linienförmige Gesten von schnellen *Swipes*[11] unterscheiden. Swipes sind elementare Bestandteile der meisten Multi-Touch-Frameworks und werden von mobilen Betriebssystemen wie Android oder iOS (vgl. Kap. 4.2.1) durch vorimplementierte Komponenten verfügbar gemacht. Sie gehören daher zu den grundlegenden Gesten, die in zahlreichen Anwendungen zu finden sind. Die detaillierte Beschreibung der Geschwindigkeit – insbesondere für Teilgesten – erlaubt eine wesentlich genauere Beschreibung verschiedener Gesten und erweitert somit die Möglichkeiten gestenbasierter Interaktion. Die Geschwindigkeit einer Geste kann sich sowohl auf die gesamte Geste als auch auf einzelne Teilgesten beziehen.

[11] Die Begriffe Swipe, Flick und Fling werden im Rahmen dieser Arbeit synonym für schnelle Wischbewegungen verwendet. Bei gängigen Frameworks kann die Bedeutung abweichen (vgl. Kap. 4.2.1).

Allerdings kann die exakte Messung der Geschwindigkeit durch technische Details erschwert werden. Zum Beispiel wird unter Android die Geschwindigkeit eines Swipes über eine vorgefertigte Komponente erfasst. Die Methode `onFling`, die vom Interface `GestureDetector.OnGestureListener` vorgegeben wird, gibt die Geschwindigkeit der Bewegung auf der X- und Y-Achse in Pixel pro Sekunde zurück (Android Developers, oJ[f]). Auf Geräten mit unterschiedlichen Pixeldichten führt dies jedoch zu verschiedenen Ergebnissen. Eine Normalisierung ist daher zwingend notwendig, um verlässliche Ergebnisse auf unterschiedlichen Geräten zu bekommen.

Zu Beachten ist jedoch, dass es für Benutzer schwierig ist, bei der Ausführung von Gesten die Geschwindigkeit konstant zu halten. Vatavu et al. (2014) zeigen, dass sich die Geschwindigkeit der Ausführung je nach Form ändern kann. Auf Touchscreens werden komplexe Formen langsamer ausgeführt als einfache. Auch Kreise, die auf Linien folgen, und Richtungswechsel lassen Benutzer in der Regel langsamer werden. Die Unterschiede in der Gestenausführung bei Verwendung unterschiedlicher Werkzeuge (vgl. Kap. 5.1.13) legen den Schluss nahe, dass sich die Schwankungen in der Geschwindigkeit bei anderen Werkzeugen auch anders auswirken. Die Autoren zeigen außerdem, dass es eher zu Fehlerkennungen von Formen kommt, wenn Gesten sehr schnell ausgeführt werden. Entsprechend sollten für schnell ausgeführte Gesten eher einfache Formen gewählt werden.

Für die Spezifikation von Gesten folgt, dass eine genaue Angabe der Geschwindigkeit nicht sinnvoll ist, da nicht nur technische Details zu unterschiedlichen Ergebnissen führen, sondern auch Unterschiede in der Gestenausführung zwischen verschiedenen geometrischen Formen und unterschiedlichen Werkzeugen deutlich sind. Gleichzeitig ist davon auszugehen, dass darüberhinaus noch Unterschiede zwischen Benutzergruppen und Varianzen je nach Einsatzkontext der Anwendung bestehen. Folglich sollte bei der Spezifikation von Gesten eher mit Intervallen oder Schlüsselwörtern für verschiedene, normalisierte Geschwindigkeiten gearbeitet werden.

5.1.9 Multiplizität

Neben der exakten Ausführung der Geste kann auch die Anzahl der Ausführungen für die Anwendung relevant sein. Teilgesten können mehrfach wiederholt werden und so unterschiedliche Auswirkungen haben. Ein einfaches Beispiel ist der *Double-Touch*, der dem klassischen Doppelklick mit einer Maus nachempfunden ist. Benutzer tippen dabei zweimal hintereinander auf z. B. ein virtuelles Objekt auf dem berührungsempfindlichen Bildschirm. Denkbar sind hier auch Erweiterungen wie Triple-Touches, bei denen dreimal auf den Bildschirm getippt wird. Saffer (2008) hingegen nimmt die Multiplizität nicht in seine Menge der Gesteneigenschaften auf.

Letztlich besteht eine derartige Geste aber aus einer bestimmten Anzahl von Wiederholungen der gleichen Teilgeste. Und eben diese muss im Rahmen der Spezifikation berücksichtigt werden: wie oft eine Geste oder Teilgeste wiederholt werden kann. Denkbar sind dabei die Festlegung auf genau eine Ausführung, eine bestimmte Anzahl oder beliebig viele Wiederholungen.

5.1.10 Ort

Der Ort einer Geste beschreibt den Gültigkeitsbereich während der Ausführung im Verhältnis zu einem bestimmten Bezugspunkt. Dies ist nicht zu verwechseln mit der räumlichen Beziehung, die das Verhältnis von Teilgesten zueinander beschreibt (vgl. Kap. 5.2.2). Dennoch ist eine Beziehung zu auf dem UI dargestellten Objekten der Anwendung denkbar. Dies bedeutet auch, dass der Ort der Geste zur Laufzeit der Anwendung festgelegt werden kann. Buxton (2014) stellt heraus, dass der Ort der Ausführung – oder eher, die Genauigkeit, mit der Benutzer diesen Ort treffen müssen – wesentliche Bedeutung für die Interaktion hat. Auch Zhai et al. (2012) kommen zu diesem Schluss und stellen verschiedene Vorteile vor, die mit der Berücksichtigung des Ortes der Ausführung einhergehen: Kopplung von Aktion und Objekt sowie einfachere Identifikation der Nutzerintention. Gleichzeitig beschreiben die Autoren, dass Ortsunabhängigkeit insbesondere Platz spart – ein Vorteil bei kleinen Bildschirmen, wie z. B. auf Smartphones oder erst recht auf Smartwatches. Auch Saffer (2008) integriert den Ort der Ausführung in seine Menge von Gesteneigenschaften und schlägt zudem vor, ihn z. B. zur Zugriffskontrolle einzusetzen. Wigdor & Wixon (2011) schlagen wiederum vor, dass Benutzer die Möglichkeit haben sollten, bestimmte Regionen für bestimmte

Aufgaben dynamisch zu definieren; hieraus ergibt sich zwangsläufig, dass sich der Gültigkeitsbereich für bestimmte Interaktionen ändern kann.

Als Bezugspunkt können unterschiedliche Dinge gewählt werden, jeweils in Abhängigkeit der Anwendung, der eingesetzten Sensoren und der Anforderungen an den jeweiligen Anwendungsfall. Bei Oberflächengesten ist dies die Oberfläche selbst und die Geste wird auf einen Teilbereich der Oberfläche beschränkt. Ein einfaches Beispiel ist die Begrenzung der Geste auf den oberen Teilbereich, wie im linken Teil der Abb. 5.2 gezeigt. Im rechten Beispiel wird der Ausführungsort der Geste durch ein Objekt bestimmt, auf dem der Benutzer die Geste ausführen muss. Dabei sollte es Entwicklern freistehen, einen gewissen Rahmen um das Objekt herum zu definieren, um der Ungenauigkeit, mit der Gesten in der Regel ausgeführt werden, Rechnung zu tragen. Beiden Möglichkeiten gemein ist, dass sie den Bereich einschränken, in dem die Geste ausgeführt werden soll.

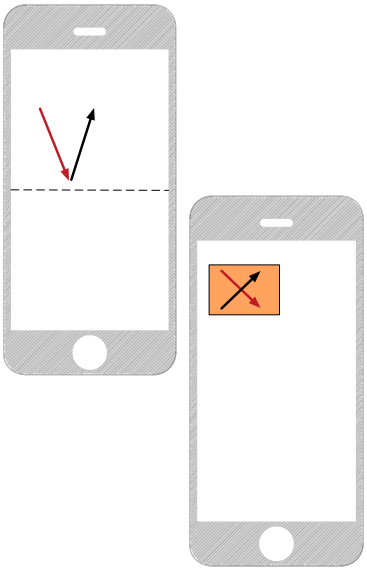


Abbildung 5.2: Zwei Beispiele zur Begrenzung des Ortes. Zum einen auf die obere Hälfte des Bildschirms: Die Geste wird nur erkannt, wenn der Benutzer die angegebene Grenze während der Ausführung nicht überschreitet. Zum anderen auf ein bestimmtes Objekt: Die Geste muss innerhalb der durch das Objekt bestimmten Grenzen ausgeführt werden.

Eine einfache Möglichkeit ist die Unterteilung des Bildschirms in Planquadrate. Die grundlegende Variante ist dabei die einfache Teilung in oben und unten (oder links und rechts), die festlegt, dass eine Geste nur im oberen oder unteren Teil des Bildschirms erkannt werden soll. Diese Aufteilung kann beliebig verfeinert werden, um weitere Granularitäten zu ermöglichen – Abbildung 5.3 zeigt verschiedene Optionen. Dabei ist die Wahl der jeweiligen Granularität stark abhängig vom verwendeten Gerät bzw. Sensor. Für große digitale Whiteboards sind andere Granularitäten denkbar als für kleine Smartphones. Des Weiteren kann die Benennung der einzelnen Felder abhängig vom Sensor bzw. der verwendeten Oberfläche sein. Während bei Smartphones und digitalen Whiteboards unabhängig von ihrer Größe einfache Positionsangaben wie oben, unten, links und rechts denkbar sind, so ist dies bei Multi-Touch-Tischen, bei denen Benutzer um den Tisch herum stehen können, schon mit einer genauen Definition der jeweiligen Positionen verbunden. Der Entwickler kann definieren, dass *oben* beispielsweise eine bestimmte

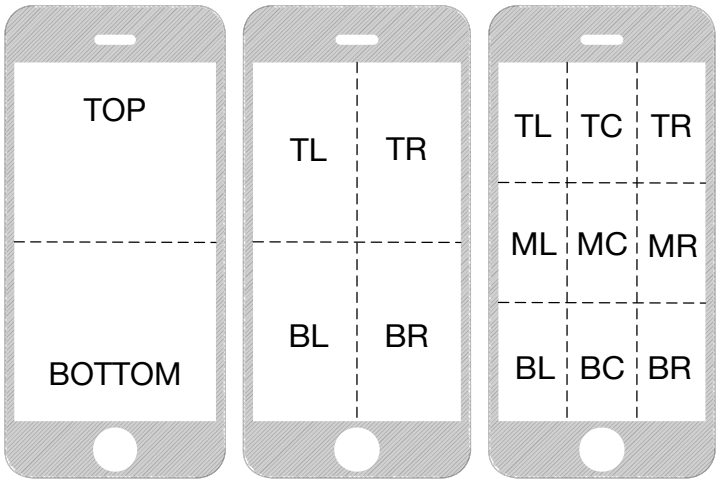


Abbildung 5.3: Verschiedene Möglichkeiten zur Aufteilung des Bildschirms in Planquadrate am Beispiel eines Smartphones

Richtung ist, die sich aber vom tatsächlichen Empfinden der Benutzer unterscheiden kann. Entsprechend ist es für Entwickler interessant, die Benennung

der Planquadrate schon in der Gestenbeschreibung den tatsächlichen Begebenheiten der Anwendung anzupassen. Dies kann aber nur mit passender Modularität der Gestenbeschreibung und des zugehörigen Erkenners erreicht werden.

Die Verwendung von Planquadraten zur Bestimmung des Ausführungsortes hat einige Vorteile. Die verschiedenen Granularitäten erlauben eine gewisse Dynamik, mit der die Genauigkeit der Beschreibung an den jeweiligen Anwendungsfall angepasst werden kann. Gleichzeitig ist diese Definition einfach zu verstehen und kann vermutlich auch von Stakeholdern ohne technische Kenntnisse einfach durchdrungen und selber eingesetzt werden. Des Weiteren können Planquadrate einfach zusammengefasst und von-bis-Relationen gebildet werden. Wird beispielsweise eine Granularität mit neun Feldern verwendet, dann lässt sich eine Geste einfach auf eine beliebige Zahl verbundener Felder verteilen, z. B. von oben Mitte über oben rechts nach Mitte rechts, wie in Abb. 5.4 dargestellt.

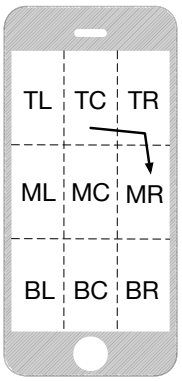


Abbildung 5.4: Beispiel einer Geste verteilt auf mehrere Planquadrate

Ein einfaches Beispiel für eine Anwendung, die die Aufteilung des Bildschirms in zwei Planquadrate mit unterschiedlicher Funktion nutzt, beschreiben Nancel et al. (2015). Die Autoren experimentieren mit unterschiedlichen Optionen zum Zugriff auf Informationen auf hochauflösenden, sehr großen Bildschirmen und setzen dazu u. a. ein Smartphone ein, dessen Oberfläche in zwei Segmente aufgeteilt wird: das obere, etwas größere, zur exakten Steuerung; das untere, etwas kleinere, zur Auswahl. Die einfache Aufteilung der Berührungsfläche soll Anwender auf zwei unterschiedliche Arten unterstützen: Zum einen ist das UI so einfach gestaltet, dass sie sich auf die eigentliche Aufgabe – die Arbeit auf dem übergroßen Bildschirm – konzentrieren und nicht auf das Steuergerät schauen müssen, zum anderen können sie je nach berührter Fläche eher feingranulare oder grobgranulare Arbeiten durchführen. Auch Choi et al. (2014) verwenden ein Raster aus Planquadraten, um den Ort, an dem sich die Hand bei der Ausführung der Geste befindet, zu beschreiben. Die Autoren orientieren sich dabei an den Arbeiten von McNeill (1992) und legen neun Planquadrate über einen sitzenden Menschen, um den Ort der Ausführung einer Geste definieren zu können, wie in Abb. 5.5 dargestellt.

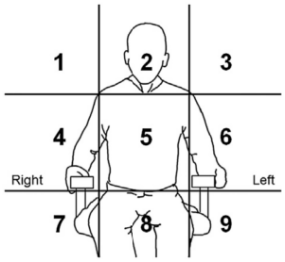


Abbildung 5.5: Planquadrate für räumliche Gesten nach McNeill (1992) (Bildquelle: Choi et al. (2014))

Bei aller Einfachheit sind Planquadrate jedoch auch eben durch diese begrenzt. Sie geben ein festes Raster vor, aus dem Entwickler nur bedingt bis gar nicht ausbrechen können. Zwar sind Kombination und von-bis-Beziehungen denkbar, um Planquadrate zu kombinieren und somit den Spielraum zu vergrößern, jedoch ist eine direkte Abhängigkeit von Objekten nur schwer darstellbar, auch wenn Szenarien wie *alle Planquadrate über dem, in dem sich das Objekt befindet* denkbar sind.

Wobbrock et al. (2009) zeigen außerdem, dass neben dem Bildschirm selbst auch ein Gebiet *Off Screen*, also außerhalb des Bildschirms, existiert, das für Gesten interessant ist. Benutzer können dieses für verschiedene Anwendungsfälle auch bewusst einsetzen und den verschiedenen Seiten des Bildschirms unterschiedliche Bedeutung geben: So könnten virtuelle Objekte nach links aus dem sichtbaren Bildschirm geschoben werden, um diese für später auf eine Ablage zu legen, die durch eine Geste, die links außerhalb des Bildschirms startet, wieder betrachtet werden kann. Abb. 5.6 veranschaulicht

dieses Beispiel. Ein ähnliches Konzept wird heute bereits auf mobilen Geräten verwendet, z. B. in Form von *Edge Swipes*, bei denen eine Geste am Rand des Bildschirms begonnen wird. Sowohl unter Android (Android Developers, oJ[b]) als auch unter iOS (Apple Developer, oJ[a]) kann beispielsweise explizit abgefragt werden, ob der Bildschirm nahe dem Rand berührt wurde und somit das Gebiet Off Screen simuliert werden. Serrano et al. (2013) verwenden einfache, kurze Interaktionen, um Anwendungen auf schlafenden mobilen Geräten direkt zu starten, ohne Umweg über die sonst notwendigen Einzelschritte (Gerät aufwecken, entsperren usw.). Diese werden über eine kurze Berührung des Geräterahmens initiiert – dem namensgebenden *Bezel-Tap* – und durch eine nachfolgende Geste auf dem Bildschirm komplettiert, die zur Identifikation der Anwendung dient.

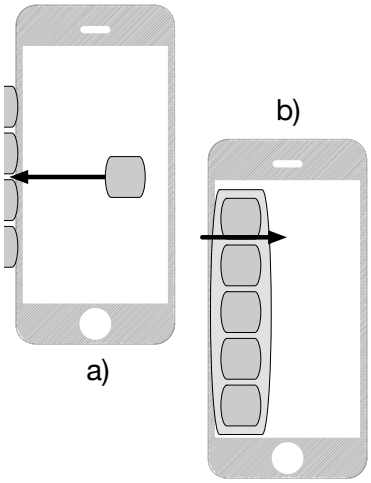


Abbildung 5.6: Gesten, die ein Gebiet außerhalb des Bildschirms einbeziehen: a) ein virtuelles Objekt nach links in eine Ablage schieben; b) die Ablage wieder sichtbar machen

Vatavu (2013) erweitert den Fernsehbildschirm und platziert zusätzliche Informationen um den Fernseher herum. Die Interaktion – z. B. der Austausch des aktuellen Bildschirminhalts – erfolgt dann auch über Gesten, die außerhalb des Bildschirms beginnen. Matsumura (2015) studiert die Möglichkeiten, die Gesten außerhalb des Bildschirms für mobile Anwendungen bieten. Er beschreibt verschiedene Anwendungsfälle und lässt seine Probanden konkrete Gesten definieren. Dabei deckt er auch Kombinationen von Gesten auf und außerhalb des Bildschirms auf. Auch Shimon et al. (2015) untersuchen alternative Ausführungsorte für Gesten auf mobilen Geräten, fokussieren aber die Rückseite von Smartphones. Sie erweitern dadurch die Möglichkeiten, die Gestensteuerung auf Smartphones bietet: Statt aufgrund der typischen Handhaltung das Gerät nur mit dem Daumen bedienen zu können, können Benutzer ohne Änderung der Handhaltung auch andere Finger einsetzen. Gleichzeitig kann die gleiche Geste auf Vorder- oder Rückseite unterschiedliche Bedeutung für die Anwendung haben. Das Yotaphone 2 wird auch bereits mit zwei unterschiedlichen Bildschirmen auf der Vorder- und Rückseite ausgeliefert und kann entsprechend beidseitig verwendet werden (Yota Devices, 2016). Zhang et al. (2016) demonstrieren, wie Multi-Touch-Gesten von einer Smartwatch erkannt werden können, die links und rechts neben der Uhr auf dem Arm ausgeführt werden. Sie erweitern dadurch die eigentlich recht kleine Interaktionsfläche der Uhr und ermöglichen eine größere Menge von Gesten; insbesondere solcher, die mehr Platz benötigen. Gleichzeitig ist eine explizite Positionierung der Gesten um die Smartwatch herum möglich, wodurch gleiche Gesten auch unterschiedliche Bedeutungen haben können.

Einige der vorgestellten Beispiele zeigen, dass nicht nur der Ausführungsort der gesamten Geste interessant ist, sondern dem Start- oder Endpunkt eine besondere Bedeutung zukommt. Des Weiteren zeigen die Beispiele, dass dieser nicht zwingend auf dem Gerät liegen muss, sondern eine Art Grauzone um die sichtbare Interaktionsfläche herum existiert, die zur Ausführung von Gesten herangezogen werden kann. Eine andere typische Aktion, die die Relevanz des Start- und Endpunkts aufzeigt, ist das Verschieben eines Objektes mit dem Finger. Die Geste beginnt dabei zwingend auf dem jeweiligen Objekt, kann aber an beliebiger Stelle auf dem Bildschirm enden. Letztlich ist aber nicht der Pfad relevant, der beim Verschieben auf dem Bildschirm zurückgelegt wird, sondern explizit der Start- und Endpunkt.

In den betrachteten Beispielen wird der Ort immer in Bezug auf den verwendeten Sensor bzw. das verwendete Gerät definiert. Scott et al. (2004) zeigen jedoch, dass bei kooperativen Anwendungen – also Anwendungen, bei denen sich mehrere Benutzer ein Gerät, z. B. einen interaktiven Tisch, teilen – jeder Benutzer ein eigenes Gebiet definiert, das für bestimmte Aufgaben benutzt wird. So unterscheiden die Autoren *persönliche* Orte sowie *Gruppen-* und *Speicherorte*. Persönliche Orte werden verwendet, um z. B. eigene Ideen zu entwickeln, bevor diese in die gemeinsamen Bereiche übertragen werden, und befinden sich oft direkt vor dem Benutzer. Gruppenorte sind die Bereiche der Anwendung, in denen gemeinsame Aktivitäten stattfinden, und typischerweise eher im Zentrum angesiedelt. Speicherorte wurden von den beobachteten Benutzern hingegen verwendet, um virtuelle Objekte abzulegen. Die unterschiedliche Verwendung der Bereiche für verschiedene Funktionalitäten ist auch für Gestensteuerung interessant – abhängig vom Benutzer können gleiche Gesten an unterschiedlichen Orten verschiedene Bedeutung haben. Cirelli & Nakamura (2014) fordern entsprechend auch von Frameworks für Gestensteuerung, dass der Ort der Ausführung explizit definiert werden kann.

Bei räumlichen Gesten wird das Problem des Bezugspunktes offensichtlicher. Eine Geste kann in räumlicher Nähe zum Sensor ausgeführt werden, was insbesondere für intelligente Umgebungen interessant ist. Gleichzeitig kann die Position zum Sensor irrelevant sein, aber der Körper des Benutzers selbst kann als Bezugspunkt dienen. Es kann dann unterschieden werden, ob eine Geste beispielsweise ober- oder unterhalb eines bestimmten Punktes durchgeführt wird. Durch unterschiedliche Körpergrößen stellen benutzerspezifische Besonderheiten erweiterte Anforderungen an Gestenerkenner.

Zu spezifizieren sind Gültigkeitsbereiche, in denen Gesten ausgeführt werden. Start- und Endpunkt einer Geste sind – wie gezeigt – Verfeinerungen des Ausführungsortes. Gleiche Gesten können an unterschiedlichen Orten starten bzw. enden und so verschiedene Bedeutungen für die Anwendung haben. Entsprechend müssen Start- und Endpunkt ggf. explizit spezifiziert werden.

5.1.11 Reaktion

Die Reaktion auf eine Geste kann auf zwei Weisen erfolgen: entweder während der Ausführung oder nach Abschluss. Ein einfaches Beispiel für eine Multi-Touch-Geste, bei der Anwendungen oft zur Ausführungszeit reagieren, ist die vielfach zum Zoomen oder Vergrößern von Objekten verwendete Spread-Geste. Ein Beispiel für eine Geste, die erst vollständig ausgeführt werden muss, ehe die Anwendung reagieren kann, ist das Zeichnen eines X in zwei Zügen auf einem Objekt, um dieses zu löschen. Dieses Beispiel verdeutlicht auch eine der Einschränkungen, denen Gesten, die zur Ausführungszeit erkannt werden sollen, unterliegen: Da es sich um eine mehrzügige Geste handelt, kann die Reaktion zur Laufzeit nicht erfolgen, weil unklar ist, ob, wann und wie der Benutzer den zweiten Zug ausführt. Davon unberührt bleibt, dass Anwendungen Benutzern grundsätzlich ein sofortiges Feedback geben sollten, wenn mit dem UI interagiert wird, um den Eindruck eines Defekts

oder Verwirrung bei Nutzern zu vermeiden (Wigdor & Wixon, 2011, S. 45). Für eine Multi-Touch-Anwendung kann dies z. B. bedeuten, dass, sobald die Oberfläche berührt wird, eine entsprechende Animation am Ort der Berührung dargestellt wird, ungeachtet der ausgeführten Geste.

Während Saffer (2008) die Reaktion in seiner Menge von Gesteneigenschaften nicht aufführt, wird sie von anderen Autoren in der Literatur mit verschiedenen Begriffen bedacht, die die beiden denkbaren Varianten dieser Gesteneigenschaft beschreiben:

- Wobbrock et al. (2009) unterscheiden diskrete (*discrete*) und kontinuierliche (*continuous*) Gesten.[12] Letztere beschreiben, dass die Anwendung während der Ausführung reagiert.

- Cirelli & Nakamura (2014)[13] hingegen unterscheiden symbolische (*symbolic*) und manipulative (*direct manipulation*) Gesten, wobei letztere eine Reaktion zur Laufzeit beschreiben (die zumeist eine Manipulation von Objekten auf dem Bildschirm zur Folge hat, daher wohl der Name). Die Bezeichnung ist jedoch verwirrend: Im o. g. Beispiel dient die Geste in Form eines X ebenfalls zur direkten Manipulation von Objekten, jedoch kann aufgrund ihrer mehrzügigen Natur nicht zur Laufzeit auf sie reagiert werden.

- Zhai et al. (2012, S. 107) unterscheiden analoge (*analogue*) und abstrakte (*abstract*) Gesten. Analoge Gesten ahmen Verhalten aus der realen Welt nach und sind entsprechend mit einer sofortigen Reaktion der Anwendung verbunden. Abstrakte Gesten – die sie auch als symbolische (*symbolic*) Gesten bezeichnen – hingegen entsprechen keinem Verhalten aus der realen Welt, sondern sind eben nur Gesten, denen eine Bedeutung für die Anwendung zugewiesen wird. Auf sie muss demnach erst nach der Ausführung reagiert werden. Die Autoren schränken allerdings ein, dass es sich bei der Kategorisierung eher um ein Spektrum denn eine strenge Zweiteilung handelt – Gesten können durchaus nur zum Teil analog sein. So kann ein virtuelles Objekt in einer Anwendung durch eine lange Berührung mit dem Finger aktiviert werden (eine eher symbolische Geste) und anschließend durch ein Wischen verschoben werden (eine analoge Geste).

- Kammer et al. (2015) sprechen in diesem Zusammenhang von *Offline*- und *Online*-Gesten. Eine Geste ist online, wenn auf eine Geste eine Reaktion zur Ausführungszeit folgt, und offline, wenn die Geste erst abgeschlossen wird, bevor die Anwendung reagiert.

Im Rahmen dieser Arbeit wird die Nomenklatur von Kammer et al. (2015) übernommen. Für die Gestenspezifikation ist die Unterscheidung, wann die Anwendung auf eine Geste reagiert, zwar recht einfach, jedoch ist sie für das Interaktionskonzept der Anwendung von entscheidender Bedeutung, weswegen sie während der Entwicklung explizit berücksichtigt und diskutiert werden muss. Viele der bekannten Standardgesten in mobilen Betriebssystemen sind Online-Gesten (Cirelli & Nakamura, 2014), wodurch sie einen gewissen Verbreitungsgrad erlangt haben und Benutzer eben mit diesen Gesten ver-

[12] Buxton (2014) verwendet die gleichen Begriffe, um pfadbasierte Gesten von einfachen Aktionen wie das Antippen eines Buttons zu unterscheiden.

[13] Die Autoren verweisen auf Wobbrock et al. (2009) als Quelle, jedoch findet sich diese Unterscheidung hier nicht.

schiedene Funktionalitäten verknüpfen. Gleichzeitig ist bei Online-Gesten zu berücksichtigen, dass Ähnlichkeiten im Interaktionskonzept zu Fehlinterpretationen führen, die wiederum falsche Reaktionen der Anwendung provozieren und für entsprechenden Frust bei Benutzern sorgen.

Offline-Gesten sind mutmaßlich einfacher zu implementieren: Die Verarbeitung findet erst statt, nachdem alle Eingaben vollständig sind, wodurch keine Unsicherheit über potentielle weitere Eingaben besteht. Nach Cirelli & Nakamura (2014) sind Offline-Gesten aber auch eher an den Kontext der Anwendung gebunden, wodurch sie für Anwender weniger offensichtlich sind und daher erst erlernt werden müssen. Dadurch sind sie schwieriger zu vermitteln. Zhai et al. (2012) stimmen dem implizit zu, wobei sich die Autoren in ihren Ausführungen darauf beziehen, dass Online-Gesten umso leichter zu erlernen sind, je näher sie an den bisherigen Erfahrungen der Benutzer liegen. Außerdem erörtern sie, dass die Anzahl an Online-Gesten limitiert ist, beziehen sich aber vorwiegend auf Gesten, die eine Entsprechung in der realen Welt haben und daher nicht frei gestaltet werden können, sondern an der physischen Interaktion in der gegebenen Domäne angelehnt werden müssen. Offline-Gesten hingegen können losgelöst von physischer Interaktion definiert werden und erlauben so mehr Freiheiten.

Für die Spezifikation einer Geste spielen die Unterschiede in der Ausführung zwischen Online- und Offline-Gesten allerdings keine Rolle. Es muss lediglich angegeben werden, ob die Anwendung auf die Geste während der Ausführung oder erst danach reagieren soll.

5.1.12 Rotation

Rotation beschreibt in Bezug auf Gesten zwei unterschiedliche Dinge: Zum einen können kreisförmige Bewegungen ausgeführt werden, die mit einer Bewegung im Raum einher gehen und besser mit der Form der Geste beschrieben werden (vgl. Kap. 5.1.7); zum anderen kann das verwendete Werkzeug während der Ausführung der Geste gedreht werden. Hoggan et al. (2013) untersuchen die Auswirkungen kreisförmiger Multi-Touch-Gesten auf interaktiven Oberflächen mit zwei Fingern um einen zentralen Mittelpunkt. Sie stellen fest, dass insbesondere bei steigendem Radius die Gesten von ihren Probanden immer ungenauer ausgeführt wurden. Darüber hinaus ergeben sich Unterschiede in Rotationen im und gegen den Uhrzeigersinn. Die Autoren untersuchen allerdings nur Gesten mit zwei Fingern.

Ein Beispiel für eine Rotation ohne Bewegung im Raum ist, dass in einer VR/AR-Anwendung das Auf- und Zudrehen eines Ventils mit einer Geste simuliert werden kann, bei dem der Anwender – analog zu einem echten Ventil – die Hand nach links oder rechts dreht. Auf einer berührungsempfindlichen Oberfläche hingegen können die Finger gedreht werden. Die jeweilige Position kann z. B. durch Auswertung der Form der Auflagefläche erkannt werden. Die Möglichkeiten zur Rotation sind jedoch je nach verwendetem Finger durch die menschliche Anatomie begrenzt (Wang & Ren, 2009).

Zu spezifizieren sind folglich zwei Informationen: in welche Richtung und wie weit gedreht wird. Während auf interaktiven Oberflächen nur zwei Richtungen möglich sind (im und gegen den Uhrzeigersinn), müssen bei räumlichen Gesten sechs Richtungen berücksichtigt werden (auf jeder Achse zwei Richtungen). Abb. 5.7 und Abb. 5.8 stellen diesen Unterschied gegenüber.

Wie weit die Rotation gehen kann, hängt vom jeweils verwendeten Werkzeug ab; insbesondere sind bei räumlichen Gesten mit der Hand anatomische Grenzen zu beachten. Eine Spezifikation kann helfen, eventuelle Überschreitungen gesunder Werte vorab zu prüfen.

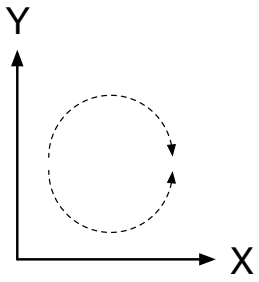


Abbildung 5.7: Rotation für Oberflächengesten

5.1.13 Werkzeug

Moderne Sensoren machen es möglich, dass Gesten mit unterschiedlichen Werkzeugen ausgeführt werden können. Bevor heute übliche Smartphones und Tablets eingeführt wurden, die meist mit den Fingern bedient werden, waren bereits Personal Digital Assistants (PDAs) verbreitet, die in der Regel mit einem speziellem Stift – dem Stylus – bedient wurden. Auch bei digitalen Whiteboards werden Stifte mitgeliefert, die anstatt der Finger für genauere Gestenzüge und zum Zeichnen oder Schreiben verwendet werden. Außerdem werden zu einem digitalen Whiteboard auch oft (nachgestellte) Wischer mitgeliefert, die einen klassischen Wischer an analogen Whiteboards imitieren und auch zum Löschen von Objekten verwendet werden können, gleichzeitig aber auch weiterführende Optionen für die Interaktion mit dem digitalen Whiteboard ermöglichen (z. B. könnte der Wischer in einen Zeichnenmodus versetzt und als dicker Stift verwendet werden).

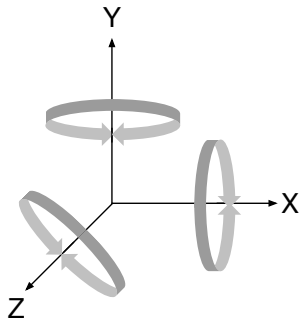


Abbildung 5.8: Rotationsachsen für räumliche Gesten

Saffer (2008) beschreibt, dass eine Eigenschaft gestenbasierter Anwendungen die explizite Berücksichtigung physischer Werkzeuge sein kann. Harrison et al. (2011) demonstrieren, wie Stifte mit unterschiedlichen Aufsätzen[14] erkannt und unterschieden werden können. Sie demonstrieren auch, wie sich verschiedene Teile des Fingers[15] erkennen und somit als Werkzeug explizit einsetzen lassen. Wigdor et al. (2011) verwenden Faust und Handkante einer Hand auf einer berührungsempfindlichen Oberfläche, um die Bedeutung einer Geste mit der anderen Hand zu ändern. Valdes et al. (2014) gehen noch einen Schritt weiter und verwenden interaktive Werkzeuge, die um individuelle Programmierung erweitert werden können. Das Werkzeug kann so an den aktuellen Zustand der Anwendung angepasst werden.

Zwar kritisieren verschiedene Autoren (z. B. Wobbrock et al. (2009) und Rekik et al. (2014)) die Verwendung unterschiedlicher Werkzeuge als zu kompliziert, jedoch lassen sie außen vor, dass auch in zwischenmenschlicher Interaktion gleiche Gesten mit verschiedenen Werkzeugen ebenfalls unterschiedliche Bedeutung haben können. So existiert ein deutlicher Unterschied der Gesten Daumen hoch, Zeigefinger hoch und Mittelfinger hoch in den meisten westlichen Kulturkreisen. Colley & Häkkilä (2014) setzen die Verwendung unterschiedlicher Finger für verschiedene Funktionalitäten auch in einer Anwendung für mobile Geräte um. Sie zeigen, dass durch die Verwendung gleicher Gesten mit unterschiedlichen Fingern die Effizienz der Bedienung gesteigert werden kann. Insbesondere äußerten Probanden

[14] u. a. Tischtennisball, Schaumstoff, Holz

[15] Fingerspitze, -kuppe, -nagel und -knöchel

den Wunsch, die Konfiguration individualisieren zu können, um bevorzugte Funktionalitäten schnell erreichbar zu haben.

Ebenso können auch gänzlich unterschiedliche Werkzeuge eingesetzt werden. Tu et al. (2012) vergleichen Gesten, die mit Finger oder Stift ausgeführt werden, und finden Gemeinsamkeiten und Unterschiede in Bezug auf verschiedene Eigenschaften: Probanden konnten z. B. Gesten mit dem Finger schneller ausführen, jedoch mit Stiften feingliedriger. Vatavu et al. (2014) weisen nach, dass Benutzer Gesten, die strukturelle Ähnlichkeit besitzen – z. B. Gesten in Form der Buchstaben *D* und *P* –, mit Stiften wesentlich genauer durchführen können als mit den Fingern. Tu et al. (2015) bestätigen die Ergebnisse und zeigen außerdem, dass Stifte den Fingern bei komplexeren Gesten überlegen sind, insbesondere auf kleinen Bildschirmen. Gleichzeitig zeigen die Autoren, dass diese Unterschiede durch Bewegung der Nutzer – z. B. wenn das Gerät im Gehen bedient wird – wieder relativiert werden und dann alle Werkzeuge in etwa gleiche Ergebnisse liefern. Diese Unterschiede sollten bewusst berücksichtigt und eingesetzt werden.

Kleffmann et al. (2014) integrieren eine explizite Trennung zwischen Werkzeugen in einer Anwendung zum kooperativen Modellieren von Prozessen auf großen, multi-touch-fähigen Displays. Sie verwenden den Stift zum Zeichnen von Diagrammen, die Finger zu deren Manipulation sowie den Wischer oder Handballen zum Löschen von Elementen. Frisch et al. (2009, 2010) sowie Frisch & Dachselt (2013) hingegen zeigen, wie man Stifte und Finger zur Ausführung von Multi-Touch-Gesten kombinieren kann. Sie betrachten ebenfalls eine Anwendung, mit der Diagramme und Geschäftsprozesse skizziert werden können und leiten mithilfe einer Studie mögliche Gesten ab. Ein Beispiel für eine Kombination von Gesten mit Finger und Stift ist das Kopieren von Elementen – dazu wird ein Element mit dem Finger berührt und mit dem Stift eine Geste vom Element zum Standort der Kopie ausgeführt.

Hinckley et al. (2010) beschreiben ebenfalls verschiedene Techniken, um Gesten mit Stift und Fingern zu kombinieren. Ihre Anwendung – eine Art virtuelle Pinnwand – integriert eine Trennung von Werkzeugen für bestimmte Funktionen. Benutzer müssen aber nicht aktiv in einen Bearbeitungs- oder Eingabemodus umschalten: Die Umschaltung erfolgt implizit durch die Verwendung des jeweiligen Werkzeugs. Sie integrieren aber auch Gesten, bei denen Finger und Stift gleichzeitig eingesetzt werden, z. B. um Objekte zu stapeln. Dies erfolgt in zwei Schritten: Zuerst werden beliebig viele Objekte mit den Fingern angetippt, um diese zu markieren; anschließend kann ein Objekt mit dem Finger berührt und gleichzeitig mit dem Stift angetippt werden, um die Erstellung des Stapels auszulösen (das Objekt, das mit Stift und Finger berührt wurde, liegt dann oben). Auch Brandl et al. (2008) und Zeleznik et al. (2010) demonstrieren Kombinationsmöglichkeiten für Finger und Stift und zeigen, wie Interaktion erweitert und verfeinert werden kann.

Ein für virtuelle 3D-Umgebungen entwickeltes Werkzeug, mit dem räumliche Gesten ausgeführt werden können, ist ein *Magic Wand*, eine einem Zauberstab nachempfundene 3D-Maus (z. B. bei Ciger et al. (2003) und de Sousa & Morimoto (2015)). Khan et al. (2012) zeigen, dass Magic Wands in virtuellen Räumen effizienter sind, als aus der Position und dem Blickwinkel des Be-

nutzers im Raum Informationen abzuleiten.[16] Daraus folgt natürlich direkt die Möglichkeit, analog zu unterschiedlichen Werkzeugen auf interaktiven Oberflächen, auch unterschiedliche Magic Wands in virtuellen Räumen zu verwenden, um verschiedene Aktionen auszulösen. Räumliche Gesten können aber auch mit mobilen Geräten ausgeführt werden.

[16] Die Autoren bezeichnen diese Art der Interaktion als *Human Joystick*.

Gleichzeitig kann je nach Anwendungsfall die Einschränkung auf bestimmte Werkzeuge besonders interessant sein. So berichten z. B. Döring et al. (2011), dass beim Führen eines Fahrzeugs Gesten auf dem Lenkrad interessant sind, jedoch aus Sicherheitsgründen nur der Daumen verwendet werden kann. Fahrzeugführer müssen beide Hände am Lenkrad lassen. Folglich kann eine exakte Spezifikation auch genutzt werden, um die definierten Gesten auf Verstöße gegen andere definierte Anforderungen der Anwendung – in diesem Falle Sicherheitsanforderungen – zu prüfen.

Für die Spezifikation einer Geste ist insbesondere relevant, mit welchem Werkzeug die Geste oder Teilgeste durchgeführt wird. Die verfügbaren Werkzeuge sind natürlich abhängig vom verwendeten Sensor, insofern ist eine fest definierte Liste von Werkzeugen schwierig zu erstellen. Vielmehr muss im Rahmen der Entwicklung festgelegt werden, welche Werkzeuge einzusetzen sind und wie diese definiert werden.

5.2 Beziehungen zwischen Teilgesten

Gesten können aus Teilgesten komponiert werden. Die Beziehung zwischen den Teilgesten ist somit wesentlicher Bestandteil der Beschreibung einer Geste. Erst diese Beziehung erlaubt letztlich die Definition komplexer, mehrzügiger Gesten, die aus einzelnen einfachen Gesten zusammengestellt werden.

5.2.1 Zeitliche Relation

Die einzelnen Teile einer Geste stehen in einem zeitlichen Bezug zueinander, der die Ausführungsreihenfolge sowie die zeitlichen Abstände zwischen den Teilgesten definiert. Dabei können sequentielle und parallele Gesten unterschieden werden. Werden Teilgesten hintereinander in einer definierten Reihenfolge ausgeführt, so handelt es sich um sequentielle Gesten. Diese können noch in einzügige (*single-stroke*) und mehrzügige (*multi-stroke*) Gesten unterteilt werden, wobei mehrzügige Gesten eine Pause zwischen den einzelnen Teilgesten enthalten. Bei Oberflächengesten wird die Pause durch das Abheben und Wiederaufsetzen des Werkzeugs deutlich; bei räumlichen Gesten ist die Unterscheidung nicht ganz so einfach, da die Sensoren üblicherweise alle Bewegungen des Benutzers aufnehmen und die Anwendung somit zwischen sinnvollen Teilgesten und Rauschen in den Pausen zwischen den Gesten unterscheiden muss. Bei parallelen Gesten werden Teilgesten gleichzeitig ausgeführt, was sowohl bei Oberflächengesten (die dann als *Multi-Touch-Gesten* bezeichnet werden) als auch bei räumlichen Gesten möglich ist. Saffer (2008) geht explizit auf Parallelisierung in seinen Gesteneigenschaften ein, lässt die unterschiedlichen sequentiellen Gesten jedoch aus.

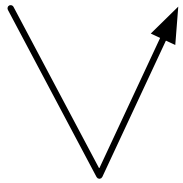


Abbildung 5.9: Einzügige Geste in Form eines V

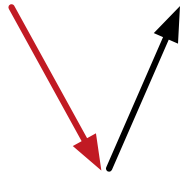


Abbildung 5.10: Mehrzügige Geste in Form eines V

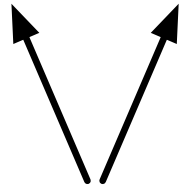


Abbildung 5.11: Multi-Touch-Geste in Form eines V

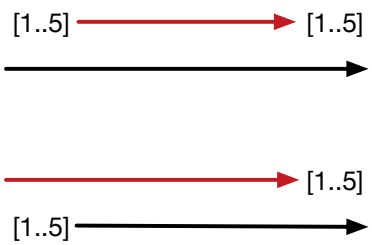


Abbildung 5.12: Zeitliche Variationen bei Multi-Touch-Gesten

Einzügige Gesten bestehen aus einem kontinuierlichen Gestenzug. Alle Teile der Geste werden mit demselben Werkzeug ausgeführt; ein Wechsel des Werkzeugs führt zu einer Unterbrechung des Gestenzuges. Zhai et al. (2012) sprechen hier von *Gesten der ersten Ordnung* (vgl. Kap. 4.1.1). Bei Oberflächengesten werden die einzelnen Teilgesten ausgeführt, ohne das Werkzeug von der Oberfläche abzuheben: Die Geste beginnt, wenn das Werkzeug die Oberfläche berührt, und endet, wenn es sie verlässt. Ein Beispiel, in dem eine einzügige Geste in Form eines V dargestellt ist, zeigt Abb. 5.9. Sowohl der linke, absteigende Schenkel, als auch der rechte, aufsteigende Schenkel werden in einem Zug ausgeführt und das Werkzeug wird nicht von der Oberfläche abgehoben. Diese Geste ist auch als räumliche Geste denkbar; sie kann z. B. mit der rechten Hand vor dem Körper ausgeführt werden.

Mehrzügige Gesten hingegen sind Sequenzen einzügiger Gesten. Dabei wird die Ausführung der Geste zwischendurch unterbrochen. Bei einer Oberflächengeste wird das Werkzeug zwischen Teilgesten von der Oberfläche abgehoben und wieder aufgesetzt. Abb. 5.10 veranschaulicht, wie das Beispiel des einzügigen V als mehrzügige Geste durchgeführt wird. Jeder Schenkel des V ist nun ein separater Gestenzug (also eine einzügige Teilgeste). Am Scheitelpunkt zwischen beiden Teilgesten wird das Werkzeug ab- und wieder aufgesetzt und die Ausführung der Geste fortgeführt. Die dadurch entstehende zeitliche Lücke gilt es zu spezifizieren. Cirelli & Nakamura (2014) kritisieren an aktuellen Möglichkeiten zur Gestenspezifikation, dass keine in der Lage ist, die zeitliche Relation an dieser Stelle individuell je Geste zu definieren. Typischerweise werden hier globale Konstanten verwendet, die in den jeweiligen Erkenner-Frameworks definiert werden.

Parallele Gesten sind im Grunde genommen mehrzügige Gesten, jedoch werden die einzelnen Teilgesten zeitgleich mit mehreren Werkzeugen ausgeführt. Zhai et al. (2012) fassen daher mehrzügige und parallele Gesten als *Gesten höherer Ordnung* zusammen (vgl. Kap. 4.1.1). Abb. 5.11 greift wieder auf das Beispiel einer Geste in Form eines V zurück und veranschaulicht, wie diese durch Parallelisierung zweier einzügiger Gesten ausgeführt werden kann. Die einzelnen Züge stellen dabei wieder die beiden Schenkel dar, welche jedoch diesmal unten am Scheitelpunkt starten und zeitlich parallel zueinander ausgeführt werden. Dabei entfernen sie sich voneinander (vgl. Kap. 5.2.2).

Während es in einzügigen Gesten – abgesehen davon, dass Teilgesten sequentiell ausgeführt werden – keine definierbare zeitliche Relation gibt und in mehrzügigen Gesten lediglich der zeitliche Abstand zwischen Zügen definiert werden kann, existieren für Multi-Touch-Gesten verschiedene Optionen: Teilgesten können in einem bestimmten zeitlichen Fenster einer anderen Teilgeste ausgeführt werden oder in einem zeitlichen Versatz zueinander stehen. Abb. 5.12 veranschaulicht zwei Optionen anhand einer einfachen Geste mit zwei Fingern, die auf der Oberfläche in gerader Linie nach rechts bewegt werden. In beiden Fällen wird zur Angabe der zeitlichen Relation eine fiktive Zeiteinheit verwendet, die im Nachfolgenden als *Tick* bezeichnet wird. Im oberen Teil der Abbildung startet die untere Teilgeste eins bis fünf Ticks vor der oberen und endet eins bis fünf Ticks nachher – mit anderen Worten, der obere Finger wird ein bis fünf Ticks später auf die Oberfläche aufgesetzt und

ein bis fünf Ticks früher abgehoben. Im unteren Beispiel wird der zeitliche Versatz zweier Teilgesten dargestellt: Die obere Geste fängt ein bis fünf Ticks vor der unteren an, hört jedoch auch ein bis fünf Ticks vorher auf. Wenn der zeitliche Versatz für die Anwendung relevant ist, dann muss er auch spezifiziert werden können.

In den bisher beschriebenen Varianten waren Teilgesten lediglich Bestandteil einer Gesamtgeste und jede Teilgeste hatte für sich genommen keine semantische Bedeutung für die Anwendung.[17] Cirelli & Nakamura (2014) erweitern dieses Konzept und führen eine weitere Kategorisierung für die zeitliche Relation zwischen Teilgesten ein. Sie unterscheiden zwischen *atomaren* und *sequentiellen* Gesten, wobei sequentielle Gesten einer Reihe von atomaren Gesten in definierten zeitlichen Intervallen entsprechen. Cirelli & Nakamura (2014) beschreiben in ihrem Ansatz somit die Kombination verschiedener Einzelgesten, die jede für sich eine eigene semantische Bedeutung für die Anwendung haben, zu *Kombo-Gesten*, ähnlich der aus Computerspielen bekannten Möglichkeit, einfache Einzelaktionen zu einem zusätzlichem Spezialangriff zu kombinieren.

[17] Mit anderen Worten: Die Teilgesten selber waren u. U. keine gültigen Gesten, sondern konnten in der Regel nur in Kombination mit anderen Teilgesten verwendet werden.

Die Möglichkeit, zeitliche Relationen explizit modellieren zu können, ermöglicht auch neue Arten von Gesten. Ghomi et al. (2012) führen das Konzept der *rhythmischen Interaktion* ein, das Ähnlichkeiten mit der Technik des Morsens aufweist. Sie beschreiben, dass rhythmische Muster von Benutzern zur Interaktion mit Geräten eingesetzt werden können. Dabei unterscheiden sie bei Geräten mit berührungsempfindlichen Bildschirmen die Dauer des Kontakts (vgl. Kap. 5.1.5) und die Dauer zwischen zwei Kontakten, die der zeitlichen Relation entspricht. Damit lässt sich das Konzept auch auf Gestensteuerung anwenden, um rhythmische Gesten durchzuführen. Hinweise darauf, dass Benutzer rhythmische Gesten auf mobilen Geräten intuitiv verwenden können, finden Shimon et al. (2015). Sie untersuchen zwar in ihrer Studie eine andere Forschungsfrage, schließen jedoch aus ihren Beobachtungen, dass rhythmische Gesten die Möglichkeiten für Gestensteuerung vergrößern. Heo et al. (2014) erweitern rhythmische Gesten noch um die Entfernung zwischen einzelnen Berührungen und führen somit noch die räumlichen Relation (vgl. Kap. 5.2.2) als zusätzlichen Faktor ein. Auch Oakley et al. (2015) demonstrieren, dass rhythmische Gesten einen Mehrwert für die Interaktion mit berührungsempfindlichen Oberflächen bringen, insbesondere wenn die Oberfläche sehr klein ist, wie bei Smartwatches üblich.

Die dargestellten Variationen der zeitlichen Relation zwischen Teilgesten sowie die Einsatzmöglichkeiten z. B. als rhythmische Gesten zeigen die Möglichkeiten auf, die sich für unterschiedliche Arten von Gesten ergeben. Zu definieren sind demnach die Unterschiede zwischen sequentiellen und parallelen sowie einzügigen und mehrzügigen Gesten. Des Weiteren müssen die Zeiträume, die zwischen Teilgesten liegen, spezifiziert werden können.

5.2.2 Räumliche Relation

Neben dem zeitlichen Bezug stehen einzelne Teilgesten einer Geste in einem räumlichen Bezug zueinander. Dabei ist sowohl die Lage zueinander als auch

die Entfernung zu unterscheiden. Die Lage zueinander beschreibt, ob sich einzelne Gestenzügen z. B. übereinander, untereinander, links oder rechts voneinander befinden.

Während Saffer (2008) und Buxton (2014) den räumlichen Bezug zwischen Teilgesten nicht explizit als Gesteneigenschaft aufnehmen, gehen Entwickler von Frameworks zur Erkennung von Gesten detaillierter auf sie ein, insbesondere wenn sie auch eine Notation zur Beschreibung von Gesten verwenden. So definieren Scholliers et al. (2011) verschiedene Operatoren, um die räumliche Relation zu beschreiben. Sie ermöglichen damit, dass Teilgesten in der Nähe anderer Teilgesten positioniert oder von ihnen umschlossen werden können. Kammer et al. (2015) erweitern den räumlichen Bezug noch um bewegungsspezifische Optionen. Sie beschreiben nicht nur die Position von Teilgesten zueinander, sondern auch ihre Bewegungsrichtung mit Bezug auf andere Teilgesten. Teilgesten können sich aufeinander zu- oder voneinander wegbewegen, sie können sich kreuzen oder sich am jeweiligen Anfang oder Ende berühren.

Heo et al. (2014) verwenden eine Kombination aus zeitlicher (vgl. Kap. 5.2.1) und räumlicher Relation, um aus Sequenzen einzelner Berührungen auf einem berührungsempfindlichen Bildschirm semantische Informationen abzuleiten. Dabei spielt insbesondere die Entfernung zwischen einzelnen Berührungen eine Rolle: Je nachdem, wie nah die zweite Berührung am Ort der ersten ist, erhält die Geste eine andere semantische Bedeutung.

Zu spezifizieren ist demnach die Positionierung von Teilgesten zueinander. Außerdem muss definiert werden können, wie weit Teilgesten voneinander entfernt ausgeführt werden sollen.

6 Verwandte Arbeiten – Notationen zur Beschreibung von Gesten

In den letzten Jahren wurden unterschiedliche Notationen zur Beschreibung von Gesten entwickelt, die gelegentlich als Gesture Description Languages (GDLs) bezeichnet werden. Viele Arbeiten fokussieren dabei Multi-Touch-Gesten, jedoch existieren auch einige Ansätze zur Beschreibung räumlicher Gesten. Ein überwiegender Teil der Arbeiten hat die Unterstützung von Software-Entwicklern zum Ziel. Daher werden sie oftmals gemeinsam mit einem Gestenerkenner definiert, der über die Notation konfiguriert werden kann.

Die Notationen lassen sich grob in drei Kategorien einteilen, die im Folgenden näher betrachtet werden sollen: musterbasierte (Kap. 6.1), textuelle (Kap. 6.2) und graphische Notationen (Kap. 6.3). Diese haben gemein, dass im Regelfall eine strikte Trennung zwischen ihnen herrscht, d. h. bestehende Notationen sind typischerweise rein musterbasiert, textuell oder graphisch. Hybride Notationen (Kap. 6.4) hingegen bilden eine Ausnahme und mischen unterschiedliche Arten der Beschreibung. Im Vergleich zu reinen Ansätzen sind sie jedoch deutlich unterrepräsentiert: Im Rahmen dieser Arbeit konnte nur eine weitere Notation identifiziert werden, die einen hybriden Ansatz verfolgt.

Darüber hinaus ist die Beschreibung von Körperbewegungen – und als nichts anderes muss Gestensteuerung betrachtet werden – noch in anderen Bereichen interessant. Abschließend sollen daher Notationen aus angrenzenden Anwendungsbereichen betrachtet werden, die in direktem Bezug zu HCI stehen und vielleicht interessante Konzepte enthalten (Kap. 6.5). So existieren verschiedene Ansätze zur Spezifikation von Gebärdensprache, die eingesetzt werden, um z. B. Avatare oder Roboter zu konfigurieren. Außerdem wurden in der Vergangenheit zahlreiche Notationen entwickelt, die zur Dokumentation von Tänzen und Choreographien eingesetzt werden und in letzter Zeit auch im HCI-Umfeld auf Interesse gestoßen sind.

6.1 Musterbasierte Ansätze

Die Beschreibung von Gesten mithilfe musterbasierter Ansätze hat ihren Ursprung eigentlich in der Konfiguration passender Erkenner und weniger in der Spezifikation von Gesten. Der Unterschied zwischen einzelnen Ansätzen liegt in der technischen Umsetzung, aber das zugrundeliegende Prinzip ist identisch: Eine Geste (oder mehrere Variationen) wird aufgezeichnet und fortan als Grundlage für die Erkennung verwendet, indem alle durchgeführten Gesten mit der zuvor aufgezeichneten verglichen werden. Vereinfacht ausgedrückt gibt es eine Beispielgeste, die als Spezifikation verwendet wird. Dieses Beispiel kann auch zur Kommunikation zwischen Projektbeteiligten

eingesetzt werden. Somit sind musterbasierte Ansätze eine unkomplizierte Art, Gesten zu spezifizieren und zu dokumentieren.

Als Urvater der verschiedenen Verfahren muss Rubine (1991) betrachtet werden, jedoch wurden in den vergangenen Jahren weitere Verfahren entwickelt, die performanter arbeiten und mehr Gesten erkennen können. Dabei muss insbesondere die $-Familie rund um $1 (Wobbrock et al., 2007), $N (Anthony & Wobbrock, 2010) und $P (Vatavu et al., 2012) hervorgehoben werden, die zur Erkennung von Oberflächengesten entwickelt wurden. Zur Familie haben sich über die Zeit noch zahlreiche optimierte Verwandte hinzugesellt (u. a. 1¢ (Herold & Stahovich, 2012), $N-Protractor (Anthony & Wobbrock, 2012) und PennyPincher (Taranta II & LaViola Jr., 2015)). Verschiedene Arbeiten (z. B. Kammer et al. (2015), Parra González et al. (2015) und Hesenius et al. (2017)) bauen zudem auf der $-Familie auf. Andere erweitern sie, um auch räumliche Gesten zu erkennen, z. B. auf Basis eines Accelerometers (Kratz & Rohs, 2010). Daneben existieren weitere Verfahren, die Techniken wie Dynamic Time Warping zur Erkennung einsetzen (z. B. Bomsdorf & Blum (2014), die auch einen passenden Editor mitliefern, und Taranta II et al. (2017)). Die $-Familie liefert jedoch gute Ergebnisse in Erkennungsleistung und Geschwindigkeit bei überschaubarer Komplexität.

Bei aller Einfachheit haben rein musterbasierte Verfahren aber auch Nachteile. Ungeachtet verschiedener technischer Einschränkungen – beispielsweise können nicht alle Mitglieder der $-Familie zur Erkennung von Multi-Touch-Gesten eingesetzt werden – fokussieren existierende musterbasierte Ansätze vor allem die Form einer Geste. Sie sind in der Regel in der Lage, verschiedene Ausführungen der gleichen Form – z. B. unterschiedlicher Ausrichtung – als identische Geste zu erkennen. Allerdings kann es sein, dass die Form einer Geste zum Zeitpunkt der Definition u. U. nicht bekannt ist. Derartige *Freiformgesten* können mit musterbasierten Ansätzen – im Gegensatz zu anderen Ansätzen – nicht definiert werden.

Darüber hinaus können die Eigenschaften einer Geste nicht explizit gemacht werden: Einer aufgezeichneten Geste sieht man Informationen wie z. B. die Geschwindigkeit der Ausführung oder den ausgeübten Druck nicht an. Musterbasierte Ansätze führen in diesem Falle nicht zu menschenlesbaren Spezifikationen und sind somit zur Kommunikation von Gesten ungeeignet, wenn außer der Form noch andere Informationen definiert werden müssen. Wie in Kap. 4 diskutiert sind diese Eigenschaften jedoch wesentlich für die unterschiedlichen Entscheidungen, die für die Implementierung einer gestenbasierten Anwendung getroffen werden müssen. Daher wird auf weitergehende Betrachtung musterbasierter Ansätze verzichtet.

6.2 Textuelle Notationen

Textuelle Notationen stellen eine Menge von Schlüsselwörtern und Operatoren zur Verfügung, die die Definition der einzelnen Teilgesten und ihre Beziehung zueinander ermöglichen. Unterschiedliche Ansätze wurden in den letzten Jahren entwickelt, die nachfolgend betrachtet werden sollen. Ihnen ist gemein, dass sie – analog zu musterbasierten Ansätzen – auch einen pas-

senden Gestenerkenner konfigurieren sollen, jedoch zum einen eine genauere Definition ermöglichen, indem sie die Beschreibung unterschiedlicher Eigenschaften integrieren, und zum anderen besser zu handhaben sind, da sie anstreben, sowohl menschen- als auch maschinenlesbar zu sein. Gerade die Notwendigkeit der Maschinenlesbarkeit führt jedoch oft zur Einführung weiterer Regeln und Zeichen in die Notation, die lediglich zur Strukturierung des Ausdrucks dienen und nicht direkt mit der Beschreibung der Geste und ihrer Eigenschaften zusammenhängen. Kammer et al. (2015) bezeichnen dies als *syntaktischen Overhead*. Es ist davon auszugehen, dass die Notationen durch zusätzliche Regeln und Zeichen für Menschen schwieriger zu lesen sind. Im Folgenden soll der beschreibende Aspekt und die Menschenlesbarkeit textueller Notationen im Vordergrund stehen. Einen Überblick über unterschiedliche Ansätze mit Blick auf technische Aspekte bieten Hoste & Signer (2014).

Während textuelle Notationen gegenüber musterbasierten Ansätzen klar eine größere Ausdrucksstärke für sich verbuchen können und mehr Gesten menschenlesbar beschreiben können, kommen sie insbesondere bei der Beschreibung der Form schnell an ihre Grenzen. Die unterschiedlichen Ansätze gehen mit der Vielfältigkeit der Formen, die Gestensteuerung ermöglicht, auf verschiedene Weise um. Ebendiese Vielfalt macht textuellen Notationen aber schnell zu schaffen: Oftmals kann Vielfalt nur über mehr Schlüsselwörter, Operatoren und syntaktische Konstrukte abgebildet werden, wodurch die Gefahr der Unübersichtlichkeit und Kompliziertheit erhöht wird. Der Einsatz einer textuellen Notation zur Spezifikation führt somit letztlich dazu, dass Teammitglieder die unterschiedlichen Elemente erlernen und verstehen müssen, wodurch sie zunächst in ihrer Kommunikationsfähigkeit gehindert werden. Die Erweiterung der Notation, z. B. um weitere Eigenschaften, verstärkt diesen Effekt: Mehr Eigenschaften führen zu mehr Schlüsselwörtern und Operatoren, die wiederum zu komplizierteren Ausdrücken führen.

6.2.1 GDL (Echtler) / GISpL

GISpL (Echtler & Butz, 2012) – kurz für *Gestural Interface Specification Language* – ist eine Weiterentwicklung der GDL nach Echtler et al. (2010). Das grundlegende Konzept beider Ansätze identisch, lediglich die Darstellungsform unterscheidet sich.[1] An dieser Stelle soll daher insbesondere auf GISpL eingegangen werden. Mit der Notation können sowohl Oberflächengesten als auch räumliche Gesten beschrieben werden.

[1] GISpL setzt auf Java Script Object Notation (JSON), während die GDL nach Echtler et al. (2010) eine eigene Notation nutzt.

GISpL unterteilt Gesten (*Gestures*) in zwei wesentliche Teile: Regionen (*Regions*) und Eigenschaften (*Features*). Gesten werden aus unterschiedlichen Eigenschaften zusammengesetzt und durch eine Region einer Fläche (oder, im Falle räumlicher Gesten, einem Raum) zugeteilt, auf der sie gültig sind. Einer Region können dabei beliebig viele Gesten zugewiesen werden, um z. B. mehrere Gesten auf einer bestimmten Fläche zu ermöglichen. Zur genaueren Spezifikation einer Geste bietet GISpL verschiedene Eigenschaften, die mit Parametern präzisiert werden können.

Abb. 6.1 zeigt ein Beispiel: ein Swipe mit zwei Fingern. Zeile 3 setzt zwei `flags`, um die Geste genauer zu spezifizieren. Das Flag `sticky` definiert,

```
1  {
2    "name":"two_finger_swipe",
3    "flags":"sticky,oneshot",
4    "custom":"userdata",
5    "features":[
6      {
7        "type":"Count",
8        "filters":2046,
9        "constraints":[2,2],
10       "result":[]
11     },{
12       "type":"Motion",
13       "filters":2046,
14       "constraints":[
15         [0.01,-0.01,-0.01],
16         [0.1,0.01,0.01]
17       ],
18       "result":[]
19     }
20   ]
21 }
```

Abbildung 6.1: Ein Swipe mit zwei Fingern in GISpL (Quelle: Echtler & Butz (2012))

dass die Geste auch außerhalb ihrer Startregion weiter ausgeführt werden kann, und das Flag `oneshot` legt fest, dass es sich um eine Offline-Geste (vgl. Kap. 5.1.11) handelt. Die beiden Eigenschaften in den Zeilen 5-20 schränken die Anzahl der Finger und die Bewegungsrichtung ein. Die Eigenschaft `Count` legt fest, dass hier die minimale und maximale Anzahl der verwendeten Werkzeuge definiert wird. In Zeile 8 definiert der Wert `2046` des Feldes `filters` die Verwendung der Finger und in Zeile 9 legt das Feld `constraints` die minimale und maximale Zahl der Berührungen (zwei) fest. Die zweite Eigenschaft vom Typ `Motion` in Zeile 12 beschreibt dann die Bewegung. Diese wird durch das Feld `constraints` in den Zeilen 14-17 festgelegt. In diesem Falle wird durch den gegebenen Vektor eine Linie beschrieben. Eine Definition der Geschwindigkeit erfolgt in diesem Beispiel nicht. Eigenschaften können außerdem ein `result` definieren, das Informationen über die Eigenschaft an die Anwendung zurückgibt, die weiterverarbeitet werden können. Im Beispiel wird davon kein Gebrauch gemacht.

GISpL unterstützt darüber hinaus noch Eigenschaften, um Rotation, Skalierung sowie Verzögerung zu definieren. Außerdem können zahlreiche Werkzeuge mithilfe festgelegter Zahlen definiert werden, um Gesten nicht nur mit den Fingern, sondern auch mit Stiften und dergleichen durchzuführen.

Mit der im Rahmen dieses Vorhabens entwickelten Lösung teilt GISpL die grundlegende Idee, einzelne Gesten mithilfe unterschiedlicher Eigenschaften zu beschreiben, setzt diese jedoch anders um. Der Einsatz von JSON erleichtert zwar die automatische Verarbeitung, bringt aber gleichzeitig eine strukturelle Vorgabe ein, die – trotz grundsätzlicher Menschenlesbarkeit – zur Kommunikation von Gesten wenig geeignet ist. So muss z. B. das verwendete Werkzeug umständlich über eine Bitmaske statt über lesbare Konstanten definiert werden, wie die Autoren selbst eingestehen (Echtler & Butz, 2012, Fußnote S. 235). Durch den Verzicht auf graphische Elemente muss z. B. die Form als Menge von Vektoren beschrieben werden, die die Bewegung einschränken (vgl. Abb. 6.1, Zeilen 15–16).

6.2.2 GDL (Hachaj)

Hachaj & Ogiela (2014) fokussieren in ihrem ebenfalls GDL genannten Ansatz die Beschreibung räumlicher Gesten und Posen. Sie beschränken sich dabei nicht auf die Hände, sondern ermöglichen die Beschreibung unterschiedlicher Körperteile. Sie setzen dabei auf Microsoft Kinect als Basistechnologie und definieren Schlüsselwörter, die die unterschiedlichen Körperteile, die Kinect erkennen kann, definieren (z. B. `LeftHand` für die linke Hand).

Die definierten Schlüsselwörter können in logischen Regeln verwendet werden, um die Positionierung von Körperteilen zueinander zu beschreiben. Dazu können detaillierte Angaben, wie z. B. die minimale oder maximale Entfernung einzelner Körperteile voneinander, eingesetzt werden, um genaue Posen zu definieren. Diese können dann in erweiterten Regeln zu Gesten zusammengesetzt werden, bei denen die Posen als Start-, Zwischen- und Endpositionen verwendet werden. Die Bewegung zwischen den einzelnen Posen wird vorwiegend implizit festgelegt.

Abb. 6.2 zeigt ein Beispiel, wie eine Geste, bei der die Hände klatschend zusammengeführt werden, in der GDL nach Hachaj & Ogiela (2014) beschrieben wird. Die Zeilen 1 und 2 definieren die beiden grundlegenden Positionen: zunächst die zusammengeführten Hände und dann die auseinandergeführten. In den eckigen Klammern steht der Zeitpunkt, der berücksichtigt werden soll; `0` steht für den aktuellen Zeitpunkt und steigende Zahlen für die Vergangenheit. In Zeile 3 wird die eigentliche Geste als Sequenz von Zusammen- und Auseinanderführen definiert. Die Beschreibung ist ungenau, denn sie definiert nicht, wo die Hände zusammengeführt werden müssen; z. B. würde eine winkende Bewegung mit zwei Händen, bei der sich beide Hände über dem Kopf kreuzen, hier nicht ausgeschlossen werden.

```
1 RULE distance(RightHand.xyz[0], LeftHand.xyz[0]) < 100 THEN HandsTogether
2 RULE distance(RightHand.xyz[0], LeftHand.xyz[0]) >= 100 THEN HandsSeparate
3 RULE sequenceexists("[HandsSeparate,0.5][HandsTogether,0.5]
4       [HandsSeparate,0.5]") THEN Clapping
```

Abbildung 6.2: Regeln zur Beschreibung einer klatschenden Bewegung in GDL nach Hachaj & Ogiela (2014) (Quelle: ibd.)

Die GDL nach Hachaj & Ogiela (2014) beschreibt keine Bewegung, sondern Körperpositionen. Eine Geste besteht somit aus der Start- und Endposition (mit optionalen Zwischenpositionen), die Bewegung dazwischen wird jedoch nicht beschrieben. Allerdings kann in der Beschreibung der Geste auf die Position zu vergangenen Zeitpunkten verwiesen werden, sodass zumindest eine Bewegung erkannt werden kann; letztlich wird hier aber auch von Positionen ausgegangen. Ungeachtet dessen, dass Hachaj & Ogiela (2014) ebenfalls auf den Einsatz graphischer Elemente verzichten, wird im Rahmen dieser Arbeit auch ein anderer Ansatz verfolgt – nämlich die explizite Beschreibung der Bewegung, anstatt diese nur implizit zwischen Positionen anzunehmen.

6.2.3 GDL (Khandkar)

Khandkar & Maurer (2010a, b) verwenden einen Ansatz zur Beschreibung von Gesten, der – im Vergleich zu anderen textuellen Notationen – mit wenig strukturierenden Elementen auskommt und dadurch eher die Eigenschaften von Gesten in den Fokus nimmt. Nach Kammer (2014) basiert ihre Lösung auf der Modellierungssprache *MGrammer* von Microsoft.

Die Autoren verwenden drei grundlegende Blöcke, die zur Definition von Multi-Touch-Gesten verwendet werden: ein Name (*name*), der eine Geste eindeutig innerhalb der Anwendung identifiziert, eine beliebige Menge von Validierungsregeln (*validate*) mit einfachen Bedingungen und eine Rückgabe (*return*), die mehrere Informationen definieren kann, die die Anwendung über die Geste erhält. Für die Beschreibung der Geste ist der Block mit den Validierungsregeln bedeutsam: Er enthält die Beschreibung notwendiger Eigenschaften, die die Geste genau abgrenzen, als Liste von Name-Wert-Paaren. Im Kern besteht ein Validierungsblock aus einer Menge logischer Regeln, die aus einfachen Bedingungen zusammengesetzt werden können.

Abb. 6.3 zeigt ein Beispiel: eine Geste, bei der eine beliebige, geschlossene Form verwendet wird. Zeile 5 begrenzt die Geste dabei auf einen Finger und Zeile 6 definiert, dass die Form geschlossen werden muss. Dazu werden noch weitere Bedingungen spezifiziert, deren Interpretation allerdings weiterführendes Wissen über die Anwendung verlangt. In diesem Beispiel wird nur eine Teilgeste verwendet, aber es können beliebig viele der `validate`-Blöcke hinzugefügt werden. Jeder Block entspricht dann einer Teilgeste, die eindeutig benannt werden muss und mit anderen Teilgesten in Bezug stehen kann. So kann z. B. mithilfe des Schlüsselwortes `perpendicularTo` dargestellt werden, dass eine Teilgeste senkrecht zu einer anderen steht.

Abbildung 6.3: Regeln zur Beschreibung einer Geste in Form eines Lassos in GDL nach Khandkar & Maurer (2010b) (Quelle: ibd.)

```
1  name: Lasso
2
3  validate
4    Touch state: TouchUp
5    Touch limit: 1
6    Closed loop and
7    Touch path bounding box: 200x200..1000x1000 and
8    Touch path length: 600..100000 and
9    Enclosed area: 5000..1000000
10
11 return
12   Touch points
```

Im Gegensatz zu den meisten textuellen Notationen werden nur wenige syntaktische Regeln vorgegeben und wenige strukturierende Zeichen verwendet. Der Fokus auf die textuelle Beschreibung sorgt aber letztlich für die damit verbundenen Probleme. So ist z. B. unklar, wie komplexe Formen mit der Notation beschrieben werden können. Darüber hinaus müssen Teile der Notation mit technischen Details gefüllt werden, die für die eigentliche Spezifikation der Interaktion nicht relevant sind (z. B. die Angabe übergroßer Zahlen, wenn der Gültigkeitsbereich der Geste sehr groß ist).

Die zugrundeliegende technische Infrastruktur wird von de Souza Alcantara et al. (2012) zum *Intelligent Gesture Toolkit* erweitert. In *ProtoActive*, einem Tool zum Prototyping gestenbasierter Anwendungen, verwenden de Souza Alcantara et al. (2013) einen Editor, der darauf aufbaut.

6.2.4 GeForMT

GeForMT (Kammer, 2014; Kammer et al., 2010, 2012, 2015) – kurz für *Gesture Formalization for Multi-Touch* – schlägt eine Brücke von der Semiotik hin zur Gestensteuerung und der Spezifikation gestenbasierter Anwendungen. Aufbauend auf Konzepten der Semiotik entwickeln die Autoren eine DSL zur Beschreibung von Oberflächengesten, die gute Lesbarkeit und die direkte Integration in Quellcode im Fokus hat. Die Notation hat fünf Schlüsselelemente: die ausgeführte Bewegung, Anzahl und Form der Kontakte, zeitliche Relation, räumliche Relation und das fokussierte Objekt. Eine

definierte Menge atomarer Gesten, die über Operatoren zu komplexen Gesten kombiniert werden können, bildet die Grundlage. Kammer et al. (2011) erweitern das Konzept auch experimentell in den dreidimensionalen Raum.

GeForMT basiert auf einer Grammatik, die verschiedene Schlüsselwörter und Operatoren vorgibt, und sechs grundlegende Formen definiert: Linie, Kreis, Halbkreis, Berühren, Halten und Freiform. Diese einfachen Formen können zur Beschreibung einzelner Teilgesten verwendet und mithilfe der Operatoren zu komplexen Gesten zusammengesetzt werden. Dazu kommen noch verschiedene Schlüsselwörter, die die räumlichen Beziehungen zwischen Teilgesten (z. B. `CROSS`) oder andere Eigenschaften (z. B. `CONNECT_END`) beschreiben. Die Anzahl der Finger wird vor den eigentlichen Ausdruck geschrieben: `1F(LINE)` beschreibt eine Linie in eine beliebige Richtung mit einem Finger. Richtungen werden mithilfe einer Kompassrose angegeben; eine Bewegung nach oben ist entsprechend eine Bewegung nach Norden. `2F(LINE_E)` beschreibt somit eine Linie nach rechts mit zwei Fingern.

Die Beispiele zeigen auch gleich einen Vorteil von GeForMT: Die Ausdrücke lassen sich direkt in Text – z. B. Quellcode – integrieren und müssen nicht als separate Blöcke definiert werden. Außerdem zeigen Kammer et al. (2015), dass GeForMT gute Ergebnisse im Vergleich zu anderen textuellen Notationen mit Blick auf Prägnanz und Lesbarkeit erreicht. Allerdings wird in Kap. 8 und Kap. 9 gezeigt, dass graphische Elemente für die Beschreibung von Gesten vorteilhaft sind.

GeForMT wurde auch in anderen Arbeiten eingesetzt. Pöhland et al. (2013) erstellen aufbauend auf GeForMT ein semantisches Modell für Gestensteuerung in Form einer Ontologie. Hesenius et al. (2014a) beschreiben Gesten mit GeForMT zur Simulation von Gestensteuerung in automatisierten Akzeptanztests. Hesenius et al. (2014b) integrieren GeForMT in eine Definitionssprache, die zur Spezifikation multimodaler Anwendungen verwendet wird.

6.2.5 GestIT

GestIT (Spano et al., 2013) ermöglicht die Beschreibung von Oberflächengesten und räumlichen Gesten. Dabei werden grundlegende Elemente und Aktionen sowie eine Menge von Operatoren eingesetzt, die zu logischen Ausdrücken kombiniert werden können. Diese können wiederum mit Prädikaten erweitert werden, um zusätzliche Eigenschaften zu beschreiben. Das zugrundeliegende Modell basiert auf Petri-Netzen (Spano et al., 2012).

GestIT ermöglicht die Definition unterschiedlicher grundlegender Elemente (*Ground Terms*). Für Multi-Touch-Gesten sind dies z. B. `Start`, `Move` und `End`. Für andere Anwendungen können entsprechend passende grundlegende Elemente definiert werden, z. B. für Kinect-basierte Anwendungen die unterschiedlichen Körperteile, die Kinect erkennen kann. Diese können nun mit verschiedenen Operatoren (z. B. || für Parallelität oder >> für Sequenz) zu Gesten zusammengesetzt werden. Darüber hinaus können Prädikate (*Predicates*) hinzugefügt werden, um z. B. anzuzeigen, dass eine Hand offen oder

geschlossen ist. Auf die gleiche Art und Weise kann z. B. die Geschwindigkeit einer Bewegung definiert werden.

Das Beispiel in Abb. 6.4 zeigt, wie ein Pinch, also vom Namen her ein Zusammenziehen zweier Finger, in GestIT definiert wird. Zunächst wird im ersten Teil ($Start_1| = |Start_2$) definiert, dass zwei Finger hintereinander die Oberfläche berühren. Der Operator $|=|$ bedeutet hier, dass die Reihenfolge egal ist. Der nächste Teil der Geste wird mit dem Operator $>>$ angefügt, der eine Sequenz definiert. Auf den ersten Teil folgt also direkt der zweite ($Move_1^*||Move_2^*$), hier eine beliebige Bewegung der beiden Finger, die auf der Oberfläche liegen. Der Operator $\star$ besagt, dass diese beliebig oft wiederholt werden. Der letzte Teil ($End_1| = |End_2$) beschreibt analog zum ersten Teil das Ende der Geste. Der vorher eingesetzte Operator $[>$ muss verwendet werden, um die endlose Wiederholung des mittleren Teils zu beenden. Der Beschreibung fehlt offensichtlich eine Einschränkung: Im mittleren Teil können die Finger beliebig bewegt werden, jedoch definiert ein Pinch, dass sie sich aufeinander zubewegen müssen. Die Autoren erläutern jedoch in der Quelle, dass die Geste auch das Auseinanderziehen beinhalten soll und der zugehörige Event Handler innerhalb des Programms die Berechnung und Entscheidung übernimmt (Spano et al., 2013).

$$Pinch = (Start_1| = |Start_2) >> (Move_1^*||Move_2^*)[> (End_1| = |End_2)$$

Abbildung 6.4: Eine Pinch-Geste in GestIT (Quelle: Spano et al. (2013))

Die Autoren nehmen für sich in Anspruch, dass sie – im Vergleich zu anderen Notationen – gängige Probleme bei der Beschreibung von Gesten gelöst haben, insbesondere das Problem des *Spaghetti Codes* (vgl. Kap. 4). Das o. g. Beispiel widerspricht dem: Statt ein entsprechendes Prädikat in der Beschreibung der Geste zu definieren, wird das wesentliche Element der Richtung in den Quellcode der Anwendung verschoben. Der Hintergrund dieser Entscheidung kann an der jeweiligen Anwendung liegen, ist jedoch nicht auf den ersten Blick ersichtlich und bleibt somit unklar. Gleichzeitig leidet die Notation unter der Nähe zur Implementierung: Die Dopplungen, die durch die Definition z. B. zweier Finger in beliebiger Reihenfolge entstehen, sorgen für unnötige Häufungen, die die Lesbarkeit mutmaßlich einschränken. Darüber hinaus besteht hier die Gefahr, dass komplexe Gesten aus genau diesem Grunde schnell sehr unleserlich werden. Ein graphischer Editor oder sonstige Hilfestellungen existieren offenbar aktuell nicht.

6.2.6 Hand Gesture Notation

Choi et al. (2014) präsentieren eine Notation zur Beschreibung räumlicher Gesten, die den Fokus ausschließlich auf die Hände legt. Die Autoren leiten die Notation aus einer Taxonomie für die unterschiedlichen Posen und Bewegungen der Hände ab. Den einzelnen Elementen werden Codes zugewiesen, die nachher zur Beschreibung von Gesten kombiniert werden können.

Die Autoren unterscheiden zwischen Einhand- und Zweihandgesten, wobei letztere nochmal in Spiegelgesten (*Mirror Type*) und heterogene Gesten (*Hete-*

ro Type) unterteilt werden. Diese Unterscheidung findet sich in der symmetrischen Anordnung: Die Teilgesten der jeweiligen Hand sind bei Spiegelgesten spiegelverkehrt[2] und bei heterogenen Gesten unterschiedlich. Außerdem unterscheiden die Autoren statische und dynamische Gesten, meinen damit aber Posen und Bewegungen. Posen bestehen aus vier Elementen: Ort, Form und Ausrichtung der Hand sowie die Form des Armes (ausgedrückt als Winkel von Schulter und Ellbogen). Bewegungen werden in drei Typen unterteilt: eine Sequenz von Posen, Bewegungen der Hand ohne Änderung der Pose (dabei werden gradlinige und kurvige Bewegungen unterschieden), sowie eine Kombination aus beiden. Außerdem können Multiplizitäten definiert werden. Die unterschiedlichen Möglichkeiten werden mit Zahlencodes versehen, die dann in einer festgelegten Struktur verwendet werden, um Gesten zu beschreiben.

[2] Die Autoren sprechen lediglich von vertikaler Symmetrie (*vertical symmetry*) und es bleibt offen, wie und ob horizontale Symmetrie berücksichtigt wird.

$$31111;2222-222 => 56666;4422-333$$

Abbildung 6.5: Eine Geste, bei der die geöffnete Hand geschlossen wird, in Hand Gesture Notation (Quelle: Choi et al. (2014))

Stellvertretend für alle Optionen, die beschrieben werden können, soll in Abb. 6.5 lediglich auf die Positionierung der Finger während einer Geste eingegangen werden. Eigentlich muss zu jeder Geste noch der Typ, der Ort und weitere Informationen – bzw. die damit verbundenen Zahlencodes – angegeben werden. Die Geste wird in zwei Teilen beschrieben: Start- und Endposition, die mit dem Operator => getrennt werden. Der Operator definiert eine Bewegung vom Start zum Ende. Jede Position wird aus zwei weiteren Teilen beschrieben, die durch ein Semikolon getrennt werden: Der erste Teil definiert die Positionierung der Finger und der zweite ihre Relation zueinander. Die Startposition beschreibt eine offene Hand: Der Daumen – die erste Zahl – zeigt zur Seite (3, *Pointing (Side)*) und die anderen Finger nach oben (1, *Pointing (Up)*). Die Finger werden, beim Daumen startend, der Reihe nach benannt. Im zweiten Teil der Startposition wird definiert, dass sich alle Finger nicht berühren (2, *Separate*). Die Endposition hingegen beschreibt die geschlossene Hand: Der Daumen ist gebeugt (5, *Bend*) und die anderen Finger geschlossen (6, *Closed*). Der zweite Teil der Endposition besagt, dass der Daumen die ersten beiden Finger kreuzt (4, *Cross in Front*) und die nächsten beiden nicht berührt. Die anderen vier Finger sind gruppiert (3, *Group*).

Die Notation ist stark strukturiert und setzt – im Gegensatz zu allen anderen Ansätzen – nicht auf Schlüsselwörter, sondern auf Zahlencodes. Jede Zahl identifiziert in Abhängigkeit ihrer Position in der Notation eine bestimmte Pose, somit müssen Anwender konstant eine Übersetzung von Zahl zu Bedeutung vornehmen. Die Autoren berichten zwar von positiven Ergebnissen in ihrer Evaluation, verzichten aber auf einen Vergleich mit anderen Notation, weshalb die Ergebnisse zweifelhaft erscheinen.

6.2.7 LEG

Viana & Medeiros Santos (2017) stellen LEG – kurz für *Language for Editing Gestures* – vor und ermöglichen die Beschreibung räumlicher Gesten, die auf Microsoft Kinect basieren. Entsprechend orientieren die Autoren sich an

den Möglichkeiten, die dieser Sensor zur Verfügung stellt, und erlauben die Integration verschiedener Körperteile.

Die Autoren stellen eine Menge von Schlüsselwörtern zur Verfügung, mit denen Posen beschrieben werden können. Diese Posen werden sequentiell abgearbeitet; Gesten ergeben sich somit durch die Beschreibung von Start-, End- und Zwischenpositionen. LEG verwendet eine menschenlesbare, natürlichsprachlich anmutende Syntax, deren Aufbau sich aber grob an der Syntax der Programmiersprache Java orientiert.

Abbildung 6.6: Beschreibung einer Geste in Form eines Winkens in LEG (Quelle: Viana & Medeiros Santos (2017))

```
1 gesture Wave {
2   right hand above right elbow;
3   repeat 2 {
4     right hand to the left of right elbow;
5     right hand to the right of right elbow;
6   }
7 }
```

Abb. 6.6 zeigt, wie das Winken mit der rechten Hand beschrieben wird. Zeile 1 definiert eine neue Geste mit dem Schlüsselwort `gesture`, die den Namen *Wave* bekommt. Zeile 2 definiert dann die Startposition: Die rechte Hand wird über dem rechten Ellbogen platziert. Die Zeilen 3–6 definieren dann die Zwischenpositionen, die zweimal erreicht werden: Die rechte Hand befindet sich links vom Ellbogen und anschließend rechts vom Ellbogen. Nach der zweiten Durchführung ist hier die Endposition erreicht.

Interessant ist der Versuch, auf eine pseudonatürlichsprachliche Syntax zurückzugreifen, die jedoch durch einige Elemente, die ganz offensichtlich bekannten Programmiersprachen entlehnt sind (geschweifte Klammern, Semikolon am Ende einer Zeile usw.), unterbrochen wird. Das Beispiel in Abb. 6.6 zeigt auch direkt, wie mit diesem Versuch gebrochen wird: Die Anweisung in Zeile 3 ist nicht mehr gänzlich in natürlicher Sprache formuliert. Außerdem beschreibt LEG vorwiegend Posen und es bleibt unklar, wie Eigenschaften der Bewegung – z. B. die Geschwindigkeit – formuliert werden.

6.2.8 Midas

Scholliers et al. (2011) entwickelten Midas unter dem Aspekt, dass bis dahin bestehende Ansätze zur Erkennung von Multi-Touch-Gesten grundlegende Regeln des SE verletzen. Insbesondere kritisieren die Autoren die mangelnde Modularität und fehlende Trennung der Anliegen (Renaux et al. (2014) gehen diesbezüglich tiefer ins Detail). Sie schlagen einen regelbasierten Ansatz vor, bei dem eingehende Berührungsinformationen des Bildschirms in logische Fakten umwandelt werden. Diese werden dann mit einem Satz von Regeln zur Gestenerkennung abgeglichen. Die Definition der für eine Geste einzuhaltenden Regeln entspricht letztlich der Beschreibung der Geste.

Regeln in Midas werden als Paare von Voraussetzungen (*prerequisites*) und Konsequenzen (*consequences*) beschrieben. Die Voraussetzungen beschreiben

dabei die zu erkennende Geste und die Konsequenzen die daraufhin auszuführende Aktion. Die Autoren widersprechen mit dieser Vorgehensweise im gewissen Sinne ihrer ursprünglichen Argumentation: In der Beschreibung einer Geste können zwei Anliegen – nämlich die Geste und die daraus folgende Aktion – formuliert werden, wodurch die angestrebte Trennung der Anliegen nicht mehr gewährleistet ist.

Abb. 6.7 zeigt, wie ein Swipe mit einem Finger nach links in Midas formuliert werden kann. In Zeile 2 werden die notwendigen Voraussetzungen definiert: Mindestens fünf Eingaben (`min: 5`), die mit dem gleichen Finger (`same: id`) innerhalb von 500 ms (`within: 500`) durchgeführt wurden, sollen berücksichtigt werden. In Zeile 3 wird dann festgestellt, dass eine Bewegung nach links durchgeführt wird – was sich genau hinter `movingLeft` verbirgt, ist mutmaßlich an anderer Stelle definiert worden und wird hier wiederverwendet. Zeile 4 definiert eben dann die Konsequenz, nämlich was bei Erkennung der Geste passieren soll.

```
1 (defrule FlickLeft
2   ?eventList[] <- (ListOf (Cursor (same: id) (within: 500) (min: 5)))
3   (movingLeft ?eventList)
4   => (assert (FlickLeft (events ?eventList))))
```

Abbildung 6.7: Ein Swipe nach links formuliert in Midas (Quelle: Scholliers et al. (2011))

Midas setzt zur Beschreibung von Gesten auf eine Notation, die stark vom zugrundeliegenden Regelsystem abhängig ist, was sich z. B. an Schlüsselwörtern wie `defrule` bemerkbar macht, deren direkter Bezug zur Beschreibung von Gesten nicht offensichtlich scheint. Kammer (2014) berichtet, dass Midas außerdem einen hohen Anteil strukturierender Zeichen benötigt, die nicht direkt mit der beschriebenen Geste zusammenhängen. Außerdem wird innerhalb der Notation direkt auf technische Elemente zurückgegriffen, z. B. die Anzahl der mindestens zu verarbeitenden Eingaben im Beispiel in Abb. 6.7. Gleichzeitig ermöglicht Midas' unterliegende Architektur das einfache Hinzufügen neuer Eigenschaften, wodurch das Framework zumindest aus Sicht des Implementierers interessant ist.

Hoste et al. (2011) nutzen Midas innerhalb ihres Frameworks *Mudra*, das die Entwicklung multimodaler Anwendungen ermöglicht. Sie zeigen, wie die gleiche sprachliche Grundlage zur Definition unterschiedlicher Modalitäten in einer Notation eingesetzt werden kann.

6.2.9 Prepose

Figueiredo et al. (2016) stellen eine Notation vor, die für Anwendungen basierend auf Kinect entwickelt wurde, jedoch auch für andere Sensoren verwendet werden kann, die ähnlich funktionieren. Grundlegend folgt Prepose dem für räumliche Gestenbeschreibungen typischen Ansatz, verschiedene Körperposen zu definieren, deren sequentielle Ausführung dann die eigentliche Geste ergibt. Prepose definiert somit auch die Reihenfolge der zu erreichenden Po-

sen, aber nicht die eigentliche Bewegung dorthin. Neben den Händen können alle Körperteile in eine Geste integriert werden, die Kinect erkennen kann.

Prepose setzt auf eine natürlichsprachliche Syntax, sodass Gesten leicht verständlich beschrieben werden können. Das grundlegende Element einer Beschreibung ist die Pose (*Pose*). Eine Pose besteht aus einer Menge von fix definierten Positionen (*Transformation*) oder Intervallen (*Restriction*) einzelner Körperteile. Mehrere Posen können durch Definition einer Ausführungsreihenfolge (*Execution*) zu einer Geste zusammengefasst werden. Die Ausführungsreihenfolge ist dabei als Sequenz zu verstehen – eine genauere Definition der zeitlichen Relation (vgl. Kap. 5.2.1) ist nicht vorgesehen.

```
1  GESTURE fourth-position-en-avant:
2    POSE cross-legs-one-behind-the-other:
3      put your left ankle behind your right ankle,
4      put your left ankle to the right of your right ankle.
5      // do not connect your ankles
6
7    POSE high-arc-arms-to-right:
8      point your arms down,
9      rotate your right arm 70 degrees up,
10     rotate your left elbow 20 degrees to your left,
11     rotate your left wrist 25 degrees to your right.
12
13   EXECUTION:
14     // fourth-position-en-avant-composed
15     stand-straight,
16     point-feet-out,
17     stretch-legs,
18     cross-legs-one-behind-the-other,
19     high-arc-arms-to-right.
```

Abbildung 6.8: Beispiel einer Ballettfigur mit Prepose. Die Darstellung ist vereinfacht: In der Execution wird auf bereits vorher definierten Posen zurückgegriffen, die nicht dargestellt sind (Quelle: Figueiredo et al. (2016)).

Abb. 6.8 zeigt, wie eine Figur aus dem Ballett mit Prepose dargestellt wird. Die Beschreibung ist unvollständig, denn es werden nicht alle der unter `EXECUTION` definierten Posen beschrieben.[3] Zur Veranschaulichung des Prinzips ist die Beschreibung aber weitgehend ausreichend. Die erste Pose (Zeile 2–5) definiert, dass das linke Sprunggelenk schräg rechts hinter dem rechten steht. Die Beschreibung definiert nicht, dass dabei auch die Füße gedreht werden müssen. Dies wird erst in der Beschreibung unter `EXECUTION` deutlich, da eine bereits vorher definierte Pose *point-feet-out* eingenommen werden soll. Des Weiteren fügen sie als Kommentar (Zeile 5) hinzu, dass sich die Sprunggelenke nicht berühren sollen; der zugehörige Erkenner kann diese Unterscheidung wohl nicht treffen. Die zweite Pose (Zeile 7–11) beschreibt die Positionierung der Arme und beginnt mit der Anweisung, beide Arme nach unten zu nehmen. Anschließend soll der rechte Arm um 70° nach oben gedreht werden, jedoch geht aus der Beschreibung nicht hervor, in welche Richtung oder ob der Arm vor den Körper oder neben dem Körper gehalten werden soll. Des Weiteren definieren die Autoren zwei Posen in einer: Erst sollen beide Arme nach unten zeigen, anschließend soll dann aber doch einer gehoben werden. Inwiefern hier ein Widerspruch vorliegt oder es sich doch um die Definition einer Bewegung handelt, bleibt offen.

[3] Posen können unter Prepose auch anderweitig definiert und dann wiederverwendet werden.

Die Autoren integrieren die Unterstützung verschiedener Eigenschaften in die Beschreibung. So können alle von Kinect unterstützten Körperteile und

verschiedene Richtungen im dreidimensionalen Raum (vorne, hinten, oben, unten, rechts, links)[4] definiert werden. Außerdem kann beschrieben werden, dass Körperteile über-, unter-, vor- oder hintereinander platziert und für einen bestimmten Zeitraum gehalten werden müssen. Prepose erlaubt darüber hinaus die Definition von Wiederholungen.

[4] Wobei unklar bleibt, wie kombinierte Richtungen (z. B. vorne rechts) definiert werden.

Der natürlichsprachliche Ansatz birgt Potential für den Einsatz in crossfunktionalen Teams, bei denen auch Mitglieder ohne technische Kenntnisse Gesten definieren können. Gleichzeitig bleiben zahlreiche Fragen offen. Prepose basiert auf der Idee, Posen zu definieren, die nach und nach erreicht werden. Die Autoren beschreiben nicht, wie und ob zeitliche Relationen definiert werden können – in den angegebenen Beispielen sind einzelne Posen sequentiell zu erreichen, die aus der Positionierung verschiedener Körperteile bestehen. Wie genau diese wann an ihre Position kommen sollen, lässt sich nicht beschreiben. Des Weiteren erwähnen die Autoren, dass mit Prepose nicht alle denkbaren Gesten dargestellt werden können (begründet mit der Turing-Unvollständigkeit der Sprache), gehen jedoch auf die Einschränkungen nicht weiter ein.

6.2.10 Proton

Kin et al. (2012a,b) ermöglichen mit Proton die Beschreibung von Multi-Touch-Gesten mithilfe regulärer Ausdrücke (Regular Expressions (RegEx)). Diese können mit zusätzlichen Eigenschaften annotiert werden, um Gesten feiner zu spezifizieren.

Die Beschreibung von Gesten orientiert sich stark an der typischen technischen Verarbeitung. Es werden drei grundlegende Elemente definiert, die der Aktionen der Finger entsprechen: `D` (*Down*, der Finger berührt die Oberfläche); `M` (*Move*; der Finger bewegt sich auf der Oberfläche) und `U` (*Up*; der Finger verlässt die Oberfläche). Zwangsläufig startet eine Geste somit stets mit einem `D` und endet mit einem `U`. Über tiefgestellte Indizes können außerdem konkrete Finger identifiziert werden, wobei hier die Finger in der Reihenfolge der Berührung der Oberfläche nummeriert werden. Zusätzliche Attribute werden hochgestellt. Außerdem werden zwei Operatoren eingesetzt: Der Kleene-Stern ($*$) zur Definition von Wiederholung (typischerweise nur beim Element `M`) und ein Pipe ($|$) zur Definition von Entscheidungen.

$$D_1 M_1^E * D_2 (M_1^E * | M_2^E *) (U_1 M_2^E * U_2 | U_2 M_1^E * U_1)$$

Abbildung 6.9: Ein Swipe nach rechts mit zwei Fingern in Proton

Abb. 6.9 zeigt, wie ein Swipe mit zwei Fingern in Proton beschrieben wird. Zuerst berührt der erste Finger die Oberfläche (D_1) und beginnt sich nach rechts zu bewegen ($M_1^E *$). Proton setzt ebenfalls auf die Richtungen einer Kompassrose; das hochgestellte `E` steht für *East*. Anschließend setzt der zweite Finger auf (D_2) und es bewegen sich beide Finger weiter nach rechts ($M_1^E * | M_2^E *$). Der Abschluss des Ausdrucks besagt, dass entweder der erste Finger zuerst abgehoben wird ($U_1 M_2^E * U_2$) und dann der zweite oder andersrum ($U_2 M_1^E * U_1$). Da unklar ist, wie der Benutzer konkret vorgehen wird, müssen beide Alternativen spezifiziert werden.

Proton ermöglicht die flexible Definition unterschiedlicher Eigenschaften, wodurch viele unterschiedliche Gesten beschrieben werden können. Allerdings besteht – vor allem ohne die Definition zusätzlicher Attribute – die Gefahr, dass unterschiedliche Gesten zu gleichen Ausdrücken führen und nicht mehr differenziert werden können. Gleichzeitig wird in Abb. 6.9 auch deutlich, wie nah die Notation an der technischen Implementierung steht: Die Dopplung am Ende, die durch die Unklarheit entsteht, ob der erste oder zweite Finger zuerst die Oberfläche verlässt, sollte als technisches Detail wegabstrahiert werden, wenn die Notation zur Kommunikation der Geste eingesetzt werden soll. Darüber hinaus führen komplexe Gesten schnell zu komplizierten, unleserlichen RegEx, weshalb die Autoren einen graphischen Editor entwickelt haben, der bei der Definition von Gesten unterstützt. Dieser wird aber lediglich eingesetzt, um den Ausdruck übersichtlicher zu strukturieren.

6.3 Graphische Notationen

Im Gegensatz zu textuellen Ansätzen verwenden graphische Ansätze visuelle Elemente, um Gesten zu beschreiben. Text wird bei diesen Arbeiten allenfalls als unterstützendes Merkmal eingesetzt – die wesentlichen Informationen werden in den verwendeten visuellen Notationselementen kodiert. Damit stehen graphische Notationen aber letztlich vor dem gleichen Dilemma wie textuelle: Die Vielfalt der Möglichkeiten, die Gestensteuerung bietet, zwingt ihnen auf, bei Erweiterung immer neue Symbole einzuführen. Darüber hinaus ist zweifelhaft, ob eine graphische Darstellung wirklich für alle Gesteneigenschaften optimal ist oder ob es nicht einige gibt, für die textuelle Elemente sinnvoll sind.

6.3.1 EventHurdle

EventHurdle (Kim & Nam, 2013) ist ein Tool zur Modellierung von Gesten, das zum explorativen Prototyping eingesetzt werden soll. Die Autoren bemängeln, dass Designer durch technisch versierte Ingenieure unterstützt werden müssen, wenn sie nicht selber über die notwendigen Kenntnisse verfügen, um unterschiedliche Sensoren zu verwenden. Außerdem kritisieren die Autoren, dass der kreative Arbeitsfluss eines Designers durch die Auseinandersetzung mit technischen Details und die Notwendigkeit der Implementierung unterbrochen wird. Designer sollen daher in die Lage versetzt werden, in frühen Phasen der Anwendungsentwicklung ohne die Unterstützung spezialisierter Ingenieure oder technisches Vorwissen Gesten zu definieren und in ihren UI-Skizzen einsetzen zu können. Dabei unterstützt die Entwicklungsumgebung von EventHurdle verschiedene Sensoren und kann somit zur Modellierung von z. B. Gesten vor einer Kamera, Gesten mit einem Mobilgerät oder Gesten auf einem Multi-Touch-Bildschirm verwendet werden. EventHurdle ist auf einzügige Gesten beschränkt, unterstützt aber sowohl Oberflächengesten als auch räumliche Gesten.

Gesten werden unter EventHurdle im sog. *Interaction Workspace* definiert. Dieser stellt eine 2D-Oberfläche zur Verfügung, auf der sogenannte *Hurdles*

platziert werden können (in Abb. 6.10 die mit den Zahlen 1 und 2 versehenen Elemente). Hurdles definieren wichtige Punkte, die bei der Gestenausführung getroffen werden müssen. Die eigentliche Bewegung wird dann mit Pfeilen eingefügt. EventHurdle erlaubt dabei auch optionale Pfade, Entscheidungen und Wiederholungen. Die Beschreibung räumlicher Gesten wird trotz zweidimensionaler Oberfläche ermöglicht, indem unterschiedliche Interaction Workspaces verknüpft werden können. In einem Workspace können Bewegungspfade und Hurdles für jeweils zwei der drei Richtungen im Raum definiert und über eine zeitliche Relation zur Gesamtgeste kombiniert werden.

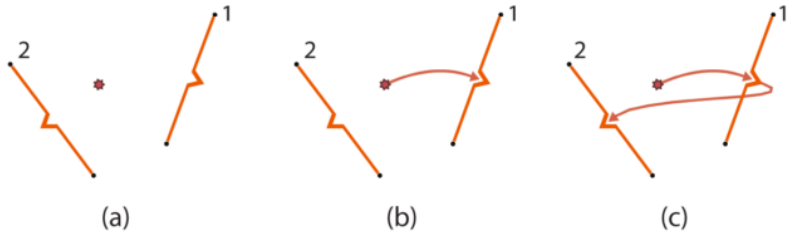


Abbildung 6.10: Eine einfache Geste in EventHurdle (Bildquelle: Kim & Nam (2013))

Abb. 6.10 zeigt, wie eine einfache Geste in EventHurdle aufgebaut wird:

- In (a) werden zwei Hurdles sowie in der Mitte der Startpunkt definiert
- In (b) wird durch Angabe des Bewegungspfades der erste Teil der Geste definiert, der vom Startpunkt bis zur ersten Hurdle reicht
- In (c) wird definiert, dass die Geste die erste Hurdle erneut kreuzen muss und sich dann nach links bewegt, bis Hurdle 2 erreicht ist

Das Beispiel definiert also eine Geste, bei der erst eine Bewegung nach rechts ausgeführt wird und anschließend eine weitere Bewegung nach links, die unterhalb der ersten Bewegung verläuft und über den Startpunkt hinausgeht. Zusätzlich zum Bewegungspfad kann die Zeit, die zwischen der Berührung zweier Hurdles vergehen darf, von 0 bis unendlich in Millisekunden festgelegt werden. Dadurch können schnelle und langsame Bewegungen sowie Haltepunkte modelliert werden.

Aus den definierten Hurdles und Bewegungspfaden werden letztlich Erkenner generiert, die direkt in Prototypen und UI-Skizzen integriert werden. EventHurdle ist jedoch auf einzügige Gesten begrenzt und unterstützt weder mehrzügige noch Multi-Touch-Gesten, kann aber für eine Vielzahl unterschiedlicher Sensoren eingesetzt werden. Somit können prototypische UIs für zahlreiche Anwendungen und Interaktionsmöglichkeiten schnell erstellt werden. Zwar können verschiedene Pfade und einfache zeitliche Informationen definiert werden, z. B. zur Bestimmung der Geschwindigkeit zwischen zwei Hurdles, eine genaue Spezifikation der meisten in Kap. 5 definierten Eigenschaften ist ohne Erweiterung aber nicht möglich.

Kim et al. (2016) bauen auf EventHurdle auf und demonstrieren, wie räumliche Gesten mit einer Anwendung für mobile Geräte definiert werden können.

Sie erweitern das bestehende Konzept dabei um die Möglichkeit, Gesten direkt mit dem Gerät aufzunehmen und mit Hurdles zu verfeinern.

6.3.2 GISMO

Romuald & Mens (2015) stellen GISMO vor, eine graphische Domain Specific Modelling Language (DSML) zur Spezifikation ausführbarer Modelle gestenbasierter Anwendungen. Die Autoren bauen dabei auf eigenen Vorarbeiten (Romuald, 2013; Romuald et al., 2013) auf und demonstrieren, wie ganze Interaktionsszenarien in einer Anwendung modelliert und als Prototyp automatisch ausgeführt werden können. Aus den Modellen kann auch die Struktur einer Anwendung exportiert werden, die von den Programmierern dann nur noch mit der Programmlogik ausgefüllt werden muss.

Die wesentlichen Elemente in GISMO sind Gestenobjekte (*Gesture Objects*), die die Gesten darstellen, und Zustandsobjekte (*State Objects*), die die Zustände der Anwendung repräsentieren. Ein Gestenobjekt definiert sowohl die Geste als solche als auch zusätzliche Eigenschaften. Die Autoren geben hierzu einen Satz vor: Move, Drag, Open, Close, Tilt, Zoom, Rotate. Es können noch Variablen hinzugefügt werden, die für die weitere Verarbeitung in der Anwendung relevant sind. Diese entsprechen einigen der Eigenschaften, die in Kap. 5 vorgestellt wurden und sind teilweise auf einige der Gestentypen beschränkt. Transitionen können außerdem mit Bedingungen (*Conditions*) ausgestattet werden, damit bestimmte Werte der Eigenschaften eingehalten werden. GISMO erlaubt auch Zustandswechsel während der Ausführung einer Geste und ermöglicht so die Integration von Online-Gesten.

Abbildung 6.11: Ein Gestenobjekt in GISMO, das das Drehen beider Hände im Uhrzeigersinn (analog zu einem Lenkrad) darstellt (Bildquelle: Romuald & Mens (2015)).

GISMO bietet eine Menge graphischer Elemente, um Gesten, Zustandsänderungen und ihre Beziehungen untereinander darzustellen. Gesten und Zustände werden über Transitionen (*Transitions*) miteinander verbunden. Auf eine Geste muss zwingend ein Zustand folgen und umgekehrt. Zustände können mehrere Ein- und Ausgänge haben und somit mehrere Gesten anbinden, Gesten jedoch haben genau einen Eingang und genau einen Ausgang, können also nur von einem Zustand aus ausgeführt werden und in einen anderen münden. Zusätzlich kann ein Bild hinzugefügt werden, um die Identifikation zu erleichtern. Ein Zustand enthält keine Anwendungslogik, sondern ist lediglich ein Platzhalter, der später durch den Entwickler ausprogrammiert werden muss. GISMO erlaubt allerdings die Definition globaler Variablen (*Global Variables*), um Informationen zu speichern und in Zuständen wiederzuverwenden.

Im oberen Abschnitt des Gestenobjekts in Abb. 6.11 wird das Werkzeug definiert, mit dem die Geste ausgeführt wird. Die parallele Verwendung mehrerer Werkzeuge wird durch die Angabe mehrerer Regionen (*Regions*) dargestellt, die jede ein eigenes Werkzeug enthalten. Jede Region kann optional noch Einschränkungen z. B. in der Bewegungsrichtung enthalten, die dann durch Pfeile dargestellt werden. Die in Abb. 6.11 dargestellte Geste entspricht den Autoren zufolge einer Rotation mit beiden Händen im Uhrzeigersinn analog zum Drehen eines Lenkrads, ist jedoch mehrdeutig. Unklar bleibt, wie Drehungen der Hände am Platz oder parallel zueinander dargestellt werden.

GISMO unterstützt aktuell nur einfache Gesten, die parallelisiert werden können, jedoch keine Sequenzen aus Teilgesten, wie sie zur Darstellung von Gesten mit mehreren Zügen notwendig sind. Des Weiteren müssen Gesten- und Zustandsobjekte stets im Wechsel definiert werden, sodass Kombo-Gesten (vgl. Kap. 5.2.1) sowie Multiplizitäten (vgl. Kap. 5.1.9) nicht dargestellt werden können. Außerdem müssen grundlegende Gesten als vordefinierte ICOs[5] vorliegen; die Verwendung individueller Gesten durch Dritte wird so erschwert. Besser wäre es, wenn diese automatisch aus den Beschreibungen generiert werden könnten. Außerdem ist unklar, warum Gestenobjekte nur genau von einem Zustandsobjekt heraus angesprochen werden können. Die Wiederverwendung von Gesten wird so verhindert.

[5] Eine für interaktive Anwendungen spezialisierte Form von Petri-Netzen (Navarre et al., 2009).

6.3.3 Monox

Monox (Ganhör & Spreicer, 2014) – kurz für *Mobile Notation Extensible* – ist eine graphische Modellierungssprache für Multi-Touch-Gesten auf mobilen Geräten. Die Autoren haben dabei zwei Ziele im Sinn: Zum einen soll Monox die Messung der UX ermöglichen, wobei die Autoren nicht im Detail darauf eingehen, wie das vonstatten gehen soll, zum anderen ein Kommunikationsmedium zwischen den Stakeholdern sein. Sie geben dazu verschiedene Elemente vor, die den gängigen Aktionen auf mobilen Geräten sowie der notwendigen Hardware entsprechen.

Abbildung 6.12: Darstellung von Berührungen in Monox (Bildquelle: Ganhör & Spreicer (2014))

Einfache Berührungen – von Ganhör & Spreicer (2014) als *Taps* bezeichnet – werden in Monox durch Kreise dargestellt. Jeder Finger auf dem Display wird durch einen eigenen Kreis symbolisiert; für mehrere Finger müssen entsprechend viele Kreise verwendet werden. Die Autoren unterscheiden dabei verschiedene Arten von Taps, wie in Abb. 6.12 dargestellt: (a) zeigt einen einfachen, kurzen Tap; (b) einen doppelten Tap; (c) einen Long Tap, also eine Berührung, die länger gehalten wird. Es kann jedoch nicht definiert werden, wie lange die Berührung gehalten werden soll und wie lang die Pause zwischen zwei Berührungen sein soll. Mehrfache Taps werden durch Schachtelung aus einfachen Taps konstruiert, wobei die Ausführung von außen nach innen passiert: Während (b) in Abb. 6.12 einen doppelten Tap bestehend aus zwei einfachen Taps zeigt, zeigen (d) und (e) jeweils Kombinationen von kurzen und langen Taps in unterschiedlicher Reihenfolge.[6] Die Darstellungen (f), (g) und (h) in Abb. 6.12 zeigen, wie der Ort der Berührung unter Monox eingegrenzt werden kann. Dazu wird das jeweilige Tap-Element mit einem Viereck umzeichnet. Zusätzliche Linien beschreiben, dass auch außerhalb der markierten Fläche Aktionen ausgelöst werden können, die ungewollt sein können. In (h) kann z. B. auch links und rechts der markierten Fläche ein Tap ausgeführt werden.

[6] Wobei nicht klar wird, wie zwei lange Taps modelliert werden sollen, denn es müssten zwei schwarze Kreise geschachtelt werden.

Bewegungen der Finger auf dem Display werden als Pfeile dargestellt, wie Abb. 6.13 veranschaulicht. Dabei unterscheiden Ganhör & Spreicer (2014) zwischen langsamen und schnellen Bewegungen, die mit einer ((a) und (d)) bzw. zwei Pfeilspitzen ((b) und (c)) dargestellt werden. Die Pfeilspitzen symbolisieren auch die Bewegungsrichtung – befinden sich an beiden Seiten Pfeilspitzen, dann kann die Bewegung in beide Richtungen ausgeführt werden.

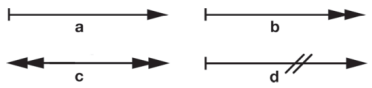


Abbildung 6.13: Darstellung von Bewegungen in Monox (Bildquelle: Ganhör & Spreicer (2014))

Die Verwendung mehrerer Finger wird wie in (d) durch kreuzende Linien dargestellt. Jede Linie entspricht einem Finger, somit wird im Beispiel eine langsame Bewegung nach rechts mit zwei Fingern veranschaulicht. Multi-Touch-Gesten, bei denen die Finger nicht die gleiche Aktion durchführen, werden mit mehreren Linien modelliert. Start- und Endzonen können mit Rechtecken definiert werden. Die Startzone verwendet dabei spitze Ecken, die Endzone abgerundete.

Die Autoren evaluieren Monox in drei Studien und erweitern die grundlegende Syntax basierend auf den Ergebnissen. Beispielsweise fügen sie noch die Möglichkeit ein, Sensordaten von Near Field Communication (NFC) oder Global Positioning System (GPS) zu notieren oder explizite Hardware-Buttons zu modellieren. Des Weiteren können modellierte Interaktionen zu Makros zusammengefasst und wiederverwendet werden. Die Studien zeigen außerdem ein grundlegendes Interesse der Probanden an den Zielen von Monox. Sie heben zwar insbesondere hervor, dass mit Monox eindeutige Interaktionen für Multi-Touch-Geräte definiert werden können, jedoch ist unklar, wie Multi-Touch- und Multi-Stroke-Gesten voneinander unterschieden werden. Außerdem bemängeln einige Probanden, dass sich Monox nur auf die Interaktion fokussiert und wünschen eine Integration in UI-Skizzen. Des Weiteren bietet Monox keine Möglichkeit, die genauen Details einzelner Eigenschaften wie der Dauer oder der Geschwindigkeit einer Geste zu definieren.

Enström et al. (2015) haben auf Monox aufbauend eine Notation speziell zur Beschreibung von Gesten auf touch-basierten Instrumenten entwickelt. Sie fügen Notationselemente hinzu, die für die Verwendung in gestenbasierten Musikinstrumenten optimiert sind. Außerdem integrieren sie Elemente aus der gängigen westlichen Musiknotation. Die Notation enthält interessante Elemente und zeigt, wie unterschiedliche graphische Elemente eingesetzt werden können: Zum Beispiel ist die Darstellung von Beschleunigung als dicker werdender Pfeil eine Neuerung gegenüber Monox. Gleichzeitig werden aber andere wesentliche Eigenschaften wie z. B. der Ort der Ausführung oder der Start- und Endpunkt nicht weiter erläutert, obwohl diese Elemente in Monox vorhanden sind. Auch ist nicht klar, wie genau einzügige und mehrzügige Gesten unterschieden werden. Die Ergebnisse der Evaluierung der Autoren lassen auch Zweifel an der Eindeutigkeit der definierten Gesten aufkommen. Außerdem bleibt weitgehend unklar, warum die Autoren zwar explizit betonen, dass sie sich an Monox orientieren, dann jedoch entscheiden, lediglich die Grundidee von Monox zu übernehmen und eigene Notationselemente zu entwerfen, statt Monox einfach um passende Elemente zu erweitern.

6.4 Hybride Notationen

Im Gegensatz zu textuellen und graphischen Notationen, die streng einem Paradigma folgen, mischen hybride Notationen unterschiedliche Darstellungsformen. Dadurch können unterschiedliche Eigenschaften auf die jeweils optimale Art und Weise dargestellt werden. Allerdings sind in den letzten Jahren nur wenige hybride Ansätze zur Beschreibung von Gesten entwickelt worden. Es ist unklar, warum die Möglichkeiten, die sie bieten, kaum Beach-

tung finden, obwohl es Hinweise gibt, dass sie das Verständnis von Gesten verbessern (Fothergill et al., 2012).

Im Rahmen dieser Arbeit konnte nur eine Notation identifiziert werden, der ein hybrider Ansatz zugrunde liegt. In Gesture Script verwenden Lü et al. (2014) einen musterbasierten Ansatz (vgl. Kap. 6.1) zur Beschreibung von Oberflächengesten, fügen aber die Möglichkeit hinzu, Teilgesten zu definieren, diese in kurzen Scripts zu referenzieren und so zusammengesetzte Gesten zu beschreiben. Aus der Kombination von Muster und Scripts können dann spezialisierte Gestenerkenner generiert werden.

In Gesture Script sind drei Schritte nötig, um Gesten zu definieren. Im ersten Schritt müssen Entwickler – analog zu musterbasierten Erkennern – die Geste aufzeichnen. Anschließend kann von Gesture Script bereits ein Erkenner generiert werden. Dieser wird verwendet, um alle definierten Gesten und die Erkennungsrate zu prüfen. Ist die Erkennungsrate zu schlecht, kann der Entwickler im letzten Schritt die Geste konkreter definieren. Dazu bestimmt er erst Teilgesten und kann dann in einer einfachen Script-Sprache das Verhalten weiter spezifizieren. Die Scriptsprache unterstützt drei Funktionen:

- `draw(x)` zeichnet die als x definierte Teilgeste
- `repeat` ermöglicht die beliebige Wiederholung einer Teilgeste
- `rotate` ermöglicht die Rotation der Teilgeste

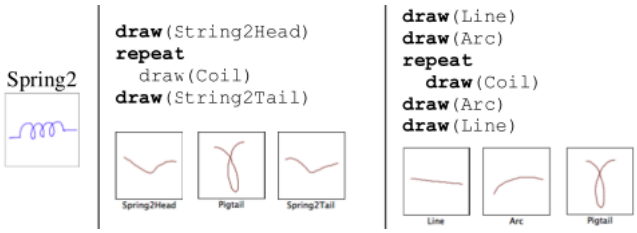


Abbildung 6.14: Verschiedene Möglichkeiten, um Gesten mit Gesture Script zu definieren (Bildquelle: Lü et al. (2014))

Abb. 6.14 zeigt ein Beispiel, wie eine Geste mit Gesture Script definiert wird. Beide Fälle beschreiben die ganz links gezeigte Vorlage und zeigen, wie Gesten auf unterschiedliche Art und Weise in Teilgesten zerlegt werden können. Die Funktion `draw` zeichnet zunächst die jeweils angegebene Teilgeste. Über das Schlüsselwort `repeat` wird anschließend die eingefasste Aktion – hier wieder `draw` – beliebig oft wiederholt.

Gesture Script ist ein Ansatz, der die unterschiedlichen Stärken der bisher vorgestellten Varianten vereint. Zum einen kann durch den musterbasierten Ansatz die Form einer Geste einfach und weitgehend grenzenlos definiert werden, zum anderen können weitere Eigenschaften durch die Scriptsprache hinzugefügt werden. Gleichzeitig ist ebendiese Scriptsprache nicht sehr mächtig und kann ausschließlich zum Zeichnen definierter Muster, der Bestimmung von Wiederholungen sowie des Drehwinkels verwendet werden. Außerdem ist Gesture Script auf einzügige Gesten beschränkt und kann somit nur für eine bestimmte Klasse von Oberflächengesten eingesetzt werden.

6.5 Sonstige Notationen

Die Spezifikation von Gesten und Posen ist nicht nur für Gestensteuerung, sondern auch für andere Fachbereiche interessant. An dieser Stelle sollen zwei Bereiche näher betrachtet werden, in denen Gesten und Posen eine zentrale Rolle spielen und deren Konzepte in verschiedenen Arbeiten mit direktem Bezug zu HCI eingesetzt wurden: die Beschreibung und Synthetisierung von Gebärdensprache (Kap. 6.5.1) und Notationen zur Dokumentation von Tänzen (Kap. 6.5.2). Beiden ist gemein, dass sie teils überschneidende, teils andere Anforderungen an die Beschreibung von Bewegung und Posen stellen als Gestensteuerung.

6.5.1 Beschreibung und Synthetisierung von Gebärdensprache

Unterschiedliche Arbeiten (z. B. Lebourque & Gibet (1999), Losson & Vannobel (1999) und Grieve-Smith (2002); Choi et al. (2014) geben einen Überblick) fokussieren Notationen, die zur Konfiguration von Systemen verwendet werden, mit denen Menschen und Computer mittels Gebärdensprache kommunizieren können. Insbesondere mit Blick auf Barrierefreiheit[7] ist Gebärdensprache als Alternative zur Sprachsteuerung interessant, um Menschen, die z. B. gehörlos sind, einbeziehen zu können. Ein wiederkehrender Ansatz ist, virtuelle Avatare zu konfigurieren, die mit Gebärdensprache angesprochen werden können bzw. mit Gebärdensprache antworten. Ein anderer Anwendungsbereich ist die Kommunikation zwischen Menschen und Robotern.

[7] Im Kontext von HCI auch als *Accessibility* oder kurz *A11y* bezeichnet.

In Abb. 6.15 ist zu sehen, wie das Wort *Geben* im Ansatz von Losson & Vannobel (1999) umgesetzt wird. Die textuelle Beschreibung verdeutlicht auch, dass Gebärdensprache auf mehr Regeln zurückgreifen muss als Gestensteuerung. So wird z. B. gleich in der ersten Zeile der Kontext des Dialogs festgelegt. Im Folgenden werden aber auch Name-Wert-Paare verwendet, um die Eigenschaften der Gebärde genauer zu definieren. Die Gebärde besteht darüber hinaus aus den drei rechts durch einen Avatar dargestellten Elemente, die sich so auch in der Beschreibung finden.

Gebärdensprache hat jedoch als Sprache, die zur menschlichen Kommunikation eingesetzt wird, z. T. andere Charakteristiken als Gestensteuerung. Dorau (2011, S. 26) bringt es auf den Punkt und grenzt Gebärdensprache

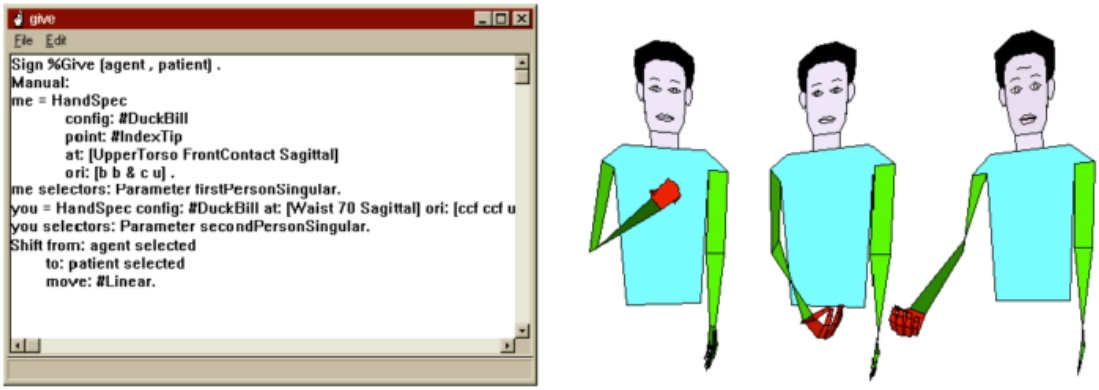


Abbildung 6.15: Das Wort *Geben* in Gebärdensprache (Bildquelle: Losson & Vannobel (1999))

von Gestensteuerung ab: Gebärdensprache ermöglicht es, ohne gesprochenes Wort zu kommunizieren, wobei Kommunikation hier als Dialog verstanden werden muss; Gestensteuerung hingegen ist kein Dialog, sondern eine einfache Kommandostruktur. Die Möglichkeiten von Gebärdensprache übersteigen also die gestenbasierter Anwendungen, da neben der reinen Formulierung von Befehlen auch grammatikalische Strukturen berücksichtigt werden müssen. So wird in Abb. 6.15 auch in Zeile 10 explizit die zweite Person Singular definiert. Es ist davon auszugehen, dass Notationen zur Beschreibung von Gebärdensprache ebendiese Regeln berücksichtigen müssen, die für Gestensteuerung aber nicht relevant sind. Daher können diese Notationen zur Spezifikation von Gesten nur bedingt Pate stehen.

6.5.2 Movement Notations

In den vergangenen Jahrhunderten wurden verschiedene Notationen zur Dokumentation von Tänzen und Choreographien entwickelt. Hutchinson Guest (1998) vergleicht die sog. *Movement Notations* aus ihrer Sicht als Choreographin und zeigt die Bandbreite der unterschiedlichen Ansätze, die im Laufe der Zeit entwickelt wurden. Im Folgenden soll aus der Fülle der Ansätze insbesondere auf *Labanotation* (Hutchinson Guest, 2005) eingegangen werden, da diese im Bereich HCI von unterschiedlichen Forschungsgruppen zur Beschreibung von Gesten eingesetzt wird (z. B. von Altakrouri & Schrader (2014) und Kordts et al. (2015)). Labanotation ermöglicht grundsätzlich die Beschreibung des ganzen Körpers, jedoch fokussiert eine Teilmenge explizit Hände und Finger (Hutchinson Guest & Kolff, 2002) und ist somit für die Beschreibung von Gestensteuerung besonders interessant.

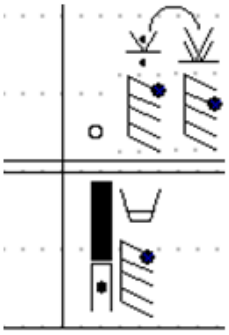


Abbildung 6.16: Ein Pinch mit der rechten Hand in Labanotation

Abb. 6.16 zeigt, wie ein Pinch mit der rechten Hand in Labanotation dargestellt wird. Der untere Teil beschreibt die Startposition und der obere die eigentliche Geste. Im Startteil wird definiert, dass der Daumen der Hand abgespreizt ist: Das fünfgliedrigere Symbol bezeichnet die Hand und der Daumen wird durch den schwarzen Punkt am obersten Glied bestimmt. Im oberen Teil werden zwei Finger der rechten Hand verwendet: Daumen und Zeigefinger, die durch die beiden fünfgliedrigen Symbole oben dargestellt werden und an unterschiedlichen Stellen Punkte haben. Die Verbindung der beiden oberen Symbole stellt die schließende Bewegung dar.

Labanotation ist eine mächtige Notation, mit der alle möglichen Körperbewegungen beschrieben werden können, und die sehr feingliedrige Definitionen ermöglicht. Gleichzeitig liegt in dieser Mächtigkeit auch eine der Schwächen: Die Notation verwendet eine große Menge unterschiedlicher Symbole. Nutzer sind gezwungen, sich ebendiese Symbole erst einzuprägen bzw. während der Arbeit beständig nachzuschlagen. Darüber hinaus fokussieren auch Movement Notations einen anderen Bereich als Gestensteuerung und müssen sich letztlich den hierfür notwendigen Anforderungen beugen – in diesem Falle die Verbindung zur Musik und den damit einhergehenden Unterscheidungen für Rhythmik, Betonung und dergleichen. Für die Definition von Gesten, die in vielen Fällen mutmaßlich sehr viel einfacher gestaltet sind, erscheinen Movement Notations daher zu umfangreich.

Teil III

Lösungsansatz und Validierung

7 GestureCards – Eine hybride Notation zur Definition von Gesten[1]

In den vorhergehenden Kapiteln wurde gezeigt, wie sich die Interaktion zwischen Mensch und Maschine in den letzten Jahren verändert hat und dass insbesondere Gestensteuerung eine zentrale Rolle einnimmt. Konsequenterweise müssen Gesten früh in der Software-Entwicklung berücksichtigt und definiert werden, um die Implementierung der Interaktionskomponenten zu ermöglichen. Die Phasen, in denen das Interaktionskonzept einer Anwendung festgelegt wird, sind jedoch heutzutage zumeist nicht von Implementierungs-, sondern von Benutzungsüberlegungen geprägt. Ebenso wie durch Storyboards und UI-Prototypen die Benutzeroberfläche definiert wird und die wesentliche Elemente wie die Positionierung von Buttons und anderen Inhalten beschrieben werden, muss auch die Steuerung der Anwendung durch Gesten an dieser Stelle zur weiteren Verarbeitung definiert und dokumentiert werden.

Altakrouri & Schrader (2015) beschreiben, dass bei der Spezifikation von Gesten insbesondere auf textuelle Beschreibung und bildliche Darstellung zurückgegriffen wird. In Kap. 6 wurde beleuchtet, wie mit verschiedenen Lösungsansätzen, die auf unterschiedlichen Grundlagen aufbauen, Gesten und Posen definiert werden. Die Betrachtung dieser Arbeiten zeigt, dass sowohl textuelle Beschreibung als auch bildliche Darstellung verschiedene Stärken und Schwächen haben, die z. T. auf die Zielstellung der Arbeiten zurückzuführen sind: Statt einer Definition von Gesten zur Spezifikation und Kommunikation, sollen sie in erster Linie Gestenerkenner konfigurieren.

Mit GestureCards soll ein anderer Weg gegangen und auf einen hybriden Ansatz zurückgegriffen werden, der die Stärken textueller Beschreibung und bildlicher Darstellung kombiniert. Im Gegensatz zu Gesture Script (vgl. Kap. 6.4), das basierend auf individuell definierbaren Teilgesten eine Script-Sprache zur Programmierung einfacher Gestenerkenner bietet, sollen Gesten mit GestureCards spezifiziert und für die Kommunikation zwischen unterschiedlichen Stakeholdern innerhalb eines Entwicklungsprojektes definiert werden können.

Im Folgenden werden die Grundlagen von GestureCards eingeführt (Kap. 7.1). Es wird zunächst gezeigt, wie einfache Gesten mit GestureCards spezifiziert werden können (Kap. 7.1.1). Dann wird gezeigt, wie zusätzliche Eigenschaften zur genaueren Spezifikation definiert und ihre Wertebereiche mit Operatoren eingeschränkt werden (Kap. 7.1.2). Daraufhin wird beschrieben, wie zusammengesetzte Gesten mit GestureCards spezifiziert (Kap. 7.1.3) und andere definierte Gesten referenziert werden (Kap. 7.1.4). Anschließend wird die Integration virtueller Objekte veranschaulicht (Kap. 7.1.5) und gezeigt, wie definierte Gesten wiederverwendet (Kap. 7.1.6) werden können. Nach Einführung dieser Grundlagen wird im Detail auf die unterschiedlichen Eigenschaften und ihre möglichen Werte eingegangen, mit denen die Gestenspezifikation verfeinert werden kann (Kap. 7.2).

[1] Eine Zusammenfassung dieses Abschnitts wurde unter Hesenius & Gruhn (2016) publiziert.

7.1 Grundlagen

GestureCards erlauben die Spezifikation zweier Arten von Gesten: einfache (auch atomare) Gesten sowie zusammengesetzte (auch komponierte oder komplexe) Gesten. Der wesentliche Unterschied ist die Granularität: Während einfache Gesten aus genau einem Element bestehen, bilden mehrere atomare Teilgesten einen Verbund und somit eine zusammengesetzte Geste. Sowohl atomare Teilgesten als auch die Gesamtgeste können mit individuellen Eigenschaften genauer ausspezifiziert werden, wobei die Summe aller Teilgesten letztlich die Eigenschaften der Gesamtgeste bestimmt.

7.1.1 Spezifikation einfacher Gesten

GestureCards bestehen aus vier Sektionen, die unterschiedliche Informationen enthalten und verschiedenen Regeln folgen. Abb. 7.1 zeigt den grundlegenden Aufbau. Die erste Sektion beinhaltet den Namen einer Geste und die zweite eine graphische Repräsentation, die insbesondere die Form der Geste zeigt. Die dritte Sektion definiert Eigenschaften, die die Geste genauer spezifizieren und die vierte eine Menge von Eigenschaften, die für die weitere Verarbeitung in der Anwendung relevant sind. Die Sektionen für den Namen und die graphische Repräsentation müssen ausgefüllt werden; die Sektionen für Eigenschaften und Rückgaben sind optional.

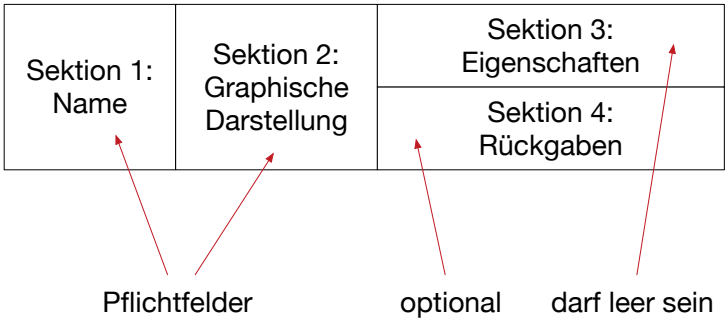


Abbildung 7.1: Grundlegender Aufbau einer GestureCard mit den vier verwendeten Sektionen

Die erste Sektion enthält den Namen der Geste. Der Name sollte innerhalb aller definierten Gesten eindeutig sein, denn er kann zur Referenzierung der Geste verwendet werden. Ansonsten kann die Namensgebung weitgehend frei erfolgen; eine Beschränkung der verwendeten Zeichen oder ihrer Anzahl ist nicht vorgesehen. Einzig die Schlüsselwörter, mit denen Eigenschaften benannt werden, können nicht als Name verwendet werden. Namen für Gesten sollten jedoch sprechend sein, analog zur Benennung von Variablen in Programmiersprachen, um das Verständnis und die Zuordnung zu erleichtern. Dabei kann auf die Geste selbst Bezug genommen werden und z. B. die Form als Namensgeber dienen. Alternativ kann z. B. ein Hinweis auf die verknüpfte Aktion verwendet werden. Eine Geste könnte auch wie die zur Geste gehörende User Story benannt werden, um eine einfache Verknüpfung zur Funktionalität und zu anderen Artefakten herzustellen, die im Entwicklungsprozess verwendet werden.

Die zweite Sektion enthält eine graphische Darstellung der Geste bzw. der Eigenschaften, die einfacher gezeichnet als textuell beschrieben werden können. Typischerweise werden hier die Form der Geste sowie die Bewegungsrichtung der Gestenzüge definiert. Während einfache Formen wie Linien kurz und knapp textuell beschrieben werden können (vgl. z. B. die Umsetzung in GeForMT, Kap. 6.2.4), sind komplizierte Formen einfacher graphisch darzustellen, weswegen template-basierte (vgl. Kap. 6.1) und graphische (vgl. Kap. 6.3) Ansätze an dieser Stelle Vorteile bieten.

Die dritte Sektion enthält eine Menge von Eigenschaften, die weitere Details über die Geste definieren. Diese stellen Bedingungen dar, die später in der Implementierung des verwendeten Gestenerkenners berücksichtigt werden müssen. Eine Geste soll also nur erkannt werden, wenn alle definierten Bedingungen erfüllt sind. Die Sektion kann leer sein, wenn die in Sektion 2 angegebene graphische Darstellung alle notwendigen Informationen enthält, muss aber angezeichnet werden – in diesem Falle als leerer Kasten. Ein simples Beispiel für diese Situation ist, dass lediglich die Form einer Geste von Bedeutung ist und keine weiteren Eigenschaften definiert werden müssen.

Die vierte Sektion enthält eine Liste von Eigenschaften, die für die weitere Verarbeitung in der Anwendung relevant sind. Um auf einen Begriff aus der Programmierung zurückzugreifen: Die hier angegebenen Eigenschaften entsprechen einem *Rückgabewert* der Geste. Dies kann nur eine einzige Eigenschaft sein, z. B. der Ort, an dem eine Oberfläche berührt wird, oder auch mehrere, wenn z. B. neben dem Ort noch der auf die Oberfläche ausgeübte Druck relevant ist. Mit einigen Ausnahmen kann jede Eigenschaft auch als Rückgabe verwendet werden. Welche Informationen allerdings genau über die Rückgabe für die Anwendung relevant sind, ist vom jeweiligen Anwendungsfall abhängig und wird im Folgenden nicht definiert, sondern muss zwischen den Projektbeteiligten vereinbart werden. Anzunehmen ist, dass alle Informationen, die zur Spezifikation der Geste verwendet werden, auch als Rückgabe denkbar sind. Die Sektion kann ausgelassen werden, wenn kein Ergebnis der Gestenausführung für die Anwendung benötigt wird, sondern nur die Erkennung als solche relevant ist. Dies kann der Fall sein, wenn eine Geste lediglich ein Short-Cut für ein Kommando ist, ähnlich einem Tastaturkürzel.

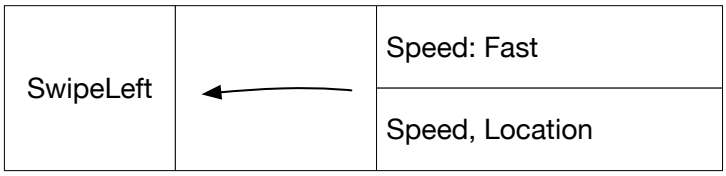


Abbildung 7.2: Beispiel einer GestureCard für eine einfache Geste: eine Linie nach links, die schnell ausgeführt werden muss

Abb. 7.2 zeigt ein Beispiel, wie eine Geste mit GestureCards definiert werden kann. Die dargestellte Geste ist ein Swipe – also eine schnelle Bewegung – nach links. Die erste Sektion ganz links benennt die Geste als *SwipeLeft*. Die zweite Sektion zeigt die Form der Geste – eine Linie – samt Bewegungsrichtung, die durch die Pfeilspitze linkerhand verdeutlicht wird. In der dritten Sektion rechts oben wird definiert, dass die Geste *schnell* ausgeführt werden soll (vgl. Kap. 7.1.2), wobei die Bedeutung von *schnell* anwendungsspezifisch ist. In der vierten Sektion rechts unten wird schließlich festgelegt, dass für die

Anwendung die Geschwindigkeit sowie der Ort der Ausführung relevant sind. Was letztlich mit den Informationen innerhalb der Anwendung geschieht, ist nicht Teil der Gestenspezifikation und muss separat dokumentiert werden.

Abb. 7.3 zeigt eine Variation dieser Geste, bei der die Geschwindigkeit der Ausführung keine Bedeutung hat: Jede linienförmige Bewegung nach links, egal wie schnell, soll als gültige Geste erkannt werden. Daher bleibt die Sektion für Eigenschaften leer, um keine zusätzlichen Bedingungen an die Ausführung zu definieren. Die Geschwindigkeit der Ausführung ist jedoch für die weitere Verarbeitung in der Anwendung interessant, denn in Abhängigkeit von ihr soll eine Aktion erfolgen bzw. beeinflusst werden. In der Sektion für Rückgaben wird also genau eine Eigenschaft definiert, nämlich eben die Geschwindigkeit, mit der die Geste ausgeführt wird.

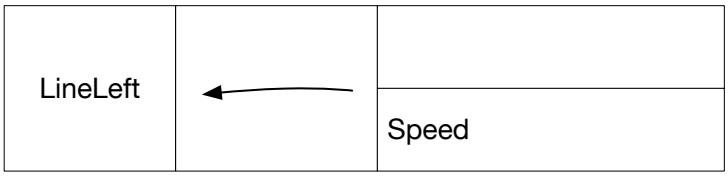


Abbildung 7.3: Beispiel einer GestureCard für eine einfache Geste: eine linienförmige Bewegung nach links, bei der die Ausführungsgeschwindigkeit für die Erkennung keine Rolle spielt, aber für die Anwendung relevant ist und ausgewertet werden muss

Einfache Gesten müssen jedoch nicht zwingend auf einen Gestenzug beschränkt sein. In der graphischen Darstellung können durchaus mehrere Gestenzüge integriert werden, für die dann jedoch alle definierten Eigenschaften gelten (ein Beispiel wird in Kap. 7.2.1.1 verwendet). Sollen die verschiedenen Gestenzüge mit unterschiedlichen Eigenschaften ausgestaltet werden, so müssen sie als separate Teilgesten definiert werden (vgl. Kap. 7.1.3).

7.1.2 Definition von Eigenschaften

Eigenschaften, die in GestureCards definiert werden, bestehen immer aus einem eindeutigen Namen, der die Eigenschaft identifiziert, sowie einem Wert oder einem Intervall, in dem gültige Werte liegen. Werte können aus zwei verschiedenen Quellen stammen, die je nach Anwendungsfall vom Projektteam abgewogen werden müssen. Sie können entweder den exakten, anhand der Sensoren messbaren Werten entsprechen, z. B. der Dauer einer Geste in Sekunden (`Duration: 2s`), oder aus einer Menge vordefinierter, qualitativer Schlüsselwörter gewählt werden (z. B. `Duration: Long`). Alle vordefinierten Schlüsselwörter abstrahieren dabei von den eigentlich über die Sensoren messbaren Werten und fassen diese auf einer höheren Ebene zusammen. Eine Bewegung kann z. B. schnell oder langsam ausgeführt werden, wobei die Ausprägung der gemessenen Werte letztlich anhand der verwendeten Sensoren bestimmt wird und auf der Ebene der Spezifikation nicht weiter relevant ist. Inwiefern die Definition mit vordefinierten Schlüsselwörtern oder exakten Werten sinnvoll ist, ist in erster Linie vom konkreten Anwendungsfall, aber auch vom Projektfortschritt oder den verwendeten Sensoren abhängig. In vielen Fällen lässt sich eine genaue Festlegung nur mit exakter Definition der verwendeten Hardware erreichen, da die unterschiedlichen Eigenschaften an spezifischen Besonderheiten der eingesetzten Geräte und Sensoren hängen können (z. B. der Auflösung des Bildschirms).

Operator	Bedeutung	Beispiel
<	Kleiner-Operator; legt eine Obergrenze fest	`Duration: <2s`
>	Größer-Operator; legt eine Untergrenze fest	`Duration: >2s`
..	Intervall-Operator; legt einen Wertebereich fest	`Duration: 1..2s`
+	Additions-Operator; addiert zwei Werte	`Duration: x + y`
−	Subtraktions-Operator; subtrahiert zwei Werte	`Duration: x - y`
*	Multiplikations-Operator; multipliziert zwei Werte	`Duration: x * y`
/	Divisions-Operator; dividiert zwei Werte	`Duration: x / y`
−>	Verweis-Operator; verweist auf den definierten Wert der Eigenschaft einer anderen Geste	`Duration: ->S1`

Tabelle 7.1: Mögliche Operatoren zur Definition von Eigenschaften

GestureCards erlauben drei grundlegende Operatoren zur Definition von Eigenschaften: den *Kleiner*-Operator (<), den *Größer*-Operator (>) sowie den *Intervall*-Operator (`..`). Des Weiteren können die einfachen arithmetischen Operationen +, −,* und / eingesetzt werden, um Werte exakter einzugrenzen. Die Operatoren werden verwendet, um die beschriebene Eigenschaft genauer zu beschreiben und Bedingungen für die Ausführung definieren zu können. Außerdem kann der *Verweis*-Operator (−>) verwendet werden, um auf die Definition eines Wertes in einer anderen GestureCard zu verweisen. Tab. 7.1 fasst die Bedeutung der Operatoren anhand von Beispielen zusammen.

7.1.3 Spezifikation zusammengesetzter Gesten

Zusammengesetzte Gesten bestehen – im Gegensatz zu einfachen Gesten – aus mehreren Teilgesten, die einzeln spezifiziert werden und einen Verbund bilden. Dabei können sowohl die einzelnen Teilgesten als auch die zusammengesetzte Gesamtgeste über verschiedene Eigenschaften verfügen. Abb. 7.4 zeigt den grundlegenden Aufbau einer zusammengesetzten Geste. Dabei werden die einzelnen Teile der zusammengesetzten Geste wie in Kap. 7.1.1 beschrieben definiert und von der zusammengesetzten Geste eingeschlossen. Übergreifende Eigenschaften, die für die Gesamtgeste gelten, werden in der rechten oberen Ecke beschrieben. Evtl. Rückgaben einer zusammengesetzten Geste werden rechts unten definiert. Wie bei der Definition einfacher Gesten ist die Angabe von Eigenschaften und Rückgaben optional.

Abbildung 7.4: Grundlegender Aufbau einer GestureCard für eine zusammengesetzte Geste

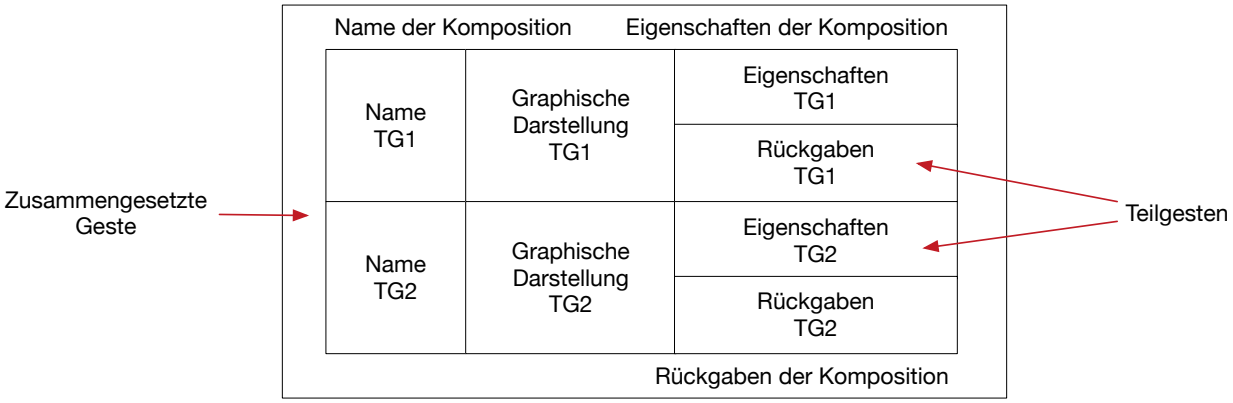


Abb. 7.5 zeigt ein Beispiel für eine zusammengesetzte Geste in Form eines Häkchens bestehend aus zwei Teilgesten, die ohne Pause hintereinander ausgeführt werden: eine Bewegung nach rechts unten mit dem Namen `S1` und eine nach rechts oben mit dem Namen `S2`. Außerdem definiert die Geste, dass die Teilgeste `S2` schnell ausgeführt wird und größer sein muss als die Teilgeste `S1`. Des Weiteren wird im Beispiel definiert, dass die komponierte Geste `Checkmark` heißt und zwischen einer und zwei Sekunden dauern darf. Außerdem definiert die GestureCard, dass für die Anwendung die Dauer und der Ort der Ausführung relevant sind.

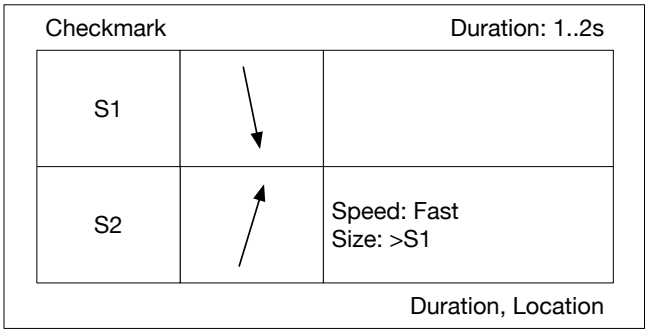


Abbildung 7.5: Beispiel einer GestureCard für eine zusammengesetzte Geste: eine Geste mit zwei Schenkeln in Form eines Häkchens

Zusammengesetzte Gesten können geschachtelt werden. Demnach müssen zusammengesetzte Gesten nicht zwangsläufig aussschließlich aus atomaren Gesten bestehen, sondern können diese noch weiter zusammenfassen. In Abb. 7.14 wird ein Beispiel gezeigt, wie eine zusammengesetzte und eine einfache Geste zu einer Gesamtgeste in Form einer umfassenden zusammengesetzten Geste vereint werden.

7.1.4 Referenzierung definierter Gesten

Als GestureCard definierte Gesten können in der Definition anderer GestureCards referenziert werden. Dazu können andere Gesten über ihren Namen angesprochen werden, daher müssen die Namen auf ihrer Ebene der Spezifikation eindeutig sein. Die Definition einer Geste befindet sich auf der höchsten Ebene und es kann nur eine Geste mit diesem Namen geben. Teilgesten hingegen werden innerhalb einer Geste definiert und müssen auch nur hier eindeutig benannt sein. In diesem Falle können Namen in anderen Gesten erneut verwendet werden.

Die in Abb. 7.5 dargestellte Geste kann entsprechend über `Checkmark` angesprochen werden. Teilgesten zusammengesetzter Gesten können über eine einfache Punktsyntax integriert werden (analog zum qualifizierten Klassennamen in der Programmiersprache Java). Zum Beispiel kann die Teilgeste `S1` in Abb. 7.5 über `Checkmark.S1` referenziert werden. Innerhalb einer zusammengesetzten Geste können andere Teilgesten direkt über den einfachen Namen referenziert werden, so wie z. B. in Abb. 7.5 in Teilgeste `S2` auf die Teilgeste `S1` verwiesen wird, um eine Abhängigkeit von ihrer Ausführung zu beschreiben. Da der Name der Gesamtgeste eindeutig sein muss, ist auch gewährleistet, dass der qualifizierte Name einer Teilgeste eindeutig ist. Re-

servierte Schlüsselwörter zur Definition von Eigenschaften dürfen nicht als Name von Gesten oder Teilgesten verwendet werden.

Zugegriffen wird bei Referenzen aber nicht auf Gesten selbst, sondern auf die Eigenschaften der referenzierten Geste, denn diese enthalten die relevanten Informationen über die Ausführung. Im Beispiel in Abb. 7.5 wird dabei implizit auf die gleiche Eigenschaft zugegriffen, in der die Referenz angegeben ist: Die Angabe `Size: >S1` zeigt an, dass die Größe der Geste `S2` die Größe der Ausführung von `S1` überschreiten muss. Soll auf eine andere Eigenschaft der Teilgeste verwiesen werden, so muss diese explizit genannt werden. Dazu wird in GestureCards ebenfalls eine einfache Syntax analog zu vielen Programmiersprachen verwendet: Andere Eigenschaften können über einen Punkt angesprochen werden. So kann z. B. formuliert werden, dass der Startpunkt von S2 der Endpunkt von S1 sein muss: `Start: S1.End`.[2] Wird die Referenz auf eine andere Teilgeste weggelassen, so beziehen sich die angegeben Eigenschaften immer auf die Geste selbst. Definiert also eine Geste die Eigenschaft `End: Start` bedeutet dies, dass sie dort enden muss, wo sie angefangen hat.

[2] Im gegebenen Beispiel ist eine explizite Definition des Startpunkts von S2 nicht nötig, da die Geste in einem Zug ausgeführt wird. Der Startpunkt von S2 ergibt sich somit implizit.

Grundsätzlich definiert eine Referenz auf eine Eigenschaft einen Wert zur Ausführungszeit der Geste. Angenommen, in einer Geste `S1` ist die Eigenschaft `Duration: >2s` definiert (`S1` muss länger als zwei Sekunden dauern), so bedeutet die Definition von `Duration: >S1` in einer Teilgeste `S2`, dass diese zweite Geste länger ausgeführt werden muss als `S1`. Das heißt, wenn `S1` drei Sekunden lang ausgeführt wird, dann muss `S2` länger als drei Sekunden ausgeführt werden. Soll hingegen die Eigenschaft in `S2` auf denselben Wert wie in `S1` gesetzt werden, dann ist der Verweis-Operator zu verwenden. `Duration: ->S1` wird dann zu `Duration: >2s` aufgelöst. In Abb. 7.9 ist ein Beispiel für die Verwendung des Verweis-Operators zu sehen.

7.1.5 Integration virtueller Objekte

In GestureCards kann in definierten Eigenschaften ein Bezug zu einem virtuellem[3] Objekt über das Schlüsselwort `Object` hergestellt werden. Dies kann sinnvoll sein, um Gesten, oder vielmehr die Bedeutung einer Geste, an ein Objekt zu binden und sie somit genauer zu spezifizieren. Zum Beispiel kann eine Spread-Geste zwei unterschiedliche Bedeutungen in einer Anwendung haben: Wird sie auf einem Objekt ausgeführt, so wird dieses vergrößert; wird sie einfach auf der Oberfläche ausgeführt, so zoomt die Anwendung hinein. Außerdem kann eine Geste explizit auf einem Objekt beginnen, z. B. um dieses zu verschieben. Abb. 7.7 zeigt beispielhaft, wie eine Objektreferenz in eine GestureCard integriert werden kann.

[3] Im Folgenden wird zwischen virtuellen und realen Objekten unterschieden. Virtuelle Objekte bezeichnen digitale Objekte, die auf einem UI zur Verfügung gestellt werden. Diese stehen im Fokus dieses Abschnitts. Reale Objekte hingegen bezeichnen Objekte der realen Welt.

Objekte können des Weiteren über frei wählbare Bezeichner näher benannt und somit unterschieden bzw. konkrete Objekte definiert werden. Der Bezeichner wird dazu in Klammern angehängt, z. B. `Object(Star)`, und ist eindeutig im Kontext der aktuellen GestureCard, d. h. eine Wiederverwendung des Bezeichners auf derselben GestureCard bezeichnet dasselbe Objekt. Das gilt auch für Bezeichner, die auf Teilgesten definiert werden, allerdings nicht für Bezeichner auf wiederverwendeten Gesten (vgl. Kap. 7.1.6), sodass

bei der Definition in diesem Fall Vorsicht angebracht ist. Es ist in der Regel sinnvoll, einen sprechenden Bezeichner für Objekte zu wählen, um das Verständnis zu erleichtern. Basis für den Namen können verschiedene Quellen sein, z. B. der Typ eines Objekts oder die Rolle, die das Objekt innerhalb der Geste einnimmt.

Durch unterschiedliche Bezeichner können auch Abhängigkeiten einer Geste zu verschiedenen Objekten definiert werden. Abb. 7.6 zeigt ein Beispiel, bei dem zwei unterschiedliche Objekte mit einer Geste beliebiger Form (durch das Schlüsselwort `FREE`, vgl. Kap. 7.2.1.8) verknüpft werden. Die Geste startet auf dem ersten Objekt (`Start: Object(Source)`) und endet auf dem zweiten (`Start: Object(Sink)`); in diesem Falle wurden als Namen die Rollen der Objekte gewählt. Da beide Objekte für die weitere Verarbeitung relevant sind, müssen sie auch beide als Rückgabe genannt werden.

Connect	FREE	Start: Object(Source) End: Object(Sink)
		Object(Source), Object(Sink)

Abbildung 7.6: Beispiel einer GestureCard, die von mehreren Objekten abhängig ist

Im Gegenzug können Objekte in einer GestureCard auch explizit ausgeschlossen werden. Die Geste soll dann quasi auf dem Hintergrund, dem sog.*Canvas*, direkt ausgeführt werden. Das verwendete Schlüsselwort ist dann `Canvas` anstelle von `Object`. Ob die Berührung eines Objekts dann während der Ausführung erlaubt ist oder nicht, ist vom jeweiligen Anwendungsfall abhängig. Gesten können z. B. auf einem Objekt beginnen und zwingend auf dem Canvas enden oder umgekehrt.

Wird die Geste, die einen Bezug zu einem virtuellen Objekt hat, von einer anderen Geste referenziert, so kann über die Eigenschaft `Object` auf das zugeordnete Objekt wie auf jede andere Eigenschaft zugegriffen werden. Die Eigenschaft kann nicht explizit gesetzt werden, sondern ist implizit belegt, wenn in der GestureCard ein Bezug zu einem Objekt hergestellt wurde, also das Schlüsselwort `Object` in einer Eigenschaft verwendet wurde. Dadurch kann z. B. sichergestellt werden, dass Teilgesten auf unterschiedlichen Objekten durchgeführt werden, indem bei einer Teilgeste das Objekt einer anderen Teilgeste explizit ausgeschlossen wird (vgl. Kap. 7.2.1.11). Eine Verwendung von `Object` als Rückgabe ist möglich; die Anwendung wird dann über das gebundene Objekt informiert.

7.1.6 Wiederverwendung definierter Gesten

Definierte Gesten können innerhalb anderer Gesten nicht nur referenziert, sondern auch wiederverwendet und als Teilgeste eingesetzt werden. Dazu wird das gesamte Feld, das eigentlich zur Definition einer Teilgeste dient, mit dem Namen der wiederzuverwendenden Geste überschrieben. Dadurch können auch Kombo-Gesten, wie von Cirelli & Nakamura (2014) gefordert, spezifiziert werden.

Im folgenden Beispiel werden zwei Gesten definiert, die letztlich die gleiche Aktion auf unterschiedlichen Wegen auslösen. Dabei soll ein Objekt auf einer berührungsempfindlichen Oberfläche an eine andere Position verschoben werden. In jedem Fall muss die Aktion durch die Aktivierung des Objekts initiiert werden, auf die dann weitere Aktionen folgenden können. Dazu wird eine Geste `Activate` definiert, die in Abb. 7.7 dargestellt ist.

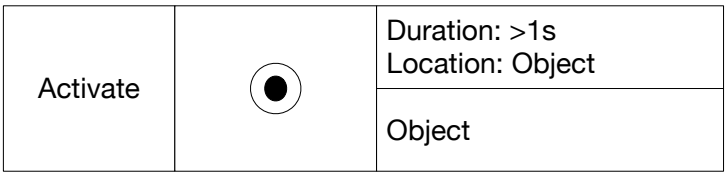


Abbildung 7.7: GestureCard, die definiert, dass eine Berührung länger als eine Sekunde dauern und auf einem Objekt ausgeführt werden muss

Bei der Geste `Activate` muss ein Objekt länger als eine Sekunde berührt werden. Das Symbol in der Sektion zur graphischen Repräsentation bedeutet eine einfache Berührung ohne Bewegung des Fingers (vgl. Kap. 7.2.1.8), also eine Pose im Sinne von Oberflächengesten. Die Eigenschaft `Duration` definiert die Dauer der Geste (länger als eine Sekunde) und die Angabe der Eigenschaft `Location` schränkt ein, dass die Geste auf einem virtuellen Objekt ausgeführt werden muss. Die wesentliche Information für die Anwendung ist, auf welchem Objekt die Geste ausgeführt wurde. Daher wird `Object` in der Sektion für Rückgaben gesetzt; dadurch wird das mit der Geste verknüpfte Objekt angesprochen (vgl. Kap. 7.1.5). Diese Geste wird in der folgenden Geste zur Initiierung der eigentlichen Aktion wiederverwendet.

Die folgenden Beispiele verdeutlichen die Vorteile, die sich aus der Wiederverwendung von Teilgesten ergeben. Es werden zwei Gesten[4] definiert, die die gleiche Aktion auf unterschiedliche Weise auslösen: Es soll ein Objekt verschoben werden. Benutzer könnten je nach eigener Präferenz, aktuellem Kontext oder aus anderen Gründen die eine oder andere Variante wählen, die zwar mit der gleichen Teilgeste eingeleitet, aber durch unterschiedliche Teilgesten vollendet wird. Abb. 7.8 zeigt die erste Variante. Hierzu muss das Objekt erst mithilfe der bereits definierten Geste `Activate` aktiviert werden, um anschließend durch eine freie Bewegung an seine neue Position gebracht zu werden. Die bereits definierte Geste kann wie dargestellt als Teilgeste wiederverwendet werden. Die gesamte Definition der Teilgeste wird vollständig durch den Namen einer zuvor definierten Geste ersetzt.

[4] Beide im Beispiel verwendeten Gesten stammen aus dem Gestenset von Wobbrock et al. (2009): $Move_1$ und $Move_2$.

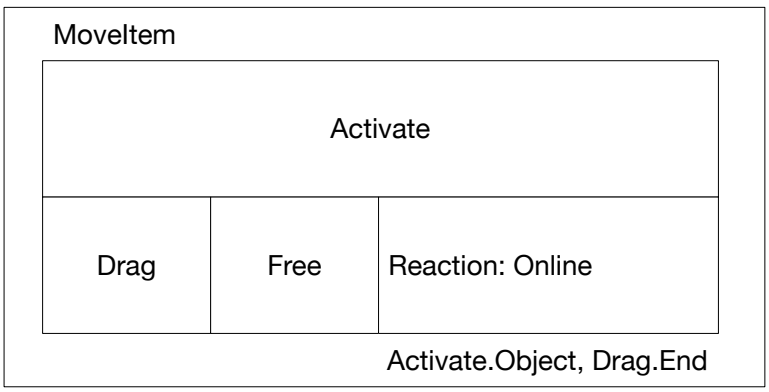


Abbildung 7.8: Beispiel einer GestureCard für die Wiederverwendung einer zuvor definierten Teilgeste

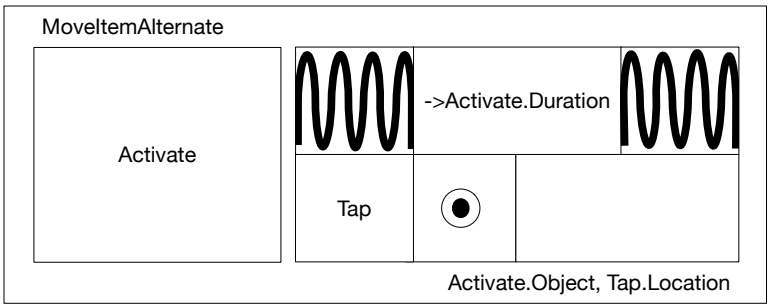


Abbildung 7.9: Alternatives Beispiel einer GestureCard für die Wiederverwendung einer zuvor definierten Teilgeste

Abb. 7.9 zeigt eine alternative Geste, bei der ein Objekt nach erfolgter Aktivierung durch ein einfaches Antippen der Oberfläche mit einem anderen Finger verschoben wird. Dabei darf die zweite Teilgeste erst ausgeführt werden, wenn die Mindestdauer der Teilgeste `Activate` überschritten wird. Diese Einschränkung wird durch die geschlängelte Linie und den eingetragenen Ausdruck definiert. Zu beachten ist an dieser Stelle die Verwendung des Verweis-Operators, der dafür sorgt, dass die Pause mindestens so lang ist, wie die Mindestdauer der Geste `Activate`. Darüber hinaus müssen bei Durchführung der zweiten Teilgeste weiterhin beide Finger auf der Oberfläche sein. Beide Teilgesten werden nebeneinander dargestellt, wodurch eine zeitliche Parallelisierung definiert wird (vgl. Kap. 7.2.2.1).

In beiden Fällen wird die bereits definierte Geste `Activate` wiederverwendet, um die Aktion zu initiieren, und es müssen zwei Informationen zurückgeben werden: das aktivierte Objekt und die neue Position auf dem Bildschirm. Im ersten Fall wird diese durch das Ende der freien Bewegung definiert, entsprechend muss der Endpunkt der Teilgeste `Drag` zurückgegeben werden. Im zweiten Fall kann einfach der Ort der Teilgeste `Tap` zurückgegeben werden, da sich Start und Ende nicht unterscheiden.

7.2 Integration von Gesteneigenschaften

Im Folgenden werden die Gesteneigenschaften beschrieben, die mit GestureCards definiert werden können. Zu jeder Eigenschaft werden Schlüsselwörter und mögliche quantitative Werte sowie qualitative Abstraktionen definiert. Dabei kann, wie in Kap. 7.1.2 eingeführt, jede Eigenschaft mit quantitativen Werten belegt werden, die direkt von den Sensoren abgeleitet werden können (z. B. `2s`), oder aber ein vordefiniertes Schlüsselwort verwendet werden, das den intendierten Wert qualitativ umschreibt (z. B. `Long`). Einige der Eigenschaften, die mit GestureCards beschrieben werden können, ergeben sich nicht direkt aus der Analyse wissenschaftlicher Literatur in Kap. 5, sondern aus der für GestureCards gewählten Struktur. Insbesondere die Möglichkeit, Gesten aus eigens definierten Teilgesten zusammenzustellen, erzwingt, dass weitere Eigenschaften definiert werden müssen, um Ungenauigkeiten in der Spezifikation zu vermeiden. Dazu gehören die Eigenschaften der Größe (Kap. 7.2.2.2) und der Distanz (Kap. 7.2.2.4). Darüber hinaus ermöglichen GestureCards die explizite Spezifikation von Start- und Endpunkt, die in Kap. 5.1.10 als besondere Form des Ortes der Ausführung identifiziert wurden. Alle Eigenschaften sowie die verwendeten Schlüsselwörtern sind

als Referenz in Anh. A zu finden. Die Definition der Eigenschaften und die Beispiele fokussieren den Einsatz der Hände zur Ausführung von Gesten, auch wenn moderne Sensoren die Erkennung von Bewegung unterschiedlicher Körperteile wie z. B. der Füße ermöglichen.

7.2.1 Atomare Eigenschaften

Atomare Eigenschaften beziehen sich auf die Geste als solche, ohne zwingend in direkter Beziehung zu anderen Gesten zu stehen. Eine Beziehung zu einer anderen Geste ist allerdings möglich. Eigenschaften erhalten einen Wert oder Wertebereich, wie in Kap. 7.1.2 definiert, der als Bedingung für die Erkennung zu verstehen ist. Dabei können allerdings andere Teilgesten und ihre Eigenschaften referenziert werden.

7.2.1.1 Anzahl der Werkzeuge

Die Anzahl der verwendeten Werkzeuge kann auf verschiedene Arten in GestureCards dargestellt werden:

- als Teil der graphischen Darstellung.
- als Eigenschaft `Tool Count`, die die Anzahl der Werkzeuge in der Sektion für Eigenschaften festlegt.
- durch die explizite Darstellung verschiedener Teilgesten, denen auch unterschiedliche Werkzeuge zugewiesen werden können (vgl. Kap. 7.2.1.15).

Alle Varianten haben Vor- und Nachteile. Die Integration in die graphische Darstellung ist kompakter und insbesondere sinnvoll, wenn den einzelnen Teilgesten keine weiteren unterschiedlichen Eigenschaften zugeordnet werden, jedoch impliziert die graphische Darstellung u. U. andere, nicht gewünschte Eigenschaften, wie z. B. die räumliche Relation. Dieser Nachteil kann durch Angabe der Anzahl der Werkzeuge als Eigenschaft umgangen werden. Die letztgenannte Variante erlaubt die explizite Spezifikation einzelner Teilgesten, wodurch jeder Teilgeste ein eigenes Werkzeug zugewiesen werden kann.

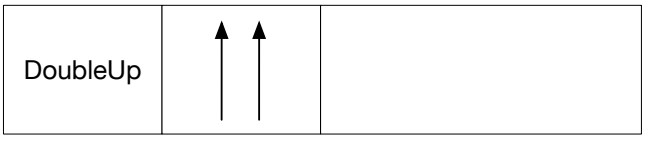


Abbildung 7.10: Beispiel einer GestureCard zur Spezifikation parallelen Werkzeugeinsatzes in der graphischen Sektion

Ein Beispiel, bei dem die Anzahl der Werkzeuge in die graphische Darstellung integriert wurde, zeigt Abb. 7.10. Sie beschreibt eine doppelte Linie nach oben, wobei das Werkzeug nicht definiert wird – die dargestellte Geste könnte also mit zwei Fingern auf einer berührungsempfindlichen Oberfläche oder auch mit zwei Händen als räumliche Geste ausgeführt werden. Die Verwendung mehrerer Werkzeuge in der graphischen Sektion definiert immer die parallele Ausführung der Gestenzüge. Sequentielle Ausführungen müssen als separate Teilgesten definiert werden (vgl. Kap. 7.2.2.1). Implizit definiert die

GestureCard auch, dass beide Teilgesten nebeneinander ausgeführt werden. Durch die Verwendung der graphischen Sektion zur Verdeutlichung der Anzahl verwendeter Werkzeuge werden also weitere Eigenschaften integriert. Weitere definierte Eigenschaften gelten in diesem Falle für die Gesamtgeste, also beide Teile der Ausführung.

Alternativ kann die Eigenschaft `Tool Count` definiert werden. Hier kann die Anzahl verwendeter Werkzeuge explizit angegeben werden, so wie in Abb. 7.11 dargestellt. Wird die GestureCard so definiert, dann wird die implizit in Abb. 7.10 vorhandene räumliche Relation – nämlich dass beide Teilgesten nebeneinander ausgeführt werden – aufgehoben. Die Positionierung der Teilgesten ist nun nicht mehr definiert und jede denkbare Variante zweier geradliniger Bewegungen nach oben wäre eine gültige Geste.

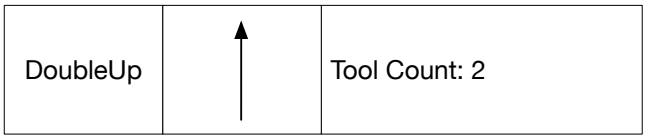
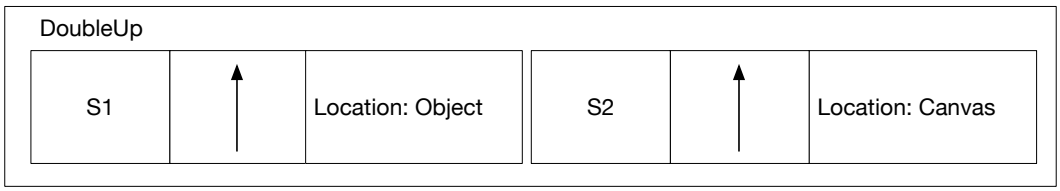


Abbildung 7.11: Beispiel einer GestureCard zur Spezifikation parallelen Werkzeugeinsatzes in der Sektion für Eigenschaften

Die dritte Variante ist in Abb. 7.12 dargestellt: Beide Teilgesten werden explizit beschrieben. Durch die Darstellung nebeneinander wird deutlich gemacht, dass beide Teilgesten parallel ausgeführt werden (vgl. Kap. 7.2.2.1). Der Vorteil dieser Variante ist, dass für beide Teilgesten unterschiedliche Eigenschaften definiert werden können; so muss im Beispiel die Teilgeste `S1` auf einem Objekt ausgeführt werden, während die zweite Teilgeste den Kontakt zu einem Objekt explizit ausschließt. Auch hier ist die Positionierung der Teilgesten zueinander – im Gegensatz zur Integration der Anzahl der Werkzeuge in die graphische Sektion und analog zur expliziten Angabe als Eigenschaft – nicht definiert, d. h. die zweite Teilgeste kann an einer beliebigen Stelle in Bezug auf die erste Teilgeste durchgeführt werden. Zur weiteren Einschränkung muss hier die räumliche Relation angegeben werden (vgl. Kap. 7.2.2.3).

Einen Standardwert gibt es für diese Eigenschaft nicht, da das Weglassen von `Tool Count` dazu führt, dass die graphische Repräsentation bzw. Anordnung der Teilgesten die Vorgabe macht. Eine Verwendung als Rückgabe ist nur sinnvoll, wenn die Eigenschaft `Tool Count` explizit angegeben ist und mit einem Operator versehen ist. Dann wird davon ausgegangen, dass die Anwendung die genaue Anzahl der Werkzeuge weiterverarbeitet. In allen anderen Fällen ist diese Information durch die genaue Definition innerhalb der Geste nicht dynamisch und gleichzeitig auch Bedingung für die korrekte Ausführung.

Abbildung 7.12: GestureCard für die gleiche Geste wie in Abb. 7.10, jedoch mit unterschiedlichen Eigenschaften je Teilgeste



7.2.1.2 Auflagefläche

Die Quintessenz der Analyse in Kap. 5.1.2 ist, dass die Auflägefläche in erster Linie zur Erkennung anderer Eigenschaften herangezogen wird und somit viele Details für eine Spezifikation an anderer Stelle wesentlich sind. Allerdings zeigt die Literatur auch, dass die Veränderung der Auflagefläche durchaus für interessante Gesten herangezogen werden kann. In GestureCards können zwei Arten der Veränderung spezifiziert werden: Die Auflagefläche kann sich vergrößern oder schrumpfen. Unter dem Schlüsselwort `Contact Area` können entsprechend die Werte `Decrease` und `Increase` verwendet werden, um Veränderungen der Auflagefläche zu spezifizieren. Die Eigenschaft hat keinen Standardwert. Wird die Auflagefläche als Rückgabe einer Geste verwendet, so wird belegte Fläche und die Information, ob die Fläche geschrumpft oder gewachsen ist, als für die Anwendung relevant erachtet.

Abb. 7.13 zeigt die Spezifikation eines Beispiels von Boring et al. (2012) als GestureCard. Dabei wird ein Finger verwendet, um auf einer Smartwatch zu zoomen – eine Funktion, für die üblicherweise zwei Finger verwendet werden, die sich aufeinander zu oder voneinander weg bewegen. Auf der recht kleinen Oberfläche einer Smartwatch ist diese Geste jedoch nur sehr schwer exakt zu verwenden. Die Autoren definieren in ihrem Beispiel explizit die Verwendung des Daumens, denkbar ist aber auch die Verwendung eines anderen Fingers. Die dargestellte GestureCard gibt das Werkzeug daher nicht explizit vor.

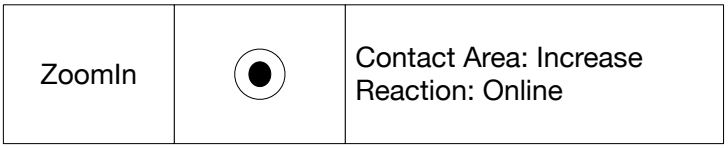


Abbildung 7.13: GestureCard für die Darstellung wachsender Auflagefläche

7.2.1.3 Ausführender Benutzer

Die verwendete Definition von Gesten, an denen unterschiedliche Benutzer beteiligt sind, basiert im wesentlichen auf den Arbeiten von Morris et al. (2006), die in Kap. 5.1.3 vorgestellt worden sind. Morris et al. (2006) unterscheiden dabei sechs Dimensionen von Mehrbenutzergesten: Symmetrie, Parallelität, Nähe, Additivität, Benutzeridentität und Anzahl der Benutzer/Geräte.

Die Definition der Symmetrie einer Mehrbenutzergeste ist in GestureCards durch die Möglichkeit, Teilgesten zu spezifizieren, gegeben. Eine Mehrbenutzergeste ist symmetrisch, wenn alle beteiligten Benutzer identische Teilgesten ausführen und asymmetrisch, wenn sich die Teilgesten einzelner Benutzer unterscheiden.

Auch Parallelität kann mit den vorhandenen Mitteln von GestureCards dargestellt werden. Dazu muss die zeitliche Relation von Teilgesten verwendet werden, die bestimmt, ob eine Mehrbenutzergeste parallel oder seriell ausgeführt wird. Bei den Spezifikation mit GestureCards kann auf die in Kap. 7.2.2.1 definierten Möglichkeiten zurückgegriffen werden.

Auch die Dimension der Nähe, so wie Morris et al. (2006) sie definieren, kann mit GestureCards ausgedrückt werden. Eine Geste ist demnach *public*, wenn alle beteiligten Benutzer ihre jeweiligen Teilgesten auf unterschiedlichen Geräten ausführen, und *social*, wenn alle Benutzer dasselbe Gerät verwenden. Beides kann in GestureCards durch Spezifikation des Ausführungsortes definiert werden. Auch Gesten, die *personal* sind, können mithilfe des Ausführungsorts definiert werden, wenn dieser bei allen Teilgesten auf ein konkretes Objekt eingegrenzt wird. Auf eine Definition von *intimate*, bei der Morris et al. (2006) eine körperliche Berührung der beteiligten Benutzer verlangen, wird in GestureCards verzichtet.

Morris et al. (2006) definieren des Weiteren die Dimension der Additivität, die sich weniger auf die Geste selbst, sondern mehr auf die semantische Bedeutung einer Geste für die Anwendung bezieht. Eine direkte Entsprechung findet sich in GestureCards daher nicht, jedoch kann Additivität auf verschiedene Arten definiert werden: Zum einen kann die gleiche Geste mehrfach, aber mit unterschiedlicher Benutzeranzahl beschrieben werden, zum anderen kann die Rückgabe einer Geste verwendet werden, um die beteiligten Benutzer zu zählen, was als Hinweis auf die Bedeutung zu verstehen ist.

Ausführende Benutzer werden über das Keyword `User` in GestureCards definiert, wodurch auch die von Morris et al. (2006) beschriebene Dimension der Benutzer-/Geräteanzahl abgedeckt werden kann. Je nach Anwendungsfall können unterschiedliche Werte verwendet werden:

- `All`: Alle Benutzer führen diese Geste aus.
- Bruch: Wenn nur eine bestimmte Teilmenge aller Benutzer eine Geste ausführen soll, kann ein Bruch verwendet werden (z. B. besagt `<1/2`, dass weniger als die Hälfte aller Benutzer die Geste durchführen sollen).
- Zahlwert: Diese Anzahl Benutzer muss an der Geste beteiligt sein (z. B. `>2` besagt, dass mindestens 3 Benutzer beteiligt sein sollen).
- Buchstabe: Zur direkten Unterscheidung verschiedener Benutzer werden diese mit Buchstaben bezeichnet, z. B. `A` und `B`. Innerhalb der Teilgesten einer Gesamtgeste bezeichnet derselbe Buchstabe auch denselben Benutzer, jedoch sind die Buchstaben nicht gestenübergreifend gültig.

Wird die Eigenschaft nicht angegeben, wird davon ausgegangen, dass die Geste von genau einem Benutzer ausgeführt wird (`User: 1`) und dass es sich bei allen Teilgesten um denselben Benutzer handelt. Eine Angabe von `User` als Rückgabe einer Geste gibt alle beteiligten Benutzer zurück. Aus dieser Information kann dann auf weitere, benutzerspezifische Daten geschlossen werden. Eine Alternative und denkbare Erweiterung zur Bezeichnung mit Buchstaben können sprechende Namen für Benutzer sein, die nicht auf eine Geste beschränkt sind, sondern für alle Gesten gelten. Damit können u. a. gestenübergreifende Rollen (z. B. *Besitzer* eines virtuellen Objekts) definiert und Gesten auf bestimmte Benutzer eingeschränkt werden. Die Dimension der Benutzeridentität von Morris et al. (2006) könnte so umgesetzt werden.

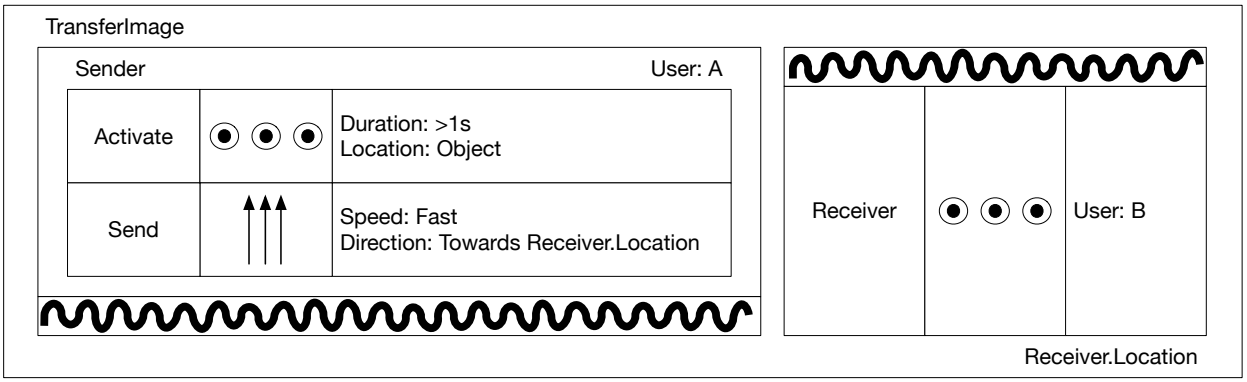


Abbildung 7.14: GestureCard für eine Geste mit zwei Benutzern

Abb. 7.14 zeigt, wie eine Mehrbenutzergeste definiert wird. Morris et al. (2006) beschreiben diese Geste als Teil ihrer beispielhaften Anwendung, mit der mehrere Benutzer Bilder auf einem Multi-Touch-Tisch bearbeiten können. Bei der beschriebenen Geste übergibt ein Benutzer ein Bild an einen anderen. Der erste Benutzer versendet das Bild, hier dargestellt durch die Teilgeste `Sender`, die wiederum aus zwei Teilgesten besteht. Zunächst muss das Bild durch eine längere Berührung mit drei Fingern ausgewählt (Teilgeste `Activate`) und dann durch einen Swipe (Teilgeste `Send`) an den anderen Benutzer übergeben werden. Dabei müssen die Finger, mit denen das Bild ausgewählt wurde, auch für den zweiten Teil verwendet werden und dürfen die Oberfläche nicht verlassen. Die Teilgeste `Receiver` wird vom zweiten Benutzer durchgeführt, der durch eine Berührung mit drei Fingern festlegen kann, wohin auf dem Tisch das Bild transferiert werden soll. Der erste Benutzer muss den Swipe in Richtung der Berührung des zweiten Benutzers ausführen. Die Spezifikation definiert darüber hinaus einen Verzug zwischen den beiden Benutzern durch die geschlängelten Linien: Benutzer A initiiert die Geste, während Benutzer B sie beendet. Allerdings ist keine genaue Zeitspanne festgelegt, wann die Teilgeste `Receiver` starten soll. Die gesamte Geste gibt letztlich den Ort der Berührung durch den zweiten Benutzer zurück, da hier das Bild abgelegt werden soll.

7.2.1.4 Ausrichtung

Die Ausrichtung einer Geste beschreibt, ob die Form der Geste im Raum gedreht werden kann oder ob sie genauso ausgeführt werden muss, wie in der graphischen Repräsentation vorgegeben. Dazu wird in GestureCards das Schlüsselwort `Orientation` mit folgenden qualitativen Werten definiert:

- `Fixed`: Die Geste muss bei der Ausführung genauso ausgerichtet sein, wie bei der Spezifikation festgelegt.
- `None`: Die Geste kann bei der Ausführung beliebig ausgerichtet sein.[5]

Wird die Eigenschaft nicht angegeben, so wird als Standardwert `Orientation: Fixed` gesetzt. Standardmäßig wird die Ausrichtung der Geste außerdem aus Sicht des Benutzers definiert, d. h. wenn der Benutzer die Möglichkeit hat, um den Sensor herumzugehen (z. B. bei Verwendung eines Multi-Touch-Tisches), ändert sich die Ausrichtung der Geste mit sei-

[5] In einer ersten Version von GestureCards konnte dieses Verhalten mit der Eigenschaft `Direction: Invariant` definiert werden, die zwischenzeitlich entfernt wurde. In den Studien, die in Kap. 8 und Kap. 9 besprochen werden, wird diese Variante daher an einigen Stellen auftauchen.

nem Standort. Sollte die Ausrichtung am Sensor festgemacht werden, dann muss die Eigenschaft mit dem Schlüsselwort `Sensor` erweitert werden. `Orientation: Sensor Fixed` legt also fest, dass die Geste aus Sicht des Sensors genauso ausgerichtet werden muss, wie in der zugehörigen GestureCard definiert. Die Ausrichtung kann auch als Rückgabe verwendet werden. Es wird dann erwartet, dass die Anwendung mit einer Angabe des Winkels weiterarbeitet, um den die Geste von der in der GestureCard definierten Ausrichtung abweicht. Für Gesten, deren Form zur Spezifikationszeit nicht bekannt ist (Freiformgesten, vgl. Kap. 7.2.1.8), kann `Orientation` nicht als Rückgabe gesetzt werden.

Zusammengesetzte Gesten verteilen allerdings die Definition der Form über mehrere Teilgesten. Die Eigenschaft `Orientation` kann daher in den Teilgesten einer zusammengesetzten Geste nicht definiert werden, da ansonsten die Form der Gesamtgeste verloren geht. Die Gesamtgeste selbst kann jedoch mit `Orientation: None` definiert werden. Abb. 7.15 zeigt ein einfaches Beispiel. Die Geste beschreibt eine Geste in Form eines V, bei der beide Gestenzüge mit einer kurzen Unterbrechung zwischendrin (als Multi-Stroke-Geste (vgl. Kap. 7.2.2.1)) durchgeführt werden. Die Gesamtgeste wird mit `Orientation: None` definiert, wodurch sie beliebig ausgerichtet sein kann. Die einzelnen Teilgesten behalten ihre Bewegungsrichtung in Bezug zueinander bei.

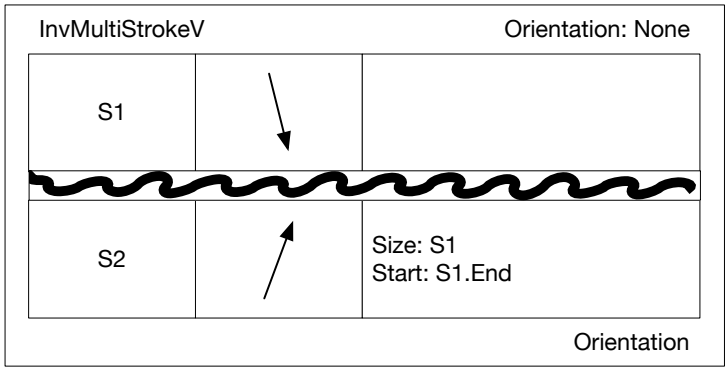


Abbildung 7.15: GestureCard für eine Geste in Form eines V, die beliebig ausgerichtet sein kann

7.2.1.5 Bewegungsrichtung

Die Richtung, in der eine Geste bzw. ein Gestenzug ausgeführt wird, kann in GestureCards an zwei Stellen definiert werden: zum einen in der graphischen Repräsentation, zum anderen mit dem Schlüsselwort `Direction` als Eigenschaft. Letzteres dient dazu, die Bewegungsrichtung in Abhängigkeit einer Teilgeste oder eines virtuellen Objekts zu beeinflussen und die Definition von Dynamik zu ermöglichen. In der graphischen Repräsentation werden statische Richtungsangaben definiert, die durch zwei Fälle beeinflusst werden können:

- Die Eigenschaft `Orientation: None` ist gesetzt, wodurch die Ausrichtung der Geste nicht relevant ist. Zwangsläufig ändert sich dann die Bewegungsrichtung der Gestenzüge.

- Die Eigenschaft `Direction` ist gesetzt, wodurch sich die Bewegungsrichtung gemäß des angegebenen Wertes ändert. Die Form der Geste wird entsprechend im Raum ausgerichtet.

Zur einfachen Definition der Richtung werden in der graphischen Repräsentation Pfeile verwendet. Die Geste startet am Beginn des Schaftes und bewegt sich in Richtung der Pfeilspitze. Ohne weitere Angabe der Eigenschaft `Direction` definiert die graphische Repräsentation an dieser Stelle die Geste so, wie sie ausgeführt werden soll. Soll die Geste darüber hinaus in weitere Richtungen gültig sein, die nicht von anderen Teilgesten oder virtuellen Objekten abhängig sind, dann muss entweder `Orientation: None` gesetzt werden, wodurch alle denkbaren Richtungen möglich sind, oder es müssen für jede mögliche Richtung zusätzliche Gesten definiert werden.

Die Bewegungsrichtung kann aber auch durch dynamische Aspekte beeinflusst werden, die zum Zeitpunkt der Gestendefinition nicht genau festgelegt werden können. Dies können z. B. andere Teilgesten oder virtuelle Objekte sein, die durch den jeweiligen Anwendungsfall bestimmt werden und auf die die Anwendung zur Laufzeit reagieren muss. GestureCards erlauben die Integration dynamischer Aspekte bei der Angabe der Richtung durch die Definition einer Relation zu einem Ziel: Gesten können sich auf etwas zu oder von etwas weg bewegen.

Eine Bewegung auf etwas zu wird in der Eigenschaft `Direction` mit dem Schlüsselwort `Towards` eingeleitet und die Bewegung von etwas weg mit dem Schlüsselwort `Away from`. Beide Schlüsselwörter erwarten die Angabe eines Bezugspunktes: entweder eine andere Teilgeste (bzw. eine mit einer Ortsangabe verknüpfte Eigenschaft wie Ort, Start- oder Endpunkt) oder ein virtuelles Objekt (über das Schlüsselwort `Object`). Abb. 7.16 zeigt ein Beispiel, bei dem sich die zweite Teilgeste auf die erste zubewegt. Die graphische Repräsentation definiert zwar wie zuvor beschrieben eine Bewegung nach rechts, jedoch wird diese durch die genauere Spezifikation über die Eigenschaft `Direction` durch die Integration eines dynamischen Aspektes so überschrieben, dass sich die Geste auf die Teilgeste `Selection` zubewegen muss. Das bedeutet letztlich, dass wenn die Teilgeste `Tap` links von der Teilgeste `Swipe` ausgeführt wird, diese sich nach rechts bewegen muss.

Die Bewegungsrichtung muss zwingend in der graphischen Repräsentation angegeben werden, daher ist kein Standardwert nötig. `Direction` kann nicht als Rückgabe verwendet werden; stattdessen soll auf `Orientation` zurückgegriffen werden.

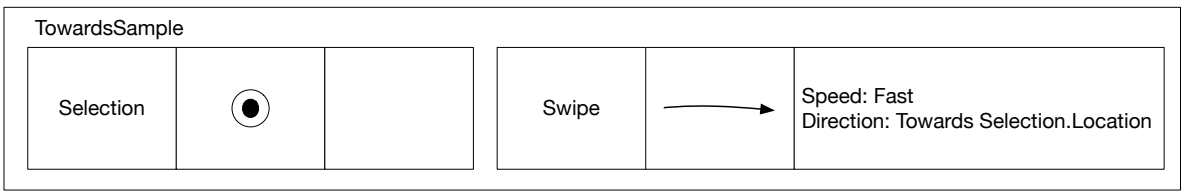


Abbildung 7.16: GestureCard für einen Swipe, der sich auf eine andere Teilgeste zubewegt

7.2.1.6 Dauer

Das zur Definition der Dauer der Ausführung verwendete Schlüsselwort ist `Duration`. Die Dauer kann in GestureCards auf zweierlei Arten definiert werden: als genaue Zeitangabe oder als eines der u. a. Schlüsselworte für vorgegebene Werte. Die Zeitangabe kann dabei in einer üblichen Einheit erfolgen; es ist jedoch davon auszugehen, dass typischerweise Millisekunden oder Sekunden verwendet werden. Die Schlüsselworte für vorgegebenen Werte orientieren sich an den von Ghomi et al. (2012) verwendeten Definitionen.[6] Sie unterscheiden drei unterschiedliche Möglichkeiten:

- `Impulse`: eine sehr kurze Ausführung, z. B. die Berührung und sofortiges Lösen von einer berührungsempfindlichen Oberfläche
- `Short`: eine kurze Ausführung, z. B. die Berührung und kurzes Verweilen auf einer berührungsempfindlichen Oberfläche
- `Long`: eine lange Ausführung, als Konvention wird mindestens die doppelte Verweildauer der kurzen Ausführung (`Short`) definiert

Die tatsächliche Länge, die sich hinter den Schlüsselwörtern verbirgt, wird an dieser Stelle nicht definiert. Sie ist abhängig von der jeweiligen Anwendung und muss zwischen den Mitgliedern des Projektteams abgestimmt werden. Ein Standardwert für diese Eigenschaft wird nicht explizit definiert, jedoch ist von `Duration: Short` als gewöhnliches Verhalten von Benutzern auszugehen. Bei Verwendung als Rückgabe der Geste wird davon ausgegangen, dass für die Anwendung die tatsächliche Dauer oder eine Einordnung in o. g. Kategorien relevant ist.

Wie Zeitangaben für die Dauer einer Geste verwendet werden, wird in zahlreichen Beispielen (u. a. in Abb. 7.7) demonstriert. Ein Beispiel für die Verwendung der definierten Werte zeigt Abb. 7.17. Die Geste beschreibt eine rhythmische Sequenz, die sich an den Beispielen von Oakley et al. (2015) orientiert. Dabei werden zu Beginn der Mittel- und der Ringfinger gleichzeitig auf eine berührungsempfindliche Oberfläche gelegt und die Oberfläche anschließend mit dem Ringfinger zweimal angetippt. Der Mittelfinger bleibt dabei die ganze Zeit auf der Oberfläche liegen und verlässt diese erst, wenn der die Teilgeste `LeftT2` durchgeführt wurde.

[6] Die Autoren basieren ihre Arbeit auf Takten, wie sie in der Musik verwendet werden, darauf wird im Rahmen dieser Arbeit jedoch verzichtet.

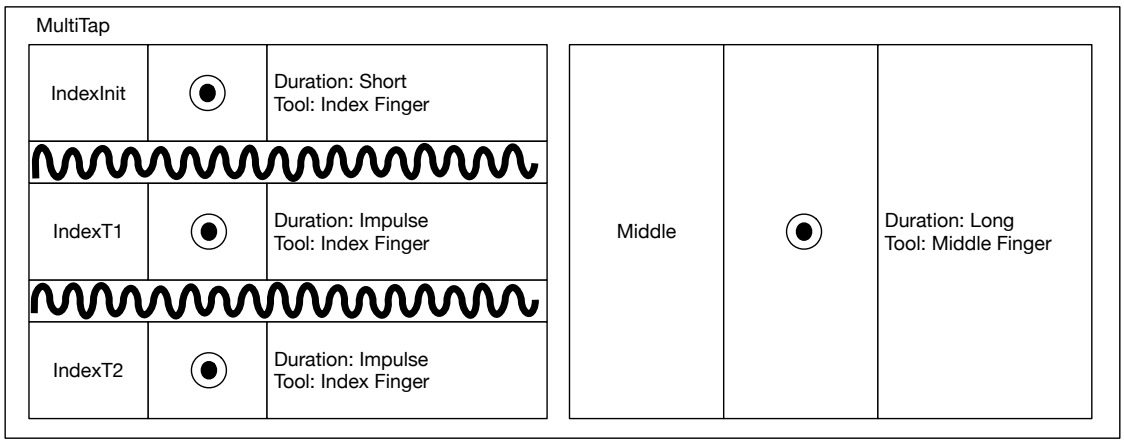


Abbildung 7.17: GestureCard für eine rhythmische Geste, bei der Teilgesten unterschiedlich lange dauern

7.2.1.7 Druck

Zur Spezifikation des Drucks auf die Oberfläche wird in GestureCards das Schlüsselwort `Pressure` verwendet. Eine explizite Angabe des Drucks, wie sie von verschiedenen Frameworks typischerweise normalisiert vorgegeben wird (vgl. Kap. 5.1.6), wird nicht eingesetzt, da sie nicht wiederverwendet und nur im Kontext des jeweiligen Frameworks sinnvoll definiert werden kann. In GestureCards werden sechs verschiedene Stufen verwendet, um unterschiedlich starken Druck zu bezeichnen:

- `Hover`: Kein Druck, Werkzeug schwebt über der Oberfläche
- `Very Weak`: Sehr schwacher Druck
- `Weak`: Schwacher Druck
- `Normal`: Üblicher Druck, z. B. wie zum Antippen eines Buttons
- `Strong`: Starker Druck
- `Very Strong`: Sehr starker Druck

Ramos et al. (2004) berichten, dass Nutzer sechs Stufen optimal unterscheiden können. Standardwert ist `Normal`, der dem üblichen Vorgehen von Benutzern entspricht. Bei der Verwendung als Rückgabe ist für die Anwendung der tatsächlich ausgeübte Druck sowie eine Einteilung in o. g. Kategorien relevant.

Abb. 7.18 zeigt ein Beispiel, bei dem der Druck auf die Oberfläche in einer Teilgeste spezifiziert wird. Die Geste wird mit einer schnellen Bewegung nach rechts auf der Oberfläche begonnen und mündet in einen starken, bewegungslosen Druck. Die Rückgabe deutet außerdem an, dass der Ort des Drucks die für die weitere Verarbeitung in der Anwendung wesentliche Information ist.

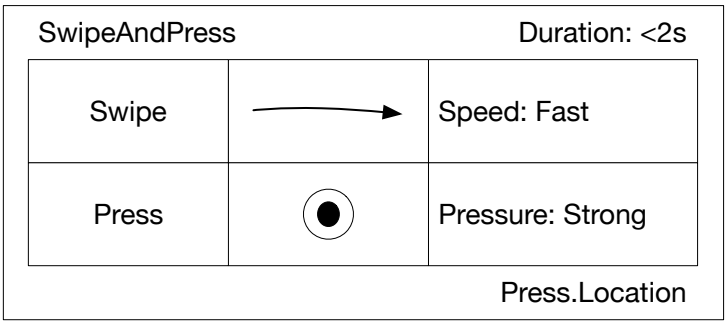


Abbildung 7.18: GestureCard, die einen Swipe nach Rechts gefolgt von einem Druck auf die Oberfläche definiert

7.2.1.8 Form

Die Form einer Geste wird über die graphische Repräsentation beschrieben. Hier werden alle Züge, die zu der definierten Teilgeste gehören, als beliebige geometrische Figur bzw. geometrischer Linienzug dargestellt. Ein Gestenzug ist eine für die Ausführung der Geste relevante Bewegung. Gesten können – wie in Kap. 5.2.1 dargestellt – aus einem oder mehreren Zügen bestehen, die sequentiell oder parallel ausgeführt werden. In GestureCards kann nicht

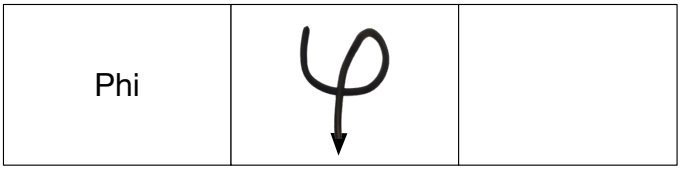


Abbildung 7.19: GestureCard für eine Geste in Form des griechischen Buchstabens Phi

definiert werden, wie einzelne Gestenzüge und irrelevante Bewegungen voneinander unterschieden werden, da hierfür eine Definition der Segmentierung notwendig wäre. Bei Oberflächengesten stellt sich dieses Problem typischerweise nicht, da die Segmentierung durch Auf- und Absetzen des Werkzeugs von der Oberfläche quasi eingebaut ist. Bei räumlichen Gesten müssen entsprechende technische Mechanismen implementiert werden, die irrelevante Bewegungen von Gestenzügen trennen (vgl. Kap. 4.1).

Abb. 7.19 zeigt ein Beispiel für eine Geste in Form des kleinen griechischen Buchstabens Phi. Die Pfeilspitze am unteren Ende gibt die Bewegungsrichtung des Werkzeugs vor (vgl. Kap. 5.1.4) und legt somit fest, wo die Geste begonnen und beendet wird. Wie dargestellt muss die graphische Repräsentation der Form keine *perfekte* Ausführung wiedergeben, sondern sollte sich stattdessen eher an der bestmöglichen Ausführung durch einen Benutzer orientieren. Sie kann somit auch ein Hinweis für den Schwierigkeitsgrad der Geste sowie die notwendige Präzision (bzw. Robustheit gegenüber Ungenauigkeiten) des in der Anwendung verwendeten Erkenners sein.

Abbildung 7.20: GestureCard für eine Geste ohne festgelegte Form, deren Start- und Endpunkt aber übereinstimmen müssen

Die Form einer Geste kann jedoch nicht immer explizit festgelegt werden. GestureCards unterstützen daher die Definition von Freiformgesten über das Schlüsselwort `FREE`, die dann mit weiteren Eigenschaften weiter ausspezifiziert werden können. Statt der Darstellung einer geometrischen Figur wird das Schlüsselwort dann in die graphische Repräsentation eingesetzt und ermöglicht eben jede denkbare Form. Abb. 7.20 zeigt ein Beispiel für eine GestureCard, die eine solche Freiformgeste definiert. Diese Geste wird allerdings durch Definition der Eigenschaft `Start` eingeschränkt: Sie muss an dem Punkt enden, an dem sie auch begonnen wurde. Ihre Form zwischen Start- und Endpunkt kann jedoch beliebig sein.

Abbildung 7.21: Symbol für unbewegliche Teilgesten in GestureCards

Außerdem müssen sich Teile von Gesten nicht zwingend bewegen (vgl. Kap. 4.1.2). Für Oberflächengesten reduziert sich eine Pose allerdings auf die Berührung der Oberfläche, sodass eine genaue Definition von Fingerpositionen wie bei räumlichen Gesten (vgl. Kap. 11.1) nicht notwendig ist. Hierzu wird das in Abb. 7.21 dargestellte Symbol verwendet, dass in die Sektion zur graphischen Repräsentation statt einer geometrischen Form eingefügt wird.

Die Definition der graphischen Sektion einer Geste ist ein Pflichtfeld, daher wird für die Form kein Standardwert festgelegt. Darüber hinaus ist eine Verwendung als Rückgabe nicht vorgesehen. Für die Beschreibung der Form

räumlicher Gesten, die dreidimensionale Bewegungen erlauben, sowie räumlicher Posen der Hand sind verschiedene Möglichkeiten denkbar. In Kap. 11 werden einige Optionen vorgeschlagen.

7.2.1.9 Geschwindigkeit

Gesten können in unterschiedlichen Geschwindigkeiten ausgeführt werden. Dadurch können insbesondere langsame Linien von schnellen Swipes unterschieden werden. In GestureCards wird die Geschwindigkeit mit dem Schlüsselwort `Speed` beschrieben. Da die Ausführungsgeschwindigkeit von verschiedenen Faktoren beeinflusst wird und es insbesondere für Benutzer sehr schwierig ist, die Geschwindigkeit konstant zu halten (vgl. Kap. 5.1.8), werden für GestureCards lediglich zwei mögliche Werte definiert, die durch zwei weitere Werte zur Definition der Beschleunigung ergänzt werden:

- `Normal`: eine normale Bewegung
- `Fast`: eine schnelle Bewegung
- `Increasing`: eine beschleunigende Bewegung
- `Decreasing`: eine bremsende Bewegung

Als Standardwert wird `Normal` angenommen. Das Schlüsselwort `Fast` ist für schnelle Wischbewegungen vorgesehen. Die Beschleunigung während der Ausführung bezieht sich auf die gesamte Ausführungszeit der Geste. Wird die Eigenschaft als Rückgabe verwendet, dann ist neben der exakten Geschwindigkeit auch eine Kategorisierung nach dem obigen Schema für die Anwendung relevant.

Alternativ kann die Geschwindigkeit explizit definiert werden, dann als Strecke/Zeit. Zu berücksichtigen ist, dass unterschiedliche Frameworks und unterschiedliche Gerätetypen verschieden genaue Möglichkeiten bieten, um die Geschwindigkeit zu erfassen. Daher sollte vom Projektteam – insbesondere zur Definition der Strecke – eine anwendungsabhängige Abstraktion definiert werden.

7.2.1.10 Multiplizität

In GestureCards können unterschiedliche Optionen zur Beschreibung von Wiederholungen definiert werden. Dazu wird das Schlüsselwort `Multiplicity` verwendet. Mögliche Optionen sind:

- `Once`: Die Geste wird exakt einmal ausgeführt
- `Repeat`: Die Geste kann beliebig oft wiederholt werden; die definierten Eigenschaften müssen auf jede Ausführung zutreffen
- Zahlwert: Die Anzahl der Ausführungen; kann mit den in Kap. 7.1.2 beschriebenen Operatoren versehen werden

Wird die Eigenschaft nicht definiert, so wird als Standardwert `Once` gesetzt. Bei der Verwendung der Multiplizität als Rückgabewert ist davon auszugehen, dass die exakte Anzahl der Durchführungen durch den Benutzer als für die weitere Verarbeitung in der Anwendung interessanter Wert erwartet wird. Außerdem wird beim Einsatz von Multiplizitäten davon ausgegangen, dass Anwendungen bei jeder Durchführung über die Geste informiert werden wollen. Wird die Eigenschaft `Multiplicity` also gesetzt, so ist eine Reaktion der Anwendung nach jeder Ausführung der Geste implizit definiert.

Die Definition der Multiplizität ist in zwei Fällen notwendig:

- Wenn in einer Anwendung Aktionen verwendet werden sollen, die durch die Anzahl der Ausführungen einer Geste bestimmt werden.
- Wenn eine Geste mehrfach hintereinander ausgeführt werden kann.

Ein Beispiel für letzteren Fall ist das Zurückspulen durch eine kreisförmige Bewegung gegen den Uhrzeigersinn in einer Anwendung, die Medien abspielt. Abb. 7.22 zeigt die Definition einer derartigen Geste mit GestureCards. Durch die Angabe von `Multiplicity: Repeat` wird definiert, dass die Geste beliebig oft wiederholt werden kann. Solange der Benutzer die Geste ausführt, soll zurückgespult werden. Im Beispiel wird außerdem die Eigenschaft `Speed` als Rückgabe verwendet; dies könnte bedeuten, dass je schneller der Benutzer die Geste ausführt, desto weiter soll zurückgespult werden. So könnte die Anwendung pro Ausführung der Geste je nach Geschwindigkeit z. B. 5, 10 oder 15 Sekunden zurückspulen. Die Anzahl der Wiederholungen selbst hat für diese Anwendung keine direkte Bedeutung.

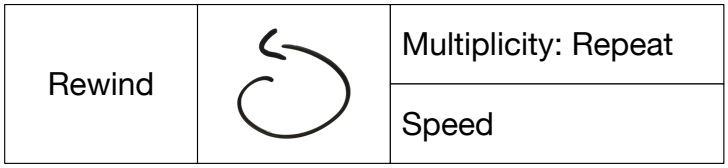


Abbildung 7.22: GestureCard, die eine kreisförmige mehrfach ausführbare Bewegung gegen den Uhrzeigersinn beschreibt

7.2.1.11 Ort

Kap. 5.1.10 hebt die Bedeutung des Ausführungsorts einer Geste hervor und diskutiert zahlreiche Möglichkeiten, diesen zu spezifizieren. Das in GestureCards verwendete Schlüsselwort ist `Location`. Mit `Location` können keine räumlichen Bezüge zu anderen Teilgesten definiert werden; dazu wird die Eigenschaft `Position` (vgl. Kap. 7.2.2.3) verwendet.

Wie beschrieben ist die einfachste Form die Definition eines Rasters von Planquadraten. Wie das Raster im Detail ausgestaltet wird, muss vom Projektteam in Abhängigkeit der zu entwickelnden Anwendung definiert werden. So verwenden z. B. Nancel et al. (2015) ein einfaches Raster bestehend aus zwei Feldern, jedoch sind auch feinere Raster möglich. Im Falle von Oberflächengesten wird das Raster über die verwendete Oberfläche gelegt. GestureCards erlauben die Definition von maximal drei Abstufungen des Rasters, so wie in Abb. 5.3 dargestellt. Vertikale Abstufungen sind dabei `Top`,

`Bottom` und `Center`, die entsprechend für den oberen, unteren und mittleren Teil der Oberfläche stehen. Horizontale Abstufungen sind `Left`, `Right` und `Middle` – der linke, rechte und mittlere Teil der Oberfläche. Zur genaueren Identifikation können vertikale und horizontale Abstufungen kombiniert werden: `Top Left` bezeichnet die Ecke links oben, `Bottom Right` die Ecke rechts unten, usw.

Das Raster kann – wenn für die Anwendung relevant – um das Gebiet um die Oberfläche herum erweitert werden. Zur Spezifikation wird dann das Schlüsselwort `Off`[7] verwendet. Dabei kann eine der o. g. Bezeichnungen zur Verfeinerung verwendet werden, z. B. bezeichnet `Location: Off` alle Bereiche außerhalb und `Location: Off Right` den äußeren Bereich rechts der Oberfläche.

[7] Als Pate für diese Bezeichnung standen Wobbrock et al. (2009), die zeigen, dass Benutzer einen Bereich außerhalb des Bildschirms (*off screen*) sinnvoll für Gesten einsetzen wollen.

Werden räumliche Gesten spezifiziert, so muss die zusätzliche Dimension berücksichtigt werden. Es gibt demnach Elemente vorne, hinten und auf gleicher Höhe. In GestureCards werden die Schlüsselworte `Front`, `Even` und `Back` verwendet. Zusätzlich muss bei räumlichen Gesten ggf. der Bezugspunkt mit angegeben werden. Der Bezugspunkt kann entweder der Benutzer (`User`) oder der Sensor bzw. der vom Sensor erfasste Raum (`Sensor`) sein. Diese Unterscheidung ist notwendig, da sich gerade bei Sensoren, die räumliche Gesten ermöglichen, die Konzepte und insbesondere Ausmaße des erfassten Bereiches unterscheiden können; u. U. ist auch denkbar, dass Anwendungen beide Varianten unterstützen. `Location: Sensor Top` bezeichnet also z. B. bei einer Anwendung, die auf dem Leap Motion Controller basiert, einen Bereich weiter oberhalb des Sensors, während `Location: Sensor Bottom` einen Bereich nahe des Sensors bezeichnet. Wird kein Bezugspunkt angegeben, so wird davon ausgegangen, dass der Ort ausgehend vom Benutzer spezifiziert wird.

Eine weitere Einschränkung kann mit einem Bezug zu einem virtuellem Objekt der Anwendung definiert werden (vgl. Kap. 7.1.5). Hierzu wird das Schlüsselwort `Object` verwendet, das anzeigt, dass eine Geste auf einem Objekt ausgeführt werden muss. Der Objektbezug kann mit o. a. Schlüsselworten erweitert und somit weiter eingeschränkt werden. `Location: Object Top Right` ermöglicht so, die rechte obere Ecke eines auf der Oberfläche dargestellten Objekts zu definieren, wodurch in unterschiedlichen Bereichen eines virtuellen Objekts verschiedene, durch Gesten aufrufbare Funktionalitäten integriert werden können. `Location: Object Off Top Right` hingegen bezeichnet durch die Angabe von `Off` einen Ausführungsort rechts oben, aber außerhalb eines Objekts. Abb. 7.23 zeigt ein Beispiel: Die Ausführung einer Geste in Form eines X wird mit der Eigenschaft `Location: Object` auf ein virtuelles Objekt beschränkt.

Darüber hinaus ist der Ort der Ausführung die einzige Eigenschaft, die mit dem Operator `NOT` expliziert negiert werden kann. Dieser wird typischerweise gemeinsam mit einem Verweis auf eine andere Geste eingesetzt, um z. B. deutlich zu machen, dass zwei Teilgesten nicht an gleicher Stelle ausgeführt werden sollen. Die Eigenschaft `Location: NOT T1` sagt demnach aus, dass eine Teilgeste nicht dort ausgeführt werden darf, wo die Teilgeste `T1` ausgeführt wurde.

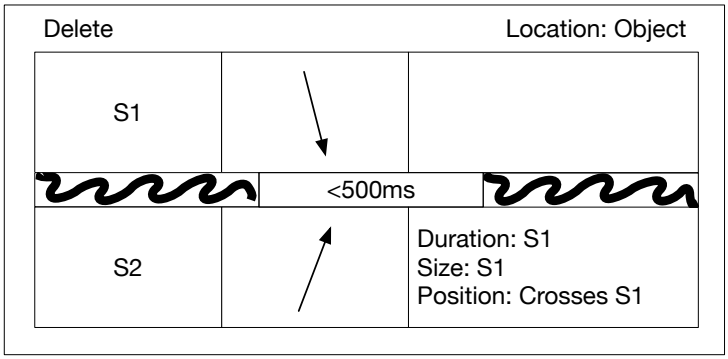


Abbildung 7.23: GestureCard für eine Geste in Form eines X, die auf einem Objekt ausgeführt werden muss

Wird der Ort nicht angegeben, so erfolgt keine Einschränkung des Ortes als standardmäßige Definition. Bei einer Verwendung als Rückgabe ist zwar der Ort der Ausführung das für die Anwendung interessante Merkmal, jedoch muss das Projektteam die genaue Ausprägung für die Anwendung definieren. Verschiedene Optionen existieren, die von der Definition der Geste sowie der Anwendung abhängen. Werden z. B. Planquadrate verwendet, so muss entschieden werden, ob eine Menge von Planquadraten oder das mit dem größten Anteil wesentlich ist. Mit Blick auf die Ungenauigkeit, mit der Benutzer Gesten ausführen, muss ihm Team auch definiert werden, wie mit dem Übertritt von Grenzen, z. B. beim der Einschränkung auf virtuelle Objekte, umgegangen wird.

7.2.1.12 Reaktion

Anwendungen können – wie in Kap. 5.1.11 beschrieben – auf zwei Arten auf Gesten reagieren: während die Geste ausgeführt wird oder nachdem die Geste vollständig ausgeführt wurde. Das in GestureCards verwendete Schlüsselwort ist `Reaction`. Für die beiden möglichen Optionen werden folgende Werte verwendet, die von Kammer et al. (2015) übernommen wurden:

- `online`: Reaktion zur Ausführungszeit
- `offline`: Reaktion nach Ausführung

Abb. 7.24 zeigt, wie die Eigenschaft verwendet wird. Wird die Reaktion nicht explizit definiert, so wird von einer Offline-Geste ausgegangen. Eine Verwendung der Reaktion als Rückgabewert ist nicht vorgesehen.

Abbildung 7.24: GestureCard für eine Geste, auf die die Anwendung während der Ausführung reagiert

In GestureCards wird lediglich definiert, *wann* eine Anwendung auf eine Geste reagiert, aber nicht *wie*, da damit die Funktionalität der Anwendung spezifiziert wird, die nicht im Fokus der Notation steht. Ungeachtet dessen kann mit GestureCards spezifiziert werden, welche Eigenschaften von der

Anwendung verarbeitet werden. Dazu ist in GestureCards der Rückgabewert in der vierten Sektion vorgesehen. Bei Online-Gesten ist davon auszugehen, dass die in der Rückgabe definierte Eigenschaft kontinuierlich verarbeitet wird. Einzige Ausnahme ist die Eigenschaft `End` (wie in Abb. 7.24 verwendet), da der Endpunkt einer Geste erst am Ende der Ausführungszeit bekannt ist.

7.2.1.13 Rotation

Durch die Definition der Rotation werden Drehungen der Werkzeuge in sich beschrieben, d. h. dass das Werkzeug während der Ausführung gedreht wird. Dazu wird das Schlüsselwort `Rotation` verwendet. Angegeben werden muss eine Drehrichtung, die entweder im Uhrzeigersinn (`Clockwise`) oder gegen den Uhrzeigersinn (`Counterclockwise`) verlaufen kann. Für Oberflächengesten sind zwei Drehrichtungen ausreichend (vgl. Abb. 5.7), räumliche Gesten erlauben jedoch sechs Drehrichtungen (zwei je Achse, vgl. Abb. 5.8). Eine detaillierte Definition aller Richtungen scheint aber mit Blick auf die Einschränkungen der menschlichen Hand, die wie eingangs erläutert die Basis für die mit GestureCards zu definierenden Gesten ist, nicht sinnvoll, da diese anatomisch bedingt auch nicht in alle Richtungen gedreht werden kann. Die Drehrichtung kann durch einen numerischen Wert erweitert werden, der in Grad ausdrückt, wie weit gedreht wird.

Wird die Eigenschaft nicht angegeben, so wird erwartet, dass das Werkzeug nicht rotiert wird. Standardwert ist also `Rotation: 0°`. Bei der Verwendung wird die Rotation als Rückgabewert wird erwartet, dass die Anwendung mit dem Drehwinkel (in Grad) sowie der Drehrichtung weiterarbeitet.

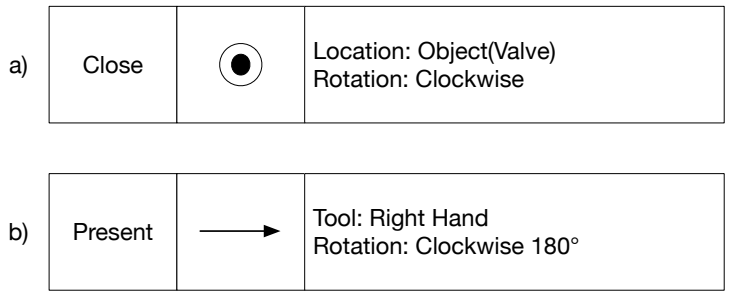


Abbildung 7.25: Beispiele für GestureCards, die Rotation verwenden: Drehung des Werkzeugs a) ohne Bewegung und b) mit Bewegung

Abb. 7.25 zeigt zwei Beispiele, die die Verwendung der Rotation verdeutlichen. In Beispiel a) wird das Werkzeug auf einem Objekt mit der Bezeichnung *Valve* im Uhrzeigersinn gedreht. Ein denkbares Szenario für diese Geste ist das Schließen eines Ventils in einem VR/AR-basiertem Spiel. In Beispiel b) wird definiert, dass die rechte Hand verwendet und bei der Durchführung einer Linie nach rechts um 180° im Uhrzeigersinn gedreht wird – ähnlich der Bewegung zum Präsentieren einer Sache.

7.2.1.14 Start- und Endpunkt

Start- und Endpunkt einer Geste sind Verfeinerungen des Ortes der Ausführung (vgl. Kap. 7.2.1.11) und folgen den gleichen Regeln. In GestureCards

kann somit explizit definiert werden, wo eine Geste starten bzw. enden soll. Die verwendeten Schlüsselwörter sind `Start` respektive `End`. Eine wiederkehrende Verwendung ist, dass Teilgesten am Start- bzw. Endpunkt anderer Teilgesten anfangen oder aufhören (z. B. in Abb. 7.30). Standardwerte für die beiden Eigenschaften sind nicht definiert, d. h. wenn die Eigenschaft weggelassen wird, dann können Gesten überall starten bzw. enden. Bei einer Verwendung als Rückgabe wird davon ausgegangen, dass die Anwendung mit dem Start- bzw. Endpunkt weiterarbeitet.

Abb. 7.26 zeigt beispielhaft die Verwendung des Startpunkts anhand einer auf mobilen Geräten weit verbreiteten Geste. Dargestellt wird, wie unter Android und iOS das Nachrichten-Center geöffnet werden kann. Dazu wird eine Geste in Form einer senkrechten Bewegung von oben nach unten verwendet, die außerhalb des Bildschirms startet.

Start- und Endpunkt können außerdem noch eine räumliche Relation haben, also mit Bezug zu anderen Gesten positioniert werden, oder eine Distanz zu anderen Gesten aufweisen. Dazu können auch die in Kap. 7.2.2.3 und Kap. 7.2.2.4 definierten Schlüsselworte einbezogen werden und Start- und Endpunkt genauer verfeinern.

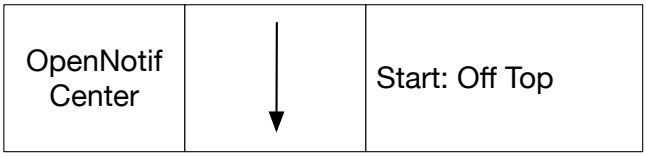


Abbildung 7.26: GestureCard für die Geste zum Öffnen des Nachrichtencenters auf gängigen mobilen Geräten

7.2.1.15 Werkzeug

GestureCards erlauben die Spezifikation unterschiedlicher Werkzeuge, mit denen Gesten ausgeführt werden. Das verwendete Schlüsselwort ist `Tool`. Grundlegend können folgende Werkzeuge definiert werden, die unterschiedlich spezialisiert werden können:

- `Finger`: Ein beliebiger Finger
- `Hand`: Eine beliebige Hand
- `Pen`: Ein Stift
- `Device`: Ein Gerät/Sensor (z. B. Smartphone, Magic Wand usw.)

Die Werkzeuge können durch Zusätze weiter verfeinert werden. So können z. B. durch `Left` und `Right` die jeweiligen Hände angesprochen werden. Außerdem können einzelne Finger (`Thumb`, `Index`, usw.) definiert werden. Empfohlen wird, Zusätze mit sprechenden Formulierungen zu versehen (z. B. `Left Thumb`), aber andere Varianten (z. B. `Thumb.Left`) sind denkbar und dem Projektteam überlassen. Die Menge an potentiellen Werkzeugen, die zur Ausführung von Gesten eingesetzt werden kann, ist denkbar groß (vgl. Kap. 5.1.13), jedoch erlauben GestureCards durch die Verwendung von Name-

Wert-Paaren eine einfache Erweiterung durch weitere Schlüsselworte, die vom Projektteam festgelegt werden können.

Wird die Eigenschaft nicht explizit festgelegt, so kann die definierte Geste mit jedem Werkzeug ausgeführt werden. Bei der Verwendung als Rückgabe wird davon ausgegangen, dass das eingesetzte Werkzeug die für die Anwendung relevante Information ist. Allerdings macht dies nur Sinn, wenn das Werkzeug nicht in der Geste festgelegt ist.

Die Möglichkeit, das verwendete Werkzeug explizit zu definieren, erlaubt es auch, Gesten zu spezifizieren, die zwei verschiedene Werkzeuge zugleich einsetzen. So definiert Abb. 7.27 eine Geste, bei der ein Finger und Stift zeitgleich verwendet werden.

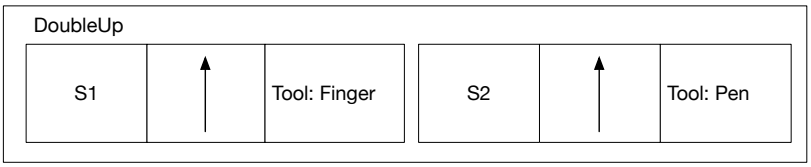


Abbildung 7.27: GestureCard für eine Multi-Touch-Geste, bei der ein Finger und ein Stift parallel verwendet werden

7.2.2 Beziehungen zwischen Teilgesten

In GestureCards wird zwischen der *zeitlichen* und *räumlichen* Beziehung zwischen Teilgesten unterschieden. Die räumliche Beziehung wird dabei in weitere Eigenschaften aufgeteilt: die Positionierung von Teilgesten zueinander, ihre Entfernung voneinander und ein Größenverhältnis. Da sie als unterschiedliche Eigenschaften definiert sind, können alle gemeinsam verwendet werden.

7.2.2.1 Zeitliche Relation

Teilgesten stehen in einer zeitlichen Relation zueinander (vgl. Kap. 5.2.1). Diese kann in der Notation über die visuelle Anordnung der betreffenden GestureCards definiert werden: Sie können untereinander angeordnet werden, um sequentielle Gestenzüge zu beschreiben, oder nebenbeinander, um parallele Gestenzüge darzustellen.

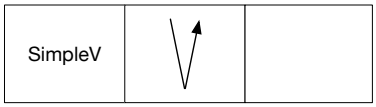


Abbildung 7.28: Alternative GestureCard für eine V-förmige Geste, die in einem Zug durchgeführt wird

Des Weiteren können zeitliche Differenzen – so wie von Cirelli & Nakamura (2014) gefordert – explizit modelliert werden. Im Folgenden werden die in Kap. 5.2.1 als Beispiele verwendeten Variationen einer V-förmigen Geste mit GestureCards beschrieben, um die Möglichkeiten aufzuzeigen. Abb. 7.28 zeigt die einfachste, einzügige Variation, für die eine Definition in der graphischen Repräsentation ausreichend ist. Definierte Eigenschaften gelten in diesem Falle für die gesamte Geste. Es ist außerdem unerheblich, ob es sich um eine Oberflächengeste oder eine räumliche Geste handelt – die Spezifikation ist in beiden Fällen identisch.

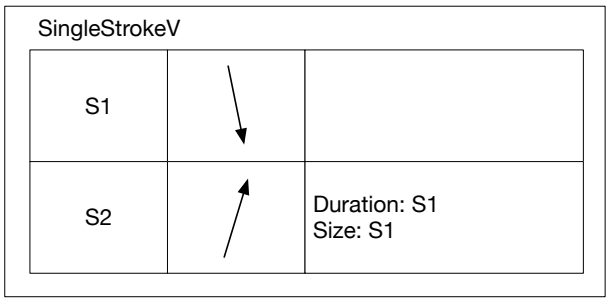


Abbildung 7.29: GestureCard für eine V-förmige Geste, die in einem Zug durchgeführt wird

Wenn für beide Gestenzüge unterschiedliche Eigenschaften definiert werden sollen, dann muss die Geste in Teilgesten aufgeteilt werden, wie in Abb. 7.29 dargestellt. Im Falle einer einzügigen Geste – bei der Teilgesten direkt nacheinander ausgeführt werden, ohne z. B. den Finger von der berührungsempfindlichen Oberfläche zu heben – werden die Teilgesten untereinander dargestellt und miteinander verbunden. Abb. 7.29 zeigt, wie das Beispiel aus Abb. 7.28 erweitert werden muss. Die Geste besteht aus zwei Teilgesten, die die beiden Schenkel des V repräsentieren. Während der erste Schenkel recht frei ausgeführt werden kann – lediglich die Bewegung nach rechts unten ist definiert –, muss der zweite Schenkel in Abhängigkeit des ersten ausgeführt werden. Hierzu definiert die GestureCard explizit, dass Dauer und Größe der des ersten Schenkels entsprechen müssen. Bei einzügigen Gesten ist eine Trennung in mehrere Teilgesten nur dann zwingend notwendig, wenn sich die Eigenschaften der Teilgesten unterscheiden[8] und somit explizit definiert werden müssen.

[8] Zum Beispiel wenn die Teilgesten unterschiedlich schnell ausgeführt werden wie in Abb. 7.5

Sequentielle Züge werden in GestureCards immer untereinander definiert. Das gilt auch für mehrzügige Gesten. Die in Abb. 7.29 dargestellte einzügige Geste kann einfach in eine mehrzügige Geste umgewandelt werden: Hierzu muss deutlich gemacht werden, dass es eine Zeitspanne zwischen beiden Gestenzügen gibt, in der die Ausführung pausiert. Für Oberflächengesten bedeutet dies, dass das verwendete Werkzeug von der Oberfläche abgehoben wird; es reicht nicht aus, die Bewegung zu unterbrechen und den Finger auf der Oberfläche ruhen zu lassen. Für räumliche Gesten, bei der mangels Kontakt zu einer Oberfläche keine scharfe Trennung zwischen Gestenzügen existiert, bedeutet dies, dass bis zur Erkennung der nächsten, gültige Geste, alle Bewegungen ignoriert werden sollen. In der Notation wird dazu eine geschlängelte Linien verwendet, die zwischen den GestureCards der Teilgesten eingefügt wird. Abb. 7.30 a) zeigt ein Beispiel.

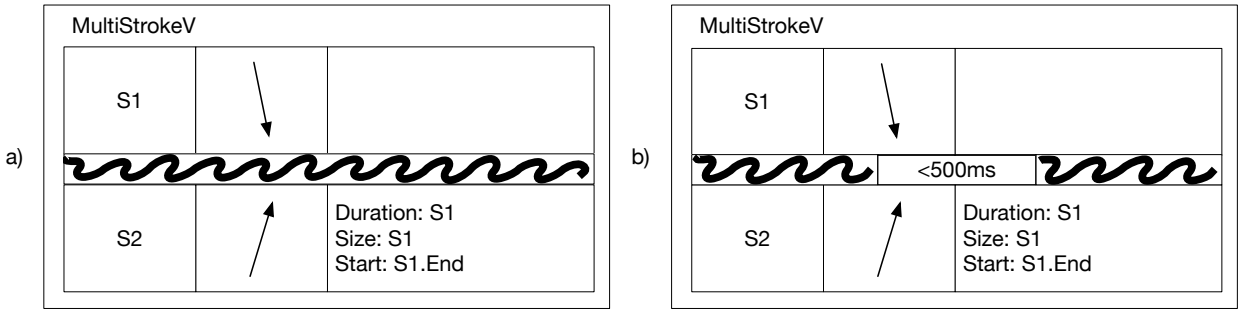


Abbildung 7.30: GestureCard für eine V-förmige Geste, die in zwei Zügen durchgeführt wird; a) ohne zeitliche Begrenzung, b) mit zeitlicher Begrenzung

In Abb. 7.30 a) wird lediglich definiert, dass zwischen beiden Zügen eine gewisse Zeitspanne vergehen soll, jedoch nicht, wie lange diese sein soll. Dazu kann die Notation um eine Zeitangabe erweitert werden, die ebenso wie andere Eigenschaften mit einem der in Kap. 7.1.2 definierten Operatoren versehen werden kann. Abb. 7.30 b) demonstriert, wie die Zeitspanne zwischen zwei Zügen auf maximal 500 ms begrenzt werden kann. Grundsätzlich gilt, dass die Definition einer Pause bedeutet, dass diese auch in der Ausführung der Geste berücksichtigt werden muss, d. h. auch wenn keine Zeitspanne angegeben ist, dann muss – im Falle einer Oberflächengeste – der Finger angehoben und wieder aufgesetzt werden. Im Falle räumlicher Gesten ist eine Pause nicht ganz so einfach zu definieren. Es liegt hier im Ermessen des Projektteams, wie lange und ob der Benutzer die Pause zwischen den Gestenzügen berücksichtigen muss.

Allerdings muss im Falle einer mehrzügigen Geste die Teilgeste `S2` um eine explizite Angabe des Startpunkts erweitert werden, andernfalls wäre jede mehrzügige Kombination zweier Teilgesten, bei der beide den dargestellten Formen entsprechen, als korrekt definiert. Die Einschränkung des Startpunkts der Teilgeste `S2` auf den Endpunkt der Teilgeste `S1` stellt sicher, dass die V-Form erhalten bleibt.

Bisher wurden Sequenzen von Teilgesten betrachtet, die hintereinander ausgeführt werden. Sowohl Oberflächengesten als auch räumliche Gesten ermöglichen aber – je nach verwendeter Sensoren – die parallele (gleichzeitige) Ausführung von Gestenzügen. Die parallele Ausführung von Teilgesten wird in der Notation durch eine Positionierung der betreffenden GestureCards nebeneinander spezifiziert. Abb. 7.31 zeigt, wie das Beispiel einer V-förmigen Geste so dargestellt werden kann, dass beide Schenkel parallel ausgeführt werden. Die Schenkel werden durch zwei Werkzeuge – hier die linke und die rechte Hand – durchgeführt.

Die Parallelisierung von Gestenzügen ermöglicht weitere zeitliche Abweichungen, wie in Kap. 5.2.1 dargestellt. Teilgesten können später und früher anfangen und enden als andere Teilgesten. Zeitlicher Verzug paralleler Teilgesten kann in GestureCards explizit modelliert werden, indem die für mehrzügige Gesten eingeführte Notation der geschlängelten Linie verwendet wird. Auch in diesem Falle kann durch eine optionale Zeitangabe die Definition weiter geschärft werden. Hierbei muss beachtet werden, dass parallele Teilgesten aufgrund der Ungenauigkeit der Ausführung immer mit einem zeitlichen Unterschied starten bzw. enden. Das heißt, auch wenn kein zeitlicher Verzug definiert ist, so ist davon auszugehen, dass dieser bei der Ausführung auftreten wird. Die explizite Modellierung des Verzuges dient dazu, weiterführende Möglichkeiten in die Spezifikation zu integrieren, die bei der Ausführung abgesehen von der üblichen Ungenauigkeit berücksichtigt

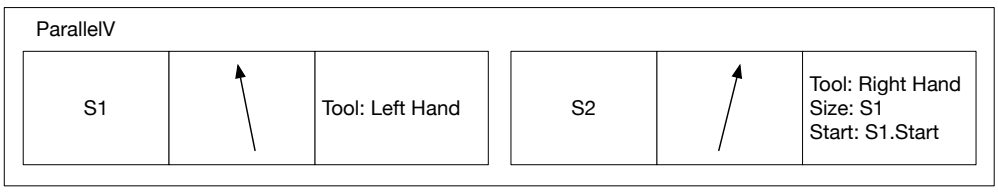


Abbildung 7.31: GestureCard für eine V-förmige Geste, die in zwei parallen Zügen durchgeführt wird

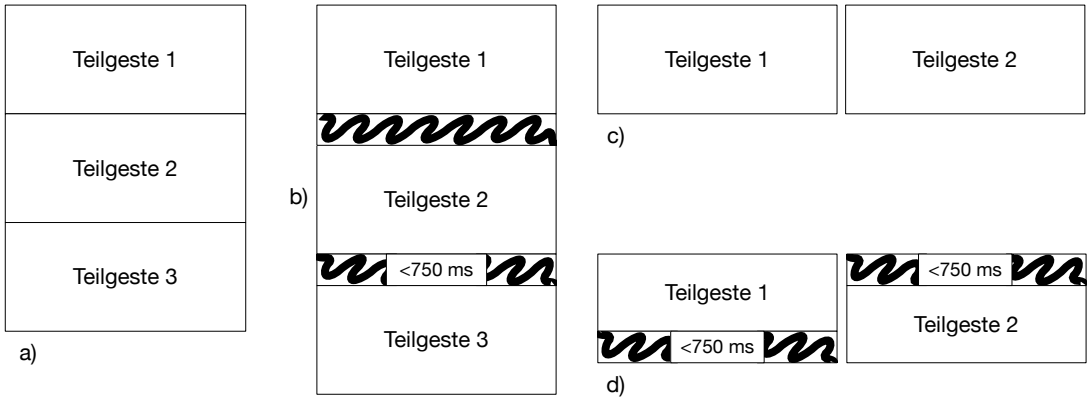


Abbildung 7.32: Gegenüberstellung der Optionen zur Darstellung der zeitlichen Relation in GestureCards: a) einzügige Gesten, b) mehrzügige Gesten, c) Parallelisierung, d) Versatz von Start- und Endzeitpunkten paralleler Teilgesten

werden müssen. Abb. 7.32 fasst die Möglichkeiten zur Definition der zeitlichen Relation von Teilgesten zusammen und zeigt in d), wie unterschiedliche Start- und Endzeitpunkte paralleler Teilgesten definiert werden können. Dabei ist zu beachten, dass die GestureCards der parallel ausgeführten Teilgesten und eventuell definierte zeitliche Verzüge in der gleichen Höhe dargestellt werden, um Verwirrung hinsichtlich der Dauer zu vermeiden.

Die zeitliche Relation muss explizit modelliert werden, daher ist kein Standardwert vorgesehen. Darüber hinaus kann die zeitliche Relation aus dem gleichen Grunde nicht als Rückgabe verwendet werden.

7.2.2.2 Größe

Die Aufteilung von Gesten in Teilgesten verschleiert u. U. die Größenverhältnisse zwischen den Teilgesten. Somit kann es sinnvoll sein, bei der Definition von Teilgesten die Eigenschaft `Size` zu verwenden, die als Wert einen Vergleich mit einer anderen Teilgeste erwartet. Damit kann definiert werden, dass Teilgesten die gleiche oder, bei Einsatz der in Kap. 7.1.2 definierten Operatoren, unterschiedliche Größen haben müssen. Die verschiedenen Beispiele in Kap. 7.2.2.1 verdeutlichen die Verwendung. Die Eigenschaft beschreibt ausschließlich ein Größenverhältnis zwischen zwei Teilgesten und kann nicht zur Einschränkung der Größe einer einzelnen Geste in Form einer quantitativ oder qualitativ definierten, atomaren Eigenschaft verwendet werden.

Die Größe muss nur angegeben werden, wenn der Größenunterschied zwischen Teilgesten explizit definiert werden soll, z. B. um Einschränkungen zu spezifizieren. Standardmäßig wird keine Angabe der Größe erwartet und Teilgesten können beliebig groß ausgeführt werden. Eine Verwendung als Rückgabe ist nicht vorgesehen.

Auf eine Verwendung der Größe als atomare Eigenschaft wird in GestureCards verzichtet, obwohl auf Hardware, die genügend Platz bietet (z. B. digitale Whiteboards), eine Differenzierung in unterschiedliche Größen denkbar ist, um Funktionalitäten mit verschieden großen Gesten aufzurufen oder zumindest mit der Größe der Geste in der Anwendung weiterzuarbeiten. Dazu könnte die Größe als atomare Eigenschaft definiert und mit vorgegebenen Schlüsselwörtern versehen werden (z. B. `Size: Large`). So könnte zwi-

schen kleinen und großen Gesten unterschieden werden. Eine Alternative wäre die Ausrichtung am Sensor selbst, z. B. könnte eine Geste mindestens ein Drittel des überwachten Bereichs umfassen (`Size: 1/3`).

7.2.2.3 Räumliche Relation

Teilgesten können im Raum zueinander positioniert werden, d. h. die Position der einen Teilgeste kann in Abhängigkeit von einer anderen Teilgeste definiert werden. In GestureCards ist es möglich, die Positionierung auf zwei Arten zu definieren: zum einen in der graphischen Repräsentation und zum anderen explizit als Eigenschaft mit dem Schlüsselwort `Position`. Die Eigenschaft ist nur für zusammengesetzte Gesten relevant. `Position` unterscheidet sich von der Eigenschaft `Location` (vgl. Kap. 7.2.1.11) durch die Abhängigkeit von einer anderen Geste. Während die Eigenschaft `Location` vom verwendeten Sensor, dem Benutzer oder einem virtuellen Objekt abhängt, bezieht sich die Definition der Eigenschaft `Position` stets auf andere Teilgesten, daher muss immer eine andere Teilgeste in der Definition angegeben werden. Dadurch können beide Eigenschaften mit unterschiedlichem Fokus verwendet werden, wodurch die einzelnen Ausdrücke einfacher zu lesen sein sollten. Wird die Eigenschaft nicht angeben, so ist die Positionierung von Teilgesten zueinander nicht definiert und Gesten können überall ausgeführt werden. Eine Verwendung als Rückgabe ist nicht vorgesehen.

Teilgesten können bei der Definition von Oberflächengesten über, unter, links oder rechts von anderen Teilgesten positioniert werden. Entsprechend werden die Schlüsselwörter `Above`, `Below`, `Left of` und `Right of` verwendet. Die Definition der Eigenschaft muss von einer Teilgeste vervollständigt werden. Die Positionierung erfolgt immer auf einer Linie, d. h. senkrecht über/unter bzw. horizontal links/rechts der anderen Geste. Zu beachten ist allerdings weiterhin, dass diese Angabe nur ungefähr sein kann, da die Ausführung solch einer Geste Ungenauigkeiten unterworfen ist. In Abb. 7.33 ist z. B. beschrieben, dass die Teilgeste `Spread` rechts von der Teilgeste `Point` positioniert sein muss. Räumliche Gesten erlauben darüber hinaus eine Positionierung von Teilgesten vor oder hinter anderen Teilgesten. Die Eigenschaft `Position` wird entsprechend um die beiden Schlüsselwörter `In front of` und `Behind` ergänzt. Analog zu den eingangs genannten Schlüsselwörtern gilt auch dies auf einer Linie zur anderen Teilgeste. Allerdings ist im Falle räumlicher Gesten die Abhängigkeit vom Sensor zu beachten. Je nach Position und Möglichkeiten des Sensors, die Umgebung zu erkennen, können Werkzeuge verdeckt werden und sind so für den Sensor nicht mehr sichtbar. Die Spezifikation mit GestureCards kann aber eingesetzt werden, um eben solche Fälle zu identifizieren (vgl. Kap. 13.2).

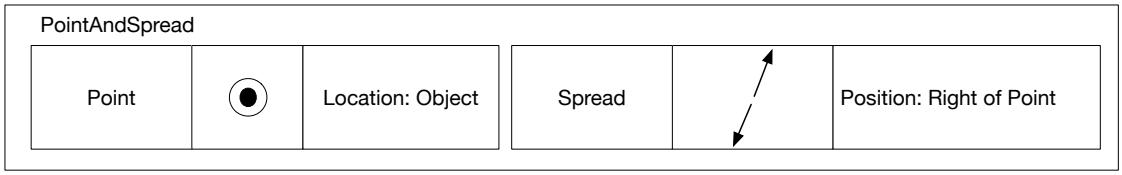


Abbildung 7.33: GestureCard zur Definition einer Geste, bei der eine Teilgeste rechts von einer anderen platziert wird

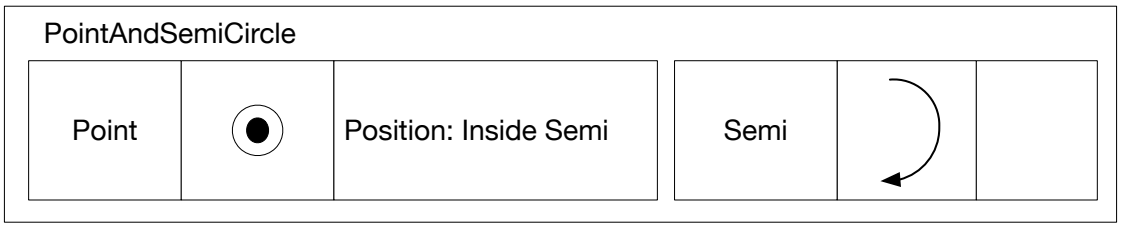


Abbildung 7.34: GestureCard zur Definition einer Geste, bei der die Teilgeste Point von der Teilgeste Semi eingeschlossen wird

Gestenzüge können sich darüber hinaus kreuzen. In GestureCards wird hierzu dass Schlüsselwort `Crosses` verwendet. Die Definition der räumlichen Beziehung als `Position: Crosses S1` bedeutet demnach, das die definierte Geste den Pfad der Geste S1 kreuzt. Des Weiteren können Teilgesten von anderen umschlossen werden, auch teilweise. Das verwendete Schlüsselwort ist `Inside`, gefolgt von der umschließenden Geste. Abb. 7.34 zeigt die Definition einer GestureCard für eine Geste, bei der der in der Teilgeste Semi beschriebene Halbkreis die andere Teilgeste umschließt.

Andere Autoren nehmen in der räumlichen Beziehung noch Bewegungsoptionen auf. So definiert z. B. Kammer (2014) in GeForMT, dass zur räumlichen Beziehung auch Werte wie `Split` und `Join` gehören, die ausdrücken, dass sich Teilgesten voneinander entfernen bzw. aufeinander zu bewegen. In GestureCards wird dies durch Definition der Bewegungsrichtung (vgl. Kap. 7.2.1.5) dargestellt. Außerdem definiert Kammer (2014) noch die räumlichen Relationen `Close`, `Connect_End` und `Connect_Start`, die in GestureCards durch Definition der Start- und Endpunkte abgebildet werden.

Die Beschreibung der räumlichen Relation ist in der Notation bewusst einfach gehalten, ermöglicht aber dadurch Spielraum für Ungenauigkeiten. Diese können mit einigen potentiellen Erweiterungen ausgeräumt werden. Aktuell können einige Positionen, z. B. *links unten*, nicht exakt definiert werden. Eine denkbare Erweiterung, neben passenden Schlüsselwörtern, könnte die Verwendung eine Kompassrose sein, die über die Geste gelegt wird und mit der (analog zur Angabe der Bewegungsrichtung in GeForMT, vgl. Kap. 6.2.4) eine Geste *südöstlich* – eben links unten – platziert werden könnte. Der Einsatz einer Kompassrose hätte mehrere Vorteile:

- Verfeinerungen, z. B. zu *südsüdöstlich*, könnten vom Projektteam bestimmt werden
- Ungenauigkeiten könnten mit dem Intervall-Operator festgelegt werden, um auszudrücken, dass eine Teilgeste z. B. ungefähr im Bereich von Norden bis Osten positioniert sein soll

Beim Schlüsselwort `Crosses` wird aktuell nicht definiert, an welcher Stelle die andere Geste gekreuzt wird. Dies könnte in Zukunft in der graphischen Repräsentation mit einem Symbol, z. B. in Form eines X, markiert werden.

7.2.2.4 Distanz

Neben der räumlichen Positionierung zueinander kann in GestureCards auch die Distanz zwischen verschiedenen Teilgesten definiert werden. Diese kön-

nen weit entfernt oder auch direkt nebeneinander ausgeführt werden. Die Distanz wird durch das Schlüsselwort `Distance` festgelegt. Folgende Werte können eingesetzt werden:

- `Far From`: eine Teilgeste befindet sich weit weg von einer anderen Gestew
- `Close To`: eine Teilgeste wird in der Nähe einer anderen Teilgeste durchgeführt
- `Touches`: Teilgesten werden direkt nebeneinander durchgeführt

Die genauen Ausprägungen (und eventuelle Verfeinerungen) der Distanz sind abhängig von der zu entwickelnden Anwendung und den verwendeten Sensoren – auf einem großen digitalen Whiteboard sind andere Werte zugrunde zu legen als auf einem kleinen Smartphone. Das Projektteam muss hier eine entsprechende Entscheidung treffen. Zur Orientierung kann die Größe der Oberfläche oder des überwachten Raumes herangezogen werden: Gesten, die weiter von aneinander entfernt sind als die Hälfte der Länge/Breite der Oberfläche/des Raumes sind weit auseinander (`Far From`), während Gesten, die weniger als die Hälfte der Oberfläche voneinander entfernt sind, nah beieinander sind (`near to`). Berührung sich die Werkzeuge, z. B. wenn zwei Finger direkt nebeneinander über die Oberfläche geführt werden, wird die dritte Ausprägung (`Touches`) verwendet.

Generell soll aber – wie bei den meisten definierbaren Eigenschaften – vom Benutzer und seinen Möglichkeiten ausgegangen werden. Muss der Benutzer die Arme weiter als Schulterbreite öffnen, um die Geste auszuführen, so ist die Distanz zwischen den Teilgesten groß (`Far From`). Sind die Arme schulterbreit geöffnet, so ist die Distanz gering (`Close To`). Sind die Hände so nah beieinander, dass sie sich berühren, so ist die letzte Option zu nehmen (`Touches`). Abb. 7.35 veranschaulicht dies am Beispiel eines Benutzers an einem digitalem Whiteboard mit großem Multi-Touch-Bildschirm.

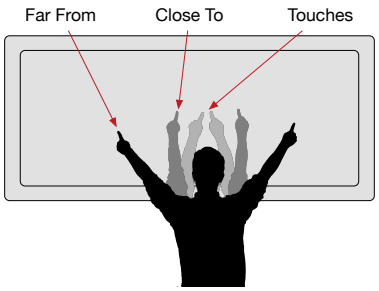


Abbildung 7.35: Distanzen zwischen Teilgesten auf einem digitalen Whiteboard

8 Lesbarkeit und Verständlichkeit textueller und hybrider Notationen

Die Menge der Gesten, die mit einer Notation beschrieben werden können, ergibt sich direkt aus den Möglichkeiten, die die Notation zur Beschreibung von Eigenschaften bietet: Je mehr und je exakter Eigenschaften beschrieben werden können, desto mehr Gesten können spezifiziert werden. Neben der Fähigkeit einer Notation, möglichst viele Gesten möglichst exakt zu beschreiben, sind im Entwicklungsprozess weitere Faktoren ausschlaggebend. Einer davon ist die *Verständlichkeit* einer Notation, die sich insbesondere durch verwendete Schlüsselwörter und die zugrundeliegende Struktur bemerkbar macht. Sie ist damit insbesondere interessant, wenn die Notation die Grundlage für Diskussion und Dokumentation darstellt. Vor allem in cross-funktionalen Teams, die unter Einbeziehung unterschiedlicher Stakeholder arbeiten, müssen verwendete Notationen schnell verständlich und ohne spezielles Wissen über die Eigenarten – also Syntax und grundlegender Aufbau – erfasst werden können. Damit ist neben der Lesbarkeit auch die Auffassungsgeschwindigkeit für die Verständlichkeit relevant.

In Kap. 5 wurde gezeigt, dass im Laufe der letzten Jahre zahlreiche Autoren viele unterschiedliche Eigenschaften von Gesten auf ihre Verwendung für HCI hin untersucht haben. Darüber hinaus wurden in Kap. 6 unterschiedliche Ansätze zur Beschreibung von Gesten vorgestellt. Diese folgen vorwiegend – im Gegensatz zur im Rahmen dieser Arbeit entwickelten Notation – streng einem zugrundeliegenden Paradigma: Sie setzen entweder rein textuelle oder rein graphische Elemente ein und lassen denkbare Vorteile, die hybride Notationen bieten, außer Acht. In der nachfolgend beschriebenen Studie soll zunächst verglichen werden, wie Probanden textuelle und hybride Notationen auffassen. Auf graphische Notationen wird in der in Kap. 9 beschriebenen Studie eingegangen. Der Studie liegt folgende Forschungsfrage zugrunde:

Hybride Notationen ermöglichen es, gezielt graphische oder textuelle Mittel einzusetzen, um Informationen über Gesten darzustellen. Werden hybride Notationen dadurch von ungeübten Probanden schneller und besser verstanden als textuelle Notationen?

In dieser Studie werden GestureCards als hybride Notation und eine textuelle Notation mit Blick auf die obige Fragestellung miteinander verglichen. Die Studie zeigt, dass diese Frage für die verglichenen Notationen bejaht werden muss: Probanden, die mit GestureCards arbeiteten, zeigten durchweg bessere Ergebnisse als Probanden, die mit der textuellen Vergleichsnotation arbeiteten. Sie waren schneller fertig, konnten mehr Gesten korrekt identifizieren und mussten seltener auf zur Verfügung gestellte Dokumentation zurückgreifen. Darüber hinaus waren sie etwas zuversichtlicher, die dargestellten Gesten korrekt identifiziert zu haben. Neben diesen messbaren Ergebnissen deuten auch die Kommentare und Aktionen der Probanden während der Studie

darauf hin. So drückten die Probanden, die mit GestureCards arbeiteten, mehr Zuversicht aus und verwechselten seltener Elemente der Notation.

Zunächst werden die Grundlagen der Studie dargelegt und der allgemeine Aufbau beschrieben (Kap. 8.1). Anschließend werden die Ergebnisse der Studie mit Blick auf die zur Messung des Verständnisses definierten Punkte im Detail betrachtet (Kap. 8.2). Dabei werden zunächst die gemessenen Daten quantitativ analysiert und anschließend über Beobachtungen bei der Durchführung und verschiedene Kommentare der Probanden berichtet. Danach werden die Ergebnisse diskutiert und denkbare Schlussfolgerungen aus den Ergebnissen gezogen (Kap. 8.3). Abschließend werden Risiken betrachtet, die die Validität der Ergebnisse potentiell gefährden können, und erläutert, wie versucht wurde, die Risiken durch den Aufbau der Studie zu minimieren (Kap. 8.4).

8.1 Aufbau der Studie

Die Beschreibung des Studienaufbaus folgt im Wesentlichen der Struktur, die Jedlitschka et al. (2008) vorgeschlagen haben. Es werden die Ziele der Studie, die teilnehmenden Probanden, die zu lösenden Aufgaben – im vorliegenden Fall also die verwendeten Gesten –, die eingesetzte Anwendung, die erfassten Variablen, die Vorgehensweise während der Studie sowie die Vorgehensweise zur Analyse der erfassten Daten beschrieben.

8.1.1 Ziele

In der Studie wird verglichen, ob und wie ungeübte Probanden Gesten identifizieren können, die in einer textuellen oder einer hybriden Notationen verfasst sind. Ungeübt bedeutet in diesem Zusammenhang, dass die Probanden über kein detailliertes Wissen über die Beschreibung von Gesten im Allgemeinen und die verwendeten Notationen im Speziellen verfügten, aber grundlegende Erfahrung mit Software-Entwicklung und der Notwendigkeit von Spezifikation haben. Die erfassten Daten sollen zeigen, wie Probanden unterschiedliche Notationen verstehen und somit Aufschluss über die Verständlichkeit der eingesetzten Notationen geben. Basis für die Messung des Verständnisses sind:

- Die Anzahl korrekt identifizierter Gesten
- Die Häufigkeit, mit der Probanden auf zur Verfügung gestellte Dokumentation der jeweiligen Notation zurückgreifen
- Die Zuversicht, mit der Probanden Gesten aus der jeweiligen Notation identifizieren können
- Die Geschwindigkeit, mit der Probanden in der jeweiligen Notation formulierte Gesten identifizieren können

Der Forschungsfrage folgend ist zu erwarten, dass alle diese Faktoren bei hybriden Notationen bessere Ergebnisse liefern als bei textuellen Notationen.

Dementsprechend sollten Probanden mit hybriden Notationen mehr Gesten korrekt identifizieren, seltener auf die zur Verfügung gestellte Dokumentation der Notation zugreifen, zuversichtlicher bei der Bewertung ihrer Ergebnisse sein und das Experiment schneller abschließen.

GestureCards als hybride Notation werden in der Studie mit der textuellen Notation GeForMT (vgl. Kap. 6.2.4) verglichen. GeForMT wurde aus folgenden Gründen gewählt:

- Kammer et al. (2015) berichten, dass GeForMT im Vergleich zu anderen textuellen Gestennotationen von Probanden als prägnanter und leserlicher bewertet wurde. Daraus kann abgeleitet werden, dass sie auch einfacher zu verstehen ist.

- Der Ansatz, der GeForMT zugrundeliegt, ist messbar minimalistischer als vergleichbare textuelle Notationen. Kammer et al. (2015) zeigen, dass bei GeForMT der syntaktische *Overhead* – also der Anteil notwendiger Zeichen, die zur Strukturierung des Ausdrucks dienen (z. B. eckige und runde Klammern, Operatoren usw.) – geringer ist und somit ein stärkerer Fokus auf die beschriebene Geste als auf notwendige syntaktische Regeln der zugrundeliegenden Notation erkennbar ist.

- GeForMT wurde erfolgreich in eigenen Arbeiten eingesetzt (Hesenius et al., 2014a,b; Hesenius et al., 2016).

8.1.2 Probanden

Die teilnehmenden Probanden wurden für die Studie zufällig in zwei Gruppen eingeteilt. Zwischen beiden Gruppen gab es keine Überschneidungen, d. h. jeder Proband war Mitglied genau einer Gruppe. Jeder Gruppe wurde eine Notation zugeordnet und die Probanden der Gruppen arbeiteten ausschließlich mit der jeweiligen Notation. Somit arbeitete kein Proband mit beiden Notationen, sondern ausschließlich mit der, der seiner Gruppe zugeordnet wurde. Die Gruppen werden im Folgenden durch die verwendete Notation identifiziert; demnach gab es die Gruppe GeForMT und die Gruppe GestureCards.

An der Studie nahmen 14 Probanden teil; 12 waren männlich und zwei weiblich. Die Probanden wurden in der Reihenfolge ihrer Teilnahme an der Studie abwechselnd auf die beiden Gruppen aufgeteilt, sodass die Zuordnung im Wesentlichen zufällig und ohne Berücksichtigung weiterer Faktoren geschah. Alle hatten einen Hintergrund im SE auf unterschiedlichem Niveau (vom Bachelor- bis zum Promotionsstudium; einige Probanden hatten auch mehrjährige Berufserfahrung) und keine bis kaum Erfahrung mit Gestennotationen. Die Probanden wurden in zwei Wellen akquiriert: In der ersten Welle konnten zehn Freiwillige gewonnen werden, die ohne weitere Belohnung an der Studie teilnahmen; in der zweiten Welle wurden vier weitere Probanden gewonnen, die einen Gutschein von einem Online-Versandhaus über fünf Euro erhielten.

Die Probanden nahmen in individuell vereinbarten Sitzungen an der Studie teil, die alle nach dem in Kap. 8.1.7 beschriebenem Schema abliefen. Wäh-

rend der Sitzungen war außer dem aktuellen Probanden und dem Studienleiter niemand anwesend. Die ersten beiden Sitzungen je Gruppe fungierten als Pilotsitzungen, in denen Feedback über das allgemeine Vorgehen und die Bedienbarkeit der verwendeten Anwendung gesammelt wurde, um potentielle Fallstricke und Probleme zu identifizieren, die mit dem Aufbau der Studie zusammenhingen. Daraus folgten Änderungen an der verwendeten Anwendung, sodass keine gleichen Bedingungen mehr gewährleistet waren. Folglich wurden die beiden Pilotsitzungen nicht in die Ergebnisse eingearbeitet. Somit bleiben 12 Probanden, deren Ergebnisse letztlich in die Auswertung der Studie flossen. Im Folgenden werden die Probanden anhand ihrer Notation sowie einer fortlaufenden Nummer mit GeForMT1-6 bzw. GC1-6 identifiziert.

8.1.3 Eingesetzte Gesten

Die in der Studie eingesetzten acht Gesten hatten einen unterschiedlichen Schwierigkeitsgrad, der sich auch in den als GeForMT-Ausdruck und GestureCards formulierten Beschreibungen widerspiegeln sollte. Es wurden Single- und Multi-Stroke-, Multi-Touch- und rhythmische Gesten verwendet, um eine möglichst große Bandbreite verschiedener Gesten abzudecken. Es ist möglich, dass verschiedene Gestenarten u. U. unterschiedliche Anforderungen an Gestennotationen haben und sich ggf. unterschiedlich gut mit einer bestimmten Notation bzw. einem bestimmten Paradigma beschreiben lassen. Darüber hinaus wurden Gesten mit verschiedenen Eigenschaften eingesetzt, z. B. unterschiedliche Dauer oder definierter Objektbezug, da in den verglichenen Notationen hier wesentliche Unterschiede in der Umsetzung vorliegen. Sowohl die GeForMT-Ausdrücke als auch die GestureCards wurden anonymisiert, um eine eventuelle Beeinflussung der Probanden durch semantische Bedeutung der Namen auszuschließen. In diesem Kontext bedeutet Anonymisierung, dass den Ausdrücken keine sprechenden Namen gegeben wurden. GeForMT-Ausdrücke wurden also nicht als Zuweisung definiert, sondern als alleinstehender Ausdruck. In GestureCards wurden alle Namen mit *Card* gefolgt von einem Buchstaben in alphabetischer Reihenfolge nummeriert. Im Folgenden werden die Gesten im Einzelnen beschrieben und ihre individuellen Herausforderungen und Schwerpunkte herausgestellt.

Geste V

Diese Geste ist eine Single-Stroke-Geste, die in Form eines V ausgeführt werden muss. Beide Schenkel müssen entsprechend hintereinander ohne Absetzen des Fingers von der Oberfläche durchgeführt werden. Die Bewegungsrichtung ist darüber hinaus in beiden Notationen vorgegeben, d. h. eine Geste in der richtigen Form, jedoch in der falschen Bewegungsrichtung (z. B. von rechts nach links) entspräche nicht der Beschreibung. Die Herausforderung für die Probanden besteht somit in der Identifikation der Bewegungsrichtung beider Schenkel und der korrekten Interpretation des Operators im GeForMT-Ausdruck, der die zeitliche Relation beschreibt. Tab. 8.1 zeigt, wie die Geste in GestureCards und GeForMT beschrieben wurde.

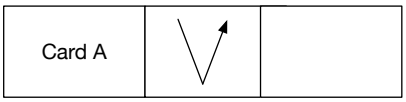
Notation	Darstellung
GeForMT	1F(LINE_SE,LINE_NE)
GestureCards	

Tabelle 8.1: Die Geste V ausgedrückt in GeForMT und GestureCards

Geste X

Diese Multi-Stroke-Geste muss in der Form eines X durchgeführt werden. Die Probanden müssen erkennen, dass sie den Finger von der Oberfläche abheben dürfen und dass die Bewegungsrichtung – analog zur Geste V – vorgegeben ist. Eine weitere Schwierigkeit ist, dass die Schenkel exakt in der gleichen Richtung wie bei der Geste V ausgeführt, jedoch unterschiedlich positioniert werden. Das äußert sich insbesondere im GeForMT-Ausdruck, der große Ähnlichkeit mit dem Ausdruck der Geste V hat. Der Operator `CROSS` bzw. die Eigenschaft `Position` definieren, dass der zweite Schenkel den ersten kreuzen muss. Außerdem muss bei der GestureCard die Referenz auf die andere Teilgeste, die dem ersten Schenkel entspricht, korrekt interpretiert werden. Bei GeForMT wird hingegen der gesamte Ausdruck vom Schlüsselwort `CROSS` umfasst. Tab. 8.2 zeigt sowohl die zugehörige GestureCard als auch den GeForMT-Ausdruck.

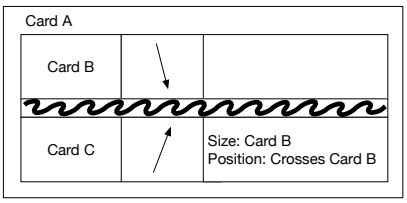
Notation	Darstellung
GeForMT	CROSS[1F(LINE_SE;LINE_NE)]
GestureCards	

Tabelle 8.2: Die Geste X ausgedrückt in GeForMT und GestureCards

Geste Y

Hierbei handelt es sich ebenfalls um eine Single-Stroke-Geste, jedoch in der Form eines Y. Statt zweier Schenkel müssen die Probanden nun eine Kombination unterschiedlicher Linien und Halbkreise korrekt interpretieren. Die Herausforderung für die Probanden besteht somit in der korrekten Identifikation der unterschiedlichen Schlüsselwörter, der Zuordnung der zusätzlichen Eigenschaften wie Bewegungsrichtung der Halbkreise sowie der Kombination der einzelnen Teilgesten. Tab. 8.3 zeigt die in der Studie verwendeten Ausdrücke. Der GeForMT-Ausdruck wurde von Kammer (2014, S. 119) übernommen.

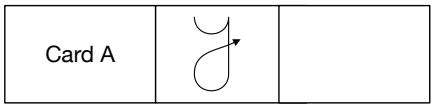
Notation	Darstellung
GeForMT	1F(SEMICIRCLE_E_CCW,LINE_S,LINE_S,SEMICIRCLE_W_CW,LINE_NE,LINE_NE)
GestureCards	

Tabelle 8.3: Die Geste Y ausgedrückt in GeForMT und GestureCards

Geste Pfeil

Diese Geste ist eine Multi-Stroke-Geste in Form eines Pfeiles. Pfeilspitze und -schaft werden als eigenständige Gestenzüge durchgeführt und bilden so die beiden Teilgesten. Die Pfeilspitze besteht aus zwei einzelnen Gestenzügen, die als Single-Stroke-Geste ausgeführt werden. Das führt insbesondere im GeForMT-Ausdruck dazu, dass unterschiedliche Operatoren parallel verwendet werden und die Probanden diese Unterscheidung erkennen und umsetzen müssen. Eine weitere Herausforderung ist die Identifikation der räumlichen Relation, die in GestureCards durch die Eigenschaft `Position` und in GeForMT durch das Schlüsselwort `ASIDE` ausgedrückt wird. Wie diese Geste mit GestureCards und GeForMT beschrieben wird, ist in Tab. 8.4 dargestellt.

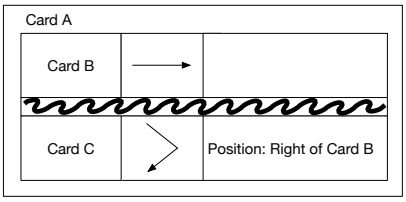
Notation	Darstellung
GeForMT	ASIDE[1F(LINE_E);1F(LINE_SE,LINE_SW)]
GestureCards	

Tabelle 8.4: Die Geste Pfeil ausgedrückt in GeForMT und GestureCards

Geste HoldMove

Dies ist eine Multi-Touch-Geste, bei der die Teilgesten unterschiedliche Formen haben, aber zeitlich parallel ausgeführt werden. Sie haben unterschiedliche Eigenschaften, die sich z. T. aufeinander beziehen, und werden an verschiedenen Orten ausgeführt: auf der Leinwand und einem beliebigen virtuellen Objekt. Außerdem ist für die eine Teilgeste eine bestimmte Dauer definiert; in der zugehörigen GestureCard durch die Eigenschaft `Duration` und im GeForMT-Ausdruck durch das Schlüsselwort `HOLD`. Die Beschreibungen weichen hier in einem Detail voneinander ab: In der GestureCard hat die eine Teilgeste die Dauer der anderen, während im GeForMT-Ausdruck die Dauer unabhängig ist – in GeForMT können keine Referenzen auf Teilgesten definiert werden. Die Ausdrücke beschreiben somit nicht mehr exakt die gleiche[1] Geste. In der GestureCard ist außerdem für die zweite Teilgeste ein Pfeil in der graphischen Repräsentation definiert, der eine Linie nach Rechts beschreibt, jedoch wird die Bewegungsrichtung durch die Eigenschaft `Direction` auf *in Richtung Card B* festlegt. Die graphische Repräsentation

[1] Damit die Ausdrücke äquivalent sind, muss in der GestureCard eigentlich die Eigenschaft *Duration* auf den Wert *Long* gesetzt werden. Der Unterschied ist erst spät in der Studie aufgefallen und wurde nicht mehr korrigiert, zeigte jedoch keine Auswirkungen.

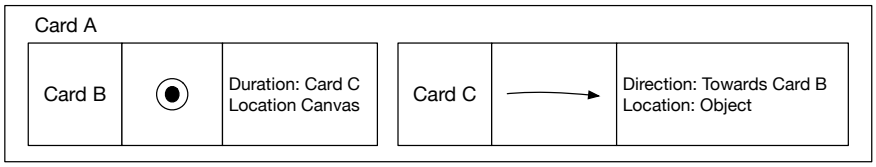
Notation	Darstellung
GeForMT	JOIN[1F(HOLD(Canvas)) * 1F(LINE(Object))]
GestureCards	

Tabelle 8.5: Die Geste HoldMove ausgedrückt in GeForMT und GestureCards

wird so überschrieben und eine dynamische, zur Spezifikationszeit nur durch eine Relation zur anderen Geste bestimmbaren Richtung definiert. Tab. 8.5 zeigt, wie diese Geste in GestureCards und GeForMT formuliert wird.

Geste Spread

Bei dieser Multi-Touch-Geste werden beide Finger gleichzeitig auseinandergezogen. Sie gehört zu den weit verbreiteten Standardgesten, die von vielen Frameworks angeboten werden. Die Herausforderung für die Probanden besteht darin, zu erkennen, dass die Richtung der Ausführung egal ist, solange die grundlegende geometrische Figur beibehalten wird. In GestureCards wird dazu die Eigenschaft `Direction` verwendet, die hier als `Invariant`[2] angegeben ist. In GeForMT hingegen wird diese Eigenschaft durch die fehlende Richtungsanweisung beim Schlüsselwort `LINE` ausgedrückt. Außerdem teilt der GeForMT-Ausdruck die Geste in zwei Teilgesten, die mit einem Operator verbunden werden. Tab. 8.6 zeigt beide Ausdrücke.

[2] Die Studie wurde durchgeführt, bevor `Direction: Invariant` gegen `Orientation: None` ausgetauscht wurde, daher wird für die Beschreibung dieser Studie an der veralteten Variante festgehalten.

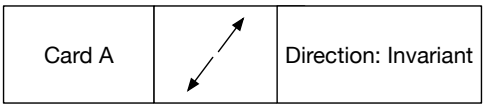
Notation	Darstellung
GeForMT	SPREAD[1F(LINE) * 1F(LINE)]
GestureCards	

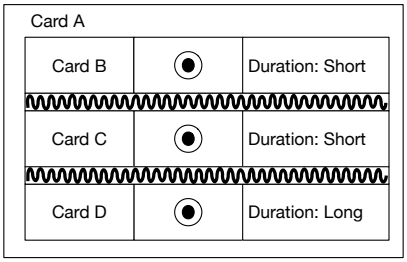
Tabelle 8.6: Die Geste Spread ausgedrückt in GeForMT und GestureCards

Geste Kurz-Kurz-Lang (SSL)

Diese Geste zeichnet neben der Anzahl der Teilgesten die rhythmische Struktur aus, die sich aus der unterschiedlichen Dauer der einzelnen Teilgesten ergibt. In GestureCards und GeForMT[3] unterscheidet sich die Definition in einem wesentlichem Detail: In GeForMT werden dedizierte, einzelne Schlüsselwörter für die kurze und lange Dauer einer Berührung definiert verwendet und in GestureCards dagegen unterschiedliche Werte für die Eigenschaft `Duration`. Tab. 8.7 veranschaulicht die Unterschiede.

[3] Proband GeForMT1 wies in der ersten Sitzung auf einen Fehler hin, der in den folgenden Sitzungen korrigiert wurde: Fälschlicherweise wurde initial der Operator für Single-Stroke-Gesten (,) verwendet, der gegen den dargestellten Operator (;) ausgetauscht werden musste.

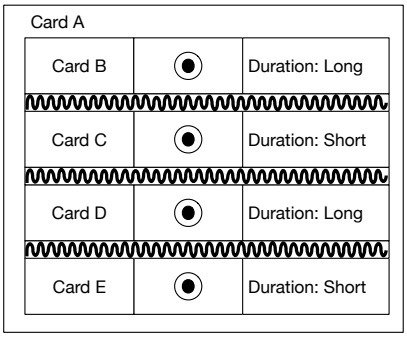
Tabelle 8.7: Die Geste SSL ausgedrückt in GeForMT und GestureCards

Notation	Darstellung
GeForMT	1F(POINT;POINT;HOLD)
GestureCards	

Geste Lang-Kurz-Lang-Kurz (LSLS)

Diese Geste ist ebenfalls eine rhythmische Geste, verwendet aber eine andere Rhythmik als die Geste SSL. Dadurch treten die Unterschiede zwischen GeForMT und GestureCards für diese Art von Geste nochmal verstärkt hervor, die in Tab. 8.8 veranschaulicht werden.

Tabelle 8.8: Die Geste LSLS ausgedrückt in GeForMT und GestureCards

Notation	Darstellung
GeForMT	1F(HOLD;POINT;HOLD;POINT)
GestureCards	

8.1.4 Eingesetzte Anwendung

Die Studie wurde mithilfe einer Anwendung durchgeführt, die die Probanden selbstständig bedienen konnten. Die Anwendung bestand aus zwei Elementen: einer Slideshow, die zur Einführung in eine der beiden verwendeten Notationen genutzt wurde, und einer Oberfläche, auf der die Gestenbeschreibungen angezeigt und Aktionen der Probanden protokolliert wurden. Außerdem wurde ein Beispiel integriert, damit die Probanden die Anwendung nach der Einführung in die Notation und vor Beginn der eigentlichen Studie einmal ausprobieren konnten.

Die Einführung in die Notation bestand für beide Notationen aus fünf Einzelteilen, die die wesentlichen Aspekte zusammenfassten:

- Eine allgemeine Einführung in das Konzept der Notation

- Die grundlegenden Regeln zur Beschreibung einfacher Gesten mit einem Beispiel
- Die grundlegenden Regeln zur Beschreibung zusammengesetzter Gesten
- Die grundlegenden Regeln zur Beschreibung zeitlicher Relationen (vgl. Kap. 5.2.1)
- Ein Beispiel für eine zusammengesetzte Geste

Für beide Notationen wurde versucht, die gleichen Informationen in der Einführung zu vermitteln, um allen Probanden eine möglichst identische Grundlage zu geben. Die dafür verwendeten Materialien sind in Anh. B zu finden. Die Probanden konnten selbstständig das nächste Element der Notationseinführung aufrufen und wurden währenddessen nicht unterbrochen oder durch weitere Informationen seitens des Studienleiters eingewiesen. Darüber hinaus durften sie keine Verständnisfragen stellen, die sich auf die Inhalte der Notation bezogen.

Außerdem stellte die Anwendung eine Oberfläche zur Verfügung, mit der die Probanden arbeiteten. Abb. 8.1 zeigt einen Screenshot des UI, das aus vier Komponenten bestand. Im oberen Teil wurde die aktuelle Geste in der jeweiligen Notation beschrieben. Je nach Gruppe befand sich an dieser Stelle also entweder eine GestureCard – wie in Abb. 8.1 dargestellt – oder ein GeForMT-Ausdruck (vgl. Kap. 8.1.3). Unter dem aktuellen Ausdruck war beispielhaft ein Fenster einer hypothetischen Anwendung abgebildet, das die Probanden zur Orientierung nutzen konnten. Zwar war grundlegend die Durchführung der Geste auf der gesamtem Oberfläche möglich, aber Feedback aus initialen Pilotsitzungen hatte offenbart, dass eine Orientierungshilfe für die Probanden praktisch ist. Durchgeführte Gesten wurden mit sichtbaren Strichen visuell hervorgehoben, sodass Probanden ihre letzten Gestenzügen für jede Geste sehen konnten.

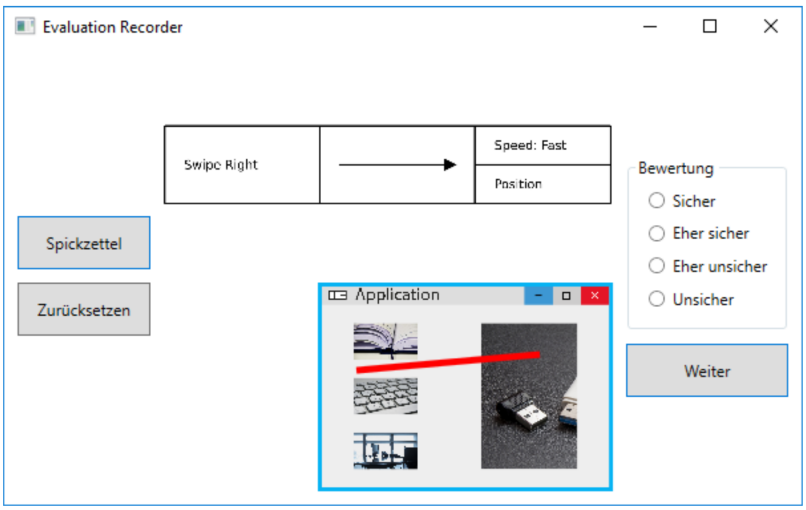


Abbildung 8.1: UI der für die Studie verwendeten Anwendung mit einer visualisierten Geste (rote Linie). Die dargestellte Abbildung war das für GestureCards verwendete Beispiel; die Geste selbst wurde nicht in der Studie eingesetzt.

Auf der linken Seite waren zwei Schaltflächen verfügbar. Mit der unteren Schaltfläche – *Zurücksetzen* – konnte die zuletzt ausgeführte Geste rückgängig gemacht werden, d. h. alle sichtbaren Gestenzüge wurden gelöscht. Mit der oberen Schaltfläche – passend *Spickzettel* benannt – konnte eine Dokumentation geöffnet werden, die weitergehende Informationen über die jeweilige Notation enthielt. Der Spickzettel bestand aus zwei Seiten und fasste im Wesentlichen die Informationen aus der Einführung nochmal zusammen, ergänzte sie aber um notationsspezifische Details. Die detaillierten Spickzettel sind in Anh. B zu finden. Der Spickzettel konnte zu jeder Zeit und beliebig oft aufgerufen, jedoch nie parallel zur Bearbeitung der aktuellen Geste angezeigt werden, d. h. er überlagerte immer die Darstellung der Geste. Die Probanden konnten allerdings selbstständig im Spickzettel hin und her blättern und sich jede Seite beliebig lange anschauen. Eine zeitliche Beschränkung war nicht vorgesehen.

Auf der rechten Seite befand sich eine Likert-Skala, mit der Probanden einschätzen mussten, wie zuversichtlich sie waren, die Beschreibung korrekt interpretiert und die Geste erfolgreich identifiziert zu haben. Diese umfasste vier Werte in von oben nach unten absteigender Reihenfolge: *Sicher*, *Eher Sicher*, *Eher Unsicher* und *Unsicher*. Die Bewertung der Zuversicht war Voraussetzung, um zur nächsten Geste wechseln zu können, und beendete die Bearbeitung der aktuellen Geste. Ein erneuter Aufruf oder eine Korrektur war danach nicht mehr möglich.

Die Anwendung war allerdings nicht in der Lage, die durchgeführte Geste zu erkennen, und konnte den Probanden somit keinerlei Feedback über ihre Ergebnisse geben. Die (technische) Erkennung der Gesten war für die Studie irrelevant und hätte über den Verlauf einer Sitzung nur schwer einschätzbare Erfolgs- oder Misserfolgserlebnisse bei den Probanden ausgelöst. Daher wurde auf Mechanismen zur Gestenerkennung verzichtet.

Die durchgeführten Gesten wurden – wie auch einige andere Daten – jedoch von der Anwendung protokolliert. Diese Protokolle bildeten die Grundlage für die Analyse und werden durch Informationen aus der Videoaufnahme ergänzt. Die Daten wurden je Sitzung separat gespeichert. Für alle Probanden wurden folgende Daten aufgezeichnet:

- Jede Berührung des Bildschirms, wodurch sich im Nachhinein alle durchgeführten Gesten rekonstruieren und vergleichen lassen
- Die Anzahl der durchgeführten Korrekturen durch Betätigung der Schaltfläche *Zurücksetzen*
- Jeder Aufruf des Spickzettels über die entsprechende Schaltfläche (die Navigation innerhalb des Spickzettels wurde nicht berücksichtigt)
- Die im Spickzettel verweilte Zeit (vom Öffnen bis zum Schließen, gemessen am Zeitpunkt der Betätigung der jeweiligen Schaltflächen)

- Die Zeit zur Betrachtung des Ausdrucks (die Zeit, in der der Ausdruck angezeigt und der Proband nicht mit dem Bildschirm interagierte oder den Spickzettel betrachtete)
- Die Zeit zur Durchführung der Geste (die Zeit, in der der Bildschirm berührt wurde)
- Das Maß an Zuversicht, das die Probanden nach Bearbeitung des jeweiligen Ausdrucks auf der Likert-Skala festlegten

8.1.5 Hypothesen und Variablen

Mit Blick auf die zu Beginn dieses Kapitels formulierte Forschungsfrage und die in Kap. 8.1.1 definierten Ziele sollen im Folgenden die der statistischen Auswertung zugrundeliegenden Hypothesen und die notwendigen Variablen definiert werden. Dazu werden die als Grundlage des Verständnisses definierten Daten verwendet, die im Rahmen der einzelnen Sitzungen erfasst werden und die die Arbeit der Probanden widerspiegeln. Somit ergeben sich für jede Geste, die die Probanden bearbeiten, vier Variablen, die zur späteren Auswertung herangezogen werden. Tab. 8.9 gibt einen Überblick über die abhängigen Variablen der Studie und definiert ihre relevanten Eigenschaften.

Die Variablen wurden auf unterschiedliche Art und Weise erfasst. Beide Variablen *Spickzettel* und *Gesamtzeit* können anhand der Aktionen der Probanden während der einzelnen Sitzungen gemessen werden. Die verwendete Anwendung erfasst dafür die notwendigen Daten. Die Variable *Zuversicht* wird ebenfalls durch die Anwendung erfasst, aber von den Probanden selbst eingeschätzt. Die Variable *Korrekt* wurde im Nachhinein von der Studienleitung festgelegt; hierzu wurden die Aufzeichnungen der einzelnen Sitzungen analysiert. In diesem Zusammenhang ist eine Geste korrekt von einem Probanden erkannt worden, wenn er die dargestellte Geste im Sinne der Beschreibung ausgeführt hat. Partielle Lösungen sind nicht möglich, d. h. alle Teilgesten und ihre Bezüge müssen korrekt identifiziert werden, damit die Gesamtgeste als korrekt gelten kann. Die Details des Vorgehens sind in Kap. 8.1.7 beschrieben.

Aus diesen vier Variablen leiten sich die Null- und Alternativhypothesen ab. Da für die zugrundeliegende Forschungsfrage überprüft werden soll, ob hybride Notationen besser zu verstehen sind als textuelle, sind auch die Alternativhypothesen für jede Variable entsprechend zu formulieren. Die

Tabelle 8.9: Überblick über alle abhängigen Variablen

Variable	Beschreibung	Einheit	Skala	Mögliche Werte
Korrekt	Die Geste wurde korrekt identifiziert	boolean	nominal	ja, nein
Spickzettel	Die Anzahl der Spickzettelaufrufe	numerisch	metrisch	0..∞
Zuversicht	Die Zuversicht der Probanden, die Geste korrekt identifiziert zu haben	Likert-Skala	ordinal	sicher, eher sicher, eher unsicher, unsicher
Gesamtzeit	Die Zeit, die die Probanden für jede Geste benötigen	Sekunden	metrisch	0..∞

Variable	Nullhypothese H_0	Alternativhypothese H_1
Korrekt	Probanden der Gruppe GestureCards identifizieren genauso viele Gesten korrekt wie Probanden der Gruppe GeForMT	Probanden der Gruppe GestureCards identifizieren mehr Gesten korrekt als Probanden der Gruppe GeForMT
Spickzettel	Probanden der Gruppe GestureCards verwenden den Spickzettel genauso oft wie Probanden der Gruppe GeForMT	Probanden der Gruppe GestureCards verwenden den Spickzettel seltener als Probanden der Gruppe GeForMT
Zuversicht	Probanden der Gruppe GestureCards sind genauso zuversichtlich, Gesten korrekt identifiziert zu haben, wie Probanden der Gruppe GeForMT	Probanden der Gruppe GestureCards sind zuversichtlicher, Gesten korrekt identifiziert zu haben, als Probanden der Gruppe GeForMT
Gesamtzeit	Probanden der Gruppe GestureCards benötigen genauso lange, um Gesten abzuleiten, wie Probanden der Gruppe GeForMT	Probanden der Gruppe GestureCards leiten Gesten schneller ab als Probanden der Gruppe GeForMT

Tabelle 8.10: Überblick über alle Hypothesen der ersten Studie

Nullhypothese geht jeweils von keinem Einfluss der Notation aus, d. h. die Probanden haben ungeachtet der Notation, mit der sie gearbeitet haben, ähnliche Ergebnisse. Tab. 8.10 fasst die Hypothesen für jede Variable zusammen. Im Folgenden werden den einzelnen Hypothesen die zugehörigen Variablen zur Identifikation hinzugesetzt; $H_{1Korrekt}$ bezeichnet also die Alternativhypothese der Variable $Korrekt$ und $H_{0Spickzettel}$ die Nullhypothese der Variable $Spickzettel$.

8.1.6 Studiendesign

Im Rahmen der Studie wurden Probanden gebeten, acht Gesten zu identifizieren, die in den korrespondierenden, in Kap. 8.1.3 beschriebenen Ausdrücken entweder in GeForMT oder in GestureCards notiert wurden. Jeder Proband arbeitete nur mit genau einer Notation. Da die Termine für die einzelnen Sitzungen individuell und ohne präferierte Reihenfolge vereinbart wurden, kann von einer zufälligen Zuweisung ausgegangen werden. Daraus ergeben sich zwei Gruppen, die sich nach der eingesetzten Notation unterscheiden. Somit ergibt sich zwangsläufig, dass die Notation ein Zwischensubjektfaktor bzw. eine unabhängige Variable der Studie ist. Die Hypothesen bilden genau diesen Faktor ab bzw. zielen auf die Unterschiede ab, die dieser Faktor auslöst.

Zur Gewährleistung der Vergleichbarkeit wurden für jede Notation die gleichen Gesten verwendet. Die Reihenfolge, in der die Gesten präsentiert wurden, war zufällig, um eine ungewollte Beeinflussung – sowohl im negativen als auch positiven Sinne – durch Festlegung einer vorgegebenen Sequenz von Gesten auszuschließen. Die Geste muss also als Innersubjektfaktor betrachtet werden. Bei der Studie handelt es sich somit um eine zweifaktorielle Studie mit den beiden unabhängigen Variablen Notation und Geste.

Die Studie wurde mithilfe einer eigens entwickelten Anwendung durchgeführt (vgl. Kap. 8.1.4), die von den Probanden selbst bedient werden konnte und auf einem großen, multi-touch-fähigem Bildschirm (digitales Whiteboard) lief. Auf die Verwendung eines Smartphones oder Tablets wurde bewusst

verzichtet, um die Probanden nicht durch einen gewohnten Formfaktor in der Erkennung und Durchführung der Gesten zu beeinflussen und mehr Freiheitsgrade zu ermöglichen. Smartphones und Tablets sind durch die Haltung in der Hand entsprechend eingeschränkt und ein digitales Whiteboard bietet durch die Möglichkeit, die Hände frei zu bewegen, mehr Optionen zur Interaktion.

8.1.7 Aufbau der einzelnen Sitzungen

Jeder Proband nahm an der Studie in einer eigenen Sitzung teil, deren Zeitpunkt individuell vereinbart wurde. Eine Sitzung fand ohne Publikum statt und dauerte etwa 15 bis 25 Minuten, hatte aber kein vorher festgelegtes Zeitlimit. Der Ablauf war immer identisch und in folgende vier Phasen gegliedert:

- Einführung in den Forschungskontext durch den Studienleiter
- Allgemeine Einführung in die Notation innerhalb der Anwendung; Steuerung durch den Probanden
- Erläuterung des UI durch den Studienleiter und Möglichkeit für den Probanden, das UI auszuprobieren und ein Gefühl für die Bedienung zu bekommen
- Start der eigentlichen Studie durch den Probanden; die Steuerung erfolgt ebenso durch den Probanden

Zunächst wurde der jeweilige Proband in die Bedeutung von Gestensteuerung für moderne HCI im Allgemeinen eingeführt und anschließend die Bedeutung von Spezifikation für die Entwicklung gestenbasierter Anwendungen im Speziellen erläutert. Danach wurde der Forschungskontext knapp beschrieben, aber ohne auf Details der unterschiedlichen Notationen einzugehen. Die Probanden erfuhren lediglich, dass es verschiedene Notationen gibt und dass sie im Folgenden eine Notation kennenlernen würden. Anschließend wurde generell das Konzept der Studie erläutert und die einzelnen Schritte kurz umrissen. An dieser Stelle hatte der Proband noch Gelegenheit, Fragen zu stellen, wobei generell darauf hingewiesen wurde, dass Fragen während der Sitzung zwar möglich seien, diese sich aber zum Vorgehen sowie der verwendeten Anwendung beschränken sollen. Inhaltliche Fragen zu den einzelnen Notationen wurden nicht beantwortet – die Probanden mussten mit den zur Verfügung gestellten Informationen auskommen. Dadurch sollte sichergestellt werden, dass das Ergebnis des jeweiligen Probanden nicht durch individuelle Fragen und Antworten beeinflusst wird.

Die Probanden wurden außerdem gebeten, ihre Gedanken im Sinne des *Think Aloud* laut zu formulieren, um bei der nachfolgenden Analyse weitere Erkenntnisse über den Prozess, den sie bei der Identifikation der Geste durchlaufen, zu gewinnen und eventuelle Stolpersteine der beiden verwendeten Notation zu identifizieren. Jede Sitzung wurde daher – nach voriger Freigabe durch den Probanden – mit Bild und Ton aufgezeichnet.

Anschließend konnten die Probanden die Anwendung verwenden und die Slideshow zur Einführung in die Notation starten. Während dieser Phase gab

es keine geplante Interaktion mit dem Studienleiter. Die Probanden konnten selbstständig das nächste Element der Notationseinführung aufrufen und wurden währenddessen nicht unterbrochen.

Im nächsten Teil wurde das UI der Anwendung anhand eines Beispiels demonstriert. Das Beispiel-UI verhielt sich genauso wie das UI während der eigentlichen Studie, allerdings wurden keine Daten erfasst. Die Probanden hatten auch Gelegenheit, die Anwendung selber auszuprobieren und ein Gefühl für die Bedienung der unterschiedlichen Elemente zu bekommen. Einige Probanden wollten diese Phase nutzen, um den zur Verfügung gestellten Spickzettel intensiv zu betrachten, was aber durch den Studienleiter mit dem freundlichen Hinweis unterbunden wurde, dass dieser ja zu jeder Zeit zur Verfügung steht. Dies war notwendig, da die Verwendung des Spickzettels essentieller Bestandteil der erfassten Daten war und die Probanden nicht mit zusätzlichem Vorwissen in die eigentliche Studie starten sollten.

Anschließend konnten die Probanden selbstständig die eigentliche Studie starten. Eine Interaktion mit dem Studienleiter war ab diesem Punkt nicht mehr vorgesehen. Den Probanden wurden keine zeitlichen Vorgaben gemacht und sie konnten die Geschwindigkeit der Gesamtdurchführung selber steuern, indem sie selbst eine Geste abschlossen und die nächste aufriefen.

8.1.8 Analyse

Die Analyse der Daten umfasste zwei unterschiedliche Elemente: Zum einen wurden die Daten, die die in den Sitzungen verwendete Anwendung aufgezeichnet hat, mit einer weiteren, zu diesem Zweck entwickelten Anwendung aufbereitet, um für die Auswertung genutzt werden zu können; zum anderen wurden die Aufzeichnungen der einzelnen Sitzungen mehrfach betrachtet, um interessante Kommentare, bemerkenswerte Aktionen oder besonderes Verhalten zu identifizieren. Im Folgenden wird erläutert, wie die einzelnen Variablen aus den erfassten Daten extrahiert wurden.

Die Variable *Korrekt* wurde bei der Analyse der Videoaufzeichnungen ermittelt, da hier die durchgeführte Geste deutlich sichtbar war. Eine Alternative wäre gewesen, die Gesten aus den Berührungsdaten zu extrahieren, die die Anwendung protokolliert hat, jedoch gingen dabei verschiedene für einzelne Gesten wesentliche Informationen verloren. So musste z. B. eine Teilgeste der Geste HoldMove auf einem virtuellen Objekt ausgeführt werden, das zwar sichtbar, aber der Anwendung als solches nicht bekannt war. Dieses Vorgehen hatte darüber hinaus den Vorteil, dass die Erkennung, ob die Geste korrekt oder nicht korrekt ausgeführt wurde, nicht durch technische Unzulänglichkeiten des Gestenerkenners beeinflusst wurde.

Zur Erfassung der Variable *Spickzettel* wurde gezählt, wie oft die Probanden je Geste den Button betätigten, der den Spickzettel öffnete. Dabei war nur das initiale Öffnen relevant; jede Weitere Interaktion im Spickzettel, insbesondere das hin und her blättern zwischen den Seiten, wurde nicht gezählt.

Zur Erfassung der Variable *Zuversicht* wurde die Likert-Skala verwendet, die auf der rechten Seite der Anwendung platziert war. Gespeichert wurde

zunächst die Beschriftung des gewählten Werts; für die Auswertung wurden die möglichen Werte in Zahlen von 1 (sicher) bis 4 (unsicher) übersetzt.

Die Variable *Gesamtzeit* wurde gemessen, indem die Aktionen der Probanden mit Zeitstempeln versehen wurden. Somit war ersichtlich, wann sie welche Geste starteten und beendeten. Darüber hinaus wurden detaillierte zeitliche Informationen erfasst: wie lange Probanden den Spickzettel betrachteten und wie viel Zeit sie mit der Ausführung von Gesten verbrachten.

Jeder Proband bearbeitete die acht in Kap. 8.1.3 definierten Gesten. Somit ergeben sich 96 Datensätze in Summe bzw. jeweils 48 für GeForMT und GestureCards zur quantitativen Auswertung. Verschiedene Rahmenbedingungen wie z. B. das unterschiedliche Skalenniveau oder die zugrundeliegende Fragestellung benötigen jedoch unterschiedliche statistische Tests. Die eingesetzten Methoden werden der einfacheren Übersicht halber in den jeweiligen Abschnitten von Kap. 8.2.1 erläutert. Für die bessere Darstellung wird im Folgenden an verschiedenen Stellen auf kumulierte Daten zurückgegriffen. Wenn nicht anders beschrieben, so wurde die statistische Auswertung aber ausschließlich auf den Daten der jeweiligen Sitzungen durchgeführt. Es wird das gängige Signifikanzniveau von 5 % für signifikante bzw. 1 % für sehr signifikante Ergebnisse verwendet (Bortz, 2004, S. 114).

Darüber hinaus wird die Effektstärke in einem für den jeweiligen statistischen Test notwendigen Maß beschrieben und auf die von Cohen (1988, S. 25f) vorgeschlagene Einteilung in kleine, mittlere und große Effektstärken zurückgegriffen. Demnach sind kleine Effektstärken vorwiegend in eher neuen Forschungsbereichen zu finden, in denen die experimentellen Rahmenbedingungen des Forschungsgegenstand noch nicht so gut bekannt sind und daher nicht gut kontrolliert werden können. Eine andere Möglichkeit für kleine Effektstärken ist, dass der Gegenstand der Untersuchung nicht im Labor nachgestellt werden kann, wodurch weitere, unkontrollierbare Rahmenbedingungen hinzukommen. Die Konsequenz ist, dass ein Effekt zwar vorhanden ist, dieser aber nur durch intensive Studien nachweisbar ist, da der Einfluss der Rahmenbedingungen unklar ist. Eine mittlere Effektstärke hingegen ist bei genauer Betrachtung deutlich sichtbar und eine große Effektstärke offenbart wesentliche Unterschiede zwischen den betrachteten Gruppen.

8.2 Ergebnisse

Im Folgenden sollen die Beobachtungen während der Durchführung und die Ergebnisse in zwei Schritten analysiert werden. Zunächst werden die durch die in Kap. 8.1.4 beschriebene Anwendung erfassten Daten statistisch ausgewertet. Anschließend werden die Gedanken und Kommentare, die die Probanden während der Durchführung formulierten, sowie einige Beobachtungen, die in einzelnen Sitzungen gemacht wurden, in einer qualitativen Analyse betrachtet.

Proband	Aufrufe Spickzettel	Median der Zuversicht	Korrekt erkannte Gesten	Durchschnittliche Gesamtzeit je Geste (s)	Durchschnittliche Betrachtungszeit je Geste (s)
GeForMT1	18	1	3	101	62
GeForMT2	8	2	6	71	41
GeForMT3	14	2	5	87	54
GeForMT4	27	2,5	5	117	70
GeForMT5	20	1,5	5	99	67
GeForMT6	21	2	1	88	41
GC1	0	2	8	26	24
GC2	0	1,5	8	36	33
GC3	2	1	6	44	35
GC4	1	1	6	30	24
GC5	2	2	5	36	27
GC6	0	1,5	8	30	28

Tabelle 8.11: Ergebnisse der Studie kumuliert nach Probanden. Die Werte für Zuversicht liegen zwischen 1 (sicher) und 4 (unsicher).

8.2.1 Quantitative Analyse: Statistische Auswertung der Ergebnisse

Im Folgenden wird zunächst ein erster Überblick über die Ergebnisse der einzelnen Probanden gegeben. Anschließend werden die Ergebnisse der statistischen Analyse betrachtet und die unterschiedlichen Erkenntnisse dargelegt. Hier wird dann konkret auf die in Kap. 8.1.5 definierten Faktoren und Variablen eingegangen. Die zugrundeliegenden Daten sind in Anh. B aufgeführt.

8.2.1.1 Überblick über alle Probanden

Tab. 8.11 zeigt einen Überblick der Ergebnisse kumuliert nach Probanden, d. h. alle acht Gesten, die ein Proband in seiner Sitzung bearbeitet hat, werden zusammengefasst. Es treten einige Unterschiede zutage, die im Folgenden einzeln betrachtet werden.

Probanden der Gruppe GeForMT haben weniger Gesten korrekt erkannt als Probanden der Gruppe GestureCards (vgl. Spalte *Korrekt erkannte Gesten* in Tab. 8.11). Alle Probanden der Gruppe GestureCards erkannten mindestens die Hälfte aller Gesten richtig, während zwei Probanden der Gruppe GeForMT weniger als die Hälfte korrekt erkannten. Proband GeForMT6 erkannte sogar nur eine einzige Geste korrekt. Proband GC5 hingegen hat die wenigsten Gesten in der Gruppe GestureCards korrekt identifiziert, aber immer noch mehr oder genauso viele wie fast alle Probanden der Gruppe GeForMT, mit Ausnahme von GeForMT2.

Die Spalte *Median der Zuversicht* zeigt, dass die Probanden der Gruppe GestureCards zumeist zuversichtlicher waren als die Probanden der Gruppe GeForMT, die Gesten korrekt identifiziert zu haben. Selbst die Probanden GC1 und GC5, die von allen Probanden der Gruppe GestureCards die geringste Zuversicht aufwiesen, sind im Median noch so zuversichtlich wie die meisten Probanden der Gruppe GeForMT. Interessant ist auch Proband GeForMT1,

der zwar sehr zuversichtlich war, die Gesten korrekt erkannt zu haben, aber dennoch nichtmal die Hälfte aller Gesten erfolgreich identifzieren konnte.

Darüber hinaus ist hier auch ersichtlich, dass sich die Probanden der Gruppe GeForMT intensiv mit der Notation auseinandersetzen und regelmäßig die Bedeutung der Elemente im Spickzettel nachschauen mussten. Zwar zeigen sich hier durchaus Unterschiede zwischen den einzelnen Probanden – so hat Proband GeForMT3 mehr als dreimal so häufig in den Spickzettel geschaut wie Proband GeForMT2 –, im direkten Vergleich mit den Probanden der Gruppe GestureCards, die den Spickzettel nur sporadisch bis gar nicht einsetzen mussten, zeigt sich aber durchgehend das gleiche Bild. Wie auch in der qualitativen Analyse in Kap. 8.2.2 deutlich wird, beschäftigten sich Probanden der Gruppe GeForMT während der Durchführung sehr stark mit der Bedeutung der unterschiedlichen Schlüsselwörter und Operatoren. Diese Beobachtung wird hier durch die vielen Aufrufe des Spickzettels bestätigt.

Probanden der Gruppe GeForMT brauchten im Schnitt auch länger zur Identifikation einer Geste. Zieht man von der Gesamtzeit die Zeit ab, die Probanden im Spickzettel verbrachten und die sie zur Ausführung von Gesten brauchten, so erhält man die reine Betrachtungszeit – also die Zeit, in der sich Probanden ausschließlich mit dem dargestellten Ausdruck auseinandersetzen konnten. Auch hier benötigten Probanden der Gruppe GeForMT mehr Zeit als Probanden der Gruppe GestureCards.

Alles in allem zeigen die Ergebnisse, dass die Probanden der Gruppe GestureCards mehr Gesten korrekt identifizierten, Elemente der Notation seltener nachschlagen mussten, zuversichtlicher waren und Gesten schneller erkannt haben. Somit scheinen sich die aufgestellten Hypothesen auf den ersten Blick zu bestätigen, was die folgende statistische Analyse nochmal untermauert.

8.2.1.2 Korrektheit identifizierter Gesten

Die Probanden konnten eine Geste nur korrekt oder nicht korrekt identifizieren. Partielle Lösungen, z. B. lediglich die Erkennung einer Teilgeste, waren nicht möglich. Ein Unterschied ist schon in Tab. 8.11 erkennbar: Probanden der Gruppe GestureCards konnten wesentlich mehr Gesten korrekt identifizieren als Probanden der Gruppe GeForMT. Ein t-Test[4] über diese kumulierten Daten zeigt, dass die Unterschiede zwischen den Probanden der Gruppe GeForMT ($M = 4,17$; $SD = 1,835$; 95 % CI $[2,24; 6,09]$) und der Gruppe GestureCards ($M = 6,83$; $SD = 1,329$; 95 % CI $[5,44; 8,23]$) statistisch signifikant sind ($t(10) = -2,88$; $p = 0,016$; $d = 1,66$). Nach Cohen (1988) liegt hier darüber hinaus eine große Effektstärke vor ($d > 0,8$). $H_{0Korrekt}$ ist also abzulehnen; es ist anzunehmen, dass die jeweilige Notation einen Effekt auf das Ergebnis der einzelnen Gruppen hat und dass GestureCards einen positiven Einfluss auf die Identifikation der dargestellten Geste haben.

Tab. 8.12 fasst alle Ergebnisse beider Gruppen in einer Kreuztabelle zusammen. Die Auswertung mit dem χ^2-Test, der zur Auswertung derartiger Kreuztabellen herangezogen werden kann (Janssen & Laatz, 2013, S. 40f), zeigt, dass dieses Ergebnis statistisch signifikant ist ($\chi^2(1, N = 96) = 12,65$;

Korrekt	GeForMT	GestureCards
ja	25	42
nein	23	6
	$p < 0,001$	

Tabelle 8.12: Korrekt und nicht korrekt identifizierte Gesten je Gruppe.

[4] Der t-Test kann an dieser Stelle eingesetzt werden, da die Daten normalverteilt (Shapiro-Wilk-Test, $p > 0,05$) und die Varianzen homogen sind (Levene-Test, $p > 0,05$) (Janssen & Laatz, 2013, S. 323ff).

Geste	GeForMT	GestureCards
SSL	5	6
X	4	5
HoldMove	1	3
Y	1	6
LSLS	5	6
Pfeil	2	4
Spread	3	6
V	4	6
Summe	25	42

Tabelle 8.13: Korrekt identifizierte Gesten je Geste und Gruppe

$p < 0,001$; $\varphi = 0,36$). Auch nach diesem Test ist also $H_{0Korrekt}$ abzulehnen. Darüber hinaus zeigt sich eine mittlere Effektstärke ($0,3 < \varphi < 0,5$), die den Unterschied zwischen beiden Gruppen bestätigt.

Neben der Notation kann aber auch die Auswahl der Gesten einen Einfluss auf das Ergebnis haben. Es kann Gesten geben, die mit einer bestimmten Notation einfacher zu verstehen sind als andere. Somit ist es möglich, dass nicht die Notation als solche, sondern die jeweilige Geste einen entscheidenden Einfluss auf die korrekte bzw. nicht korrekte Identifikation hat. Tab. 8.13 stellt je Geste gegenüber, wie viele Probanden die Geste korrekt identifizieren konnten. Fünf der acht Gesten wurden erfolgreich von allen Probanden der Gruppe GestureCards erkannt; in der Gruppe GeForMT hingegen konnte keine Geste von allen Probanden erkannt werden. Lediglich die rhythmischen Gesten SSL und LSLS konnten von fünf Probanden identifiziert werden. In der Gruppe GeForMT konnten alle anderen Gesten nur noch von vier oder weniger Probanden korrekt identifiziert werden; in der Gruppe GestureCards trifft dies nur auf die Gesten HoldMove und Pfeil zu.

Tabelle 8.14: Kreuztabellen der korrekt und nicht korrekt identifizierten Gesten je Geste und Gruppe

SSL

Korrekt	GeForMT	GestureCards
ja	5	6
nein	1	0
$p = 0,518$		

X

Korrekt	GeForMT	GestureCards
ja	4	5
nein	2	1
$p = 0,667$		

HoldMove

Korrekt	GeForMT	GestureCards
ja	1	3
nein	5	3
$p = 0,303$		

Y

Korrekt	GeForMT	GestureCards
ja	1	6
nein	5	0
$p = 0,006$		

LSLS

Korrekt	GeForMT	GestureCards
ja	5	6
nein	1	0
$p = 0,518$		

Pfeil

Korrekt	GeForMT	GestureCards
ja	2	4
nein	4	2
$p = 0,388$		

Spread

Korrekt	GeForMT	GestureCards
ja	3	6
nein	3	0
$p = 0,068$		

V

Korrekt	GeForMT	GestureCards
ja	4	6
nein	2	0
$p = 0,197$		

Tab. 8.14 zeigt detaillierte Kreuztabellen für jede einzelne Geste sowie die Ergebnisse der statistischen Auswertung. Da einige Werte kleiner als fünf sind, kann der χ^2-Test, der für die kumulierte Kreuztabelle (Tab. 8.12) eingesetzt wurde, hier nicht verwendet werden (Janssen & Laatz, 2013, S. 264). Stattdessen kann auf Fishers oder Barnards exakte Tests zurückgegriffen werden. Fishers Test ist hier nicht zulässig, da alle Summen (sowohl die vertikalen als auch die horizontalen) fix und unabhängig der einzelnen Werte sein müssen, deswegen wird Barnards exakter Test angewendet (Ludbrook, 2008).

Nur die Geste Y zeigt ein statistisch signifikantes Ergebnis. Ein übermäßig starker Einfluss dieser Geste auf das statistische Gesamtergebnis ist möglich

und kann die oben gezogene Schlussfolgerung verfälschen. Entfernt man jedoch das Ergebnis dieser Geste aus den kumulierten Daten in Tab. 8.12, so ist die Auswirkung auf die statistische Signifikanz der Gesamtdaten zwar sichtbar, das Ergebnis aber weiterhin statistisch sehr signifikant ($\chi^2(1, N = 84) = 7,06$; $p = 0,008$; $\varphi = 0,29$). Der Effekt ist weiterhin sichtbar, auch wenn er nun knapp an der Schwelle von kleiner zu mittlerer Effektstärke liegt ($0,1 < \varphi < 0,3$). Somit kann angenommen werden, dass die Notation den entscheidenen Einfluss auf die Ergebnisse der Probanden ausübt und der Einfluss der Geste vernachlässigt werden kann.

Bei der Geste Spread ist das Ergebnis in Tab. 8.14 zwar nicht mehr statistisch signifikant, aber so knapp davor ($p = 0,068$), dass eine Prüfung zur Sicherheit sinnvoll erscheint. Wird auch diese Geste vom kumulierten Gesamtergebnis abgezogen, so ist das Gesamtergebnis weiterhin statistisch signifikant und es ist eine kleine – allerdings weiterhin knapp zur Schwelle zur mittleren – Effektstärke vorhanden ($\chi^2(1, N = 72) = 4,03$; $p = 0,038$; $\varphi = 0,24$). Somit kann die nach Entfernung von Geste Y getroffene Annahme als bestätigt betrachtet werden: Der Einfluss der Gesten auf das Gesamtergebnis ist geringer als der Einfluss der Notation. Gleichzeitig muss die zugrundeliegende Nullhypothese $H_{0Korrekt}$ – dass es keinen Unterschied zwischen den Probanden beider Gruppen gibt – abgelehnt werden.

8.2.1.3 Verwendung des Spickzettels

Die Probanden hatten die Möglichkeit, jederzeit während des Versuchs einen Spickzettel aufzurufen, der weiterführende Informationen zu jeder Notation enthält. Tab. 8.15 zeigt, wie oft die Probanden der jeweiligen Gruppe den Spickzettel bei welcher Geste aufgerufen haben. Probanden der Gruppe GeForMT benutzten den Spickzettel häufiger ($M = 2,25$ je Geste und Proband; 95 % CI $[1,66; 2,84]$) als Probanden der Gruppe GestureCards ($M = 0,1$ je Geste und Proband; 95 % CI $[0,01; 0,19]$), um Schlüsselwörter und die Bedeutung von Operatoren nachzuschlagen. Dies unterlag in der Gruppe GeForMT auch je nach Geste starken Schwankungen: So wurde bei den Gesten SSL und LSLS der Spickzettel seltener benutzt als bei den anderen Gesten; insbesondere die Gesten HoldMove und Y stehen dazu im Kontrast, bei denen der Spickzettel am häufigsten aufgerufen wurde.

Geste	GeForMT	GestureCards
SSL	6	1
X	10	1
HoldMove	23	1
Y	26	0
LSLS	4	1
Pfeil	15	1
Spread	14	0
V	10	0
Summe	108	5

Tabelle 8.15: Aufrufe des Spickzettels je Gruppe und Geste

Da die Daten nicht normalverteilt sind (Shapiro-Wilk-Test; $p < 0,05$), können parametrische Verfahren nicht eingesetzt werden (Janssen & Laatz, 2013, S. 501). Es wird auf den Mann-Whitney-U-Test zurückgegriffen, der zeigt, dass das Ergebnis statistisch sehr signifikant ist ($U = 199$; $p < 0,001$; $r = 0,77$) und die Nullhypothese $H_{0Spickzettel}$ – Probanden der Gruppe GestureCards verwenden den Spickzettel genauso oft wie Probanden der Gruppe GeForMT – abgelehnt werden muss. Darüber hinaus kann eine große Effektstärke beobachtet werden ($r > 0,5$).

Geste	GeForMT	GestureCards
SSL	2	1
X	2	1,5
HoldMove	3,5	3
Y	2,5	1
LSLS	2	2
Pfeil	2	2
Spread	2	1,5
V	2	1

Tabelle 8.16: Median der Zuversicht je Gruppe und Geste; Werte reichen von *sicher* (1) bis *unsicher* (4)

Tabelle 8.17: Zuversicht der Probanden je Notation für alle Gesten

8.2.1.4 Zuversicht

Zur Bestimmung der Zuversicht mussten die Probanden nach der Ableitung jeder Geste auf einer vierstufigen Skala bewerten, wie zuversichtlich sie sind, dass sie die Geste korrekt identifiziert haben. Tab. 8.16 zeigt den Median der Zuversicht aller Probanden je Geste. Probanden der Gruppe GestureCards waren bei sechs Gesten zuversichtlicher, diese korrekt identifiziert zu haben; bei der Hälfte dieser Gesten sogar um mindestens eine Stufe zuversichtlicher (SSL, Y und V). Lediglich bei den Gesten LSLS und Pfeil waren die Probanden beider Gruppen gleich zuversichtlich. In keinem Fall waren Probanden der Gruppe GeForMT zuversichtlicher.

Zuversicht	GeForMT	GestureCards
Sicher	12	25
Eher sicher	22	16
Eher unsicher	11	5
Unsicher	3	2

Zählt man außerdem das Vorkommen der einzelnen Zuversichtslevel, so verstärkt sich dieses Ergebnis. Während die Probanden der Gruppe GestureCards ($Mdn = 1$ (*sicher*)) bei über der Hälfte der Gesten *sicher* und auch für die andere Hälfte überwiegend *eher sicher* waren, verteilt sich die Zuversicht bei der Gruppe GeForMT ($Mdn = 2$ (*eher sicher*)) auf die Level *sicher*, *eher sicher* und *eher unsicher*. Tab. 8.17 zeigt eine Übersicht und Abb. 8.2 stellt die kumulierte Zuversicht beider Gruppen gegenüber.

Da die Daten ordinalskaliert sind, muss wieder auf ein nichtparametrisches Verfahren zurückgegriffen werden. Der Mann-Whitney-U-Test zeigt, dass auch die Ergebnisse dieser Analyse statistisch sehr signifikant sind ($U = 808,5$; $p = 0,007$; $r = 0,27$) und die Alternativhypothese $H_{1Zuversicht}$ angenommen werden muss: Probanden der Gruppe GestureCards sind zuversichtlicher, die Geste korrekt identifiziert zu haben, als Probanden der Gruppe GeForMT. Darüber hinaus ist eine kleine Effektstärke zu sehen, die knapp an der Grenze zur mittleren Effektstärke liegt ($0,1 < r < 0,3$).

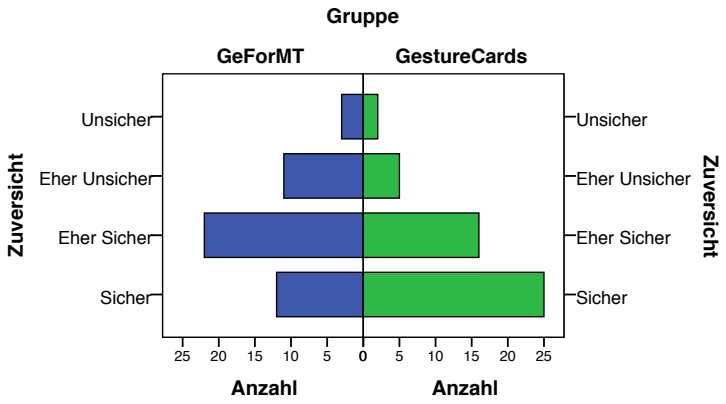


Abbildung 8.2: Vergleich der Zuversicht der Gruppen GeForMT und GestureCards über alle Gesten

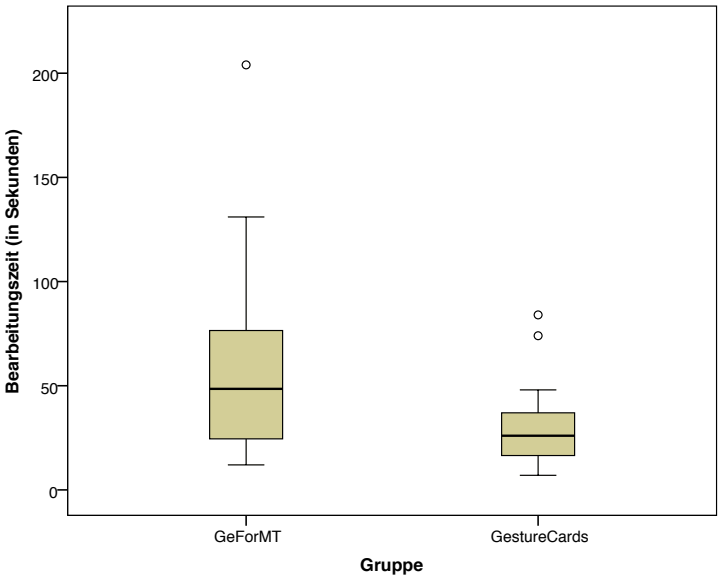


Abbildung 8.3: Vergleich der Betrachtungszeit aller Gesten beider Gruppen

8.2.1.5 Dauer

Abschließend soll die Zeit betrachtet werden, die Probanden zur Ableitung der unterschiedlichen Gesten benötigt haben. Tab. 8.18 zeigt, wie lange die Probanden je Notation im Durchschnitt für jede Geste benötigten. Probanden der Gruppe GeForMT ($M = 93,98\ s$; 95 % CI $[75,39s; 112,56\ s]$) benötigten dabei länger als Probanden der Gruppe GestureCards ($M = 33,77\ s$; 95 % CI $[27,77s; 39,77s]$). Mit Ausnahme der Geste LSLS waren die Probanden der Gruppe GestureCards auch für jede Geste schneller. Bei Daten dieser Art bietet sich auf den ersten Blick eine Varianzanalyse an, jedoch zeigt eine Überprüfung mit dem Shapiro-Wilk-Test, dass die Daten nicht normalverteilt sind ($p < 0,05$). Daher muss auf ein nichtparametrisches Verfahren zurückgegriffen werden. Die Ergebnisse des Mann-Whitney-U-Test sind statistisch sehr signifikant ($U = 388$; $p < 0,001$; $r = 0,57$), was zur Annahme der Alternativhypothese $H_{1Gesamtzeit}$ – Probanden der Gruppe GestureCards sind schneller als Probanden der Gruppe GeForMT – führt. Außerdem kann eine große Effektstärke beobachtet werden ($r > 0,5$).

Geste	GeForMT	GestureCards
SSL	56	31
X	59	36
HoldMove	152	60
Y	186	17
LSLS	38	42
Pfeil	104	41
Spread	81	30
V	75	12

Tabelle 8.18: Durchschnittliche Gesamtzeit je Geste (in Sekunden)

Wie in Kap. 8.2.1.3 gezeigt, verwendeten Probanden der Gruppe GeForMT wesentlich häufiger den Spickzettel als Probanden der Gruppe GestureCards, d. h. sie beschäftigten sich nicht nur mit der Geste als solches, sondern verwendeten auch Zeit auf die Bedeutung der einzelnen Elemente der Notation. Entfernt man die Zeit, in der die Probanden den Spickzettel betrachteten sowie die Zeit, die sie mit der Ausführung von Gesten verbrachten, aus der Gesamtzeit je Geste und berücksichtigt stattdessen nur die in Tab. 8.19 dargestellte reine Betrachtungszeit, so bestätigt sich obiges Ergebnis. Mit Ausnahme der Geste LSLS verbrachten die Probanden der Gruppe GestureCards allesamt weniger Zeit mit der Bearbeitung des dargestellten Ausdrucks als Probanden der Gruppe GeForMT. Probanden der Gruppe GeForMT bearbeiteten jede Geste länger ($M = 55,88\ s$; 95 % CI $[44,88\ s; 66,87\ s]$) als Probanden der Gruppe GestureCards ($M = 28,6\ s$; 95 % CI $[24,01\ s; 33,2s]$). Abb. 8.3 stellt die Betrachtungszeit beider Gruppen gegenüber und verdeutlicht nochmal die Unterschiede. Insbesondere wird sichtbar, dass in beiden Gruppen

Geste	GeForMT	GestureCards
SSL	30	25
X	40	29
HoldMove	83	49
Y	110	15
LSLS	27	36
Pfeil	73	34
Spread	44	28
V	41	11

Tabelle 8.19: Durchschnittliche Betrachtungszeit je Geste (in Sekunden)

nur wenige Ausreißer vorhanden sind. Auch diese Ergebnisse sind statistisch sehr signifikant (Mann-Whitney-U-Test; $U = 586,5$; $p < 0,001$; $r = 0,42$) und es liegt eine mittlere Effektstärke vor ($r > 0,3$).

8.2.2 Qualitative Analyse: Auswertung von Think-Aloud-Protokollen und Beobachtungen

In den verschiedenen Sitzungen finden sich wiederkehrende Muster und Beobachtungen. Ein wesentliches Muster ist, dass Probanden der Gruppe GeForMT zu Beginn jeder Geste recht zügig zunächst den Spickzettel öffneten, um Schlüsselwörter und Operatoren nachzuschlagen. Nach einigen Gesten stellte sich zwar ein gewisser Wiedererkennungswert ein – so kommentiert Proband GeForMT1 seine letzte Geste mit „oh, das Komma wieder; also ohne Unterbrechung" –, trotzdem ist der Spickzettel ein häufig frequentiertes Element in der Gruppe GeForMT. Probanden der Gruppe GestureCards hingegen legen dieses Verhalten nicht an den Tag. Zwar schauen sie auch vereinzelt in den Spickzettel, befassen sich aber bei Weitem nicht so intensiv mit den einzelnen Elementen der Notationen.

Generell war die Bedeutung von Operatoren für die Probanden der Gruppe GeForMT ein zentrales Thema. So kommentiert Proband GeForMT3 seine erste Geste mit „1F ist klar, ein Finger, und das Semikolon war hintereinander ausführen, aber das gucke ich nochmal nach". Probanden vergaßen gelegentlich auch die Bedeutung eines Operators – so kommentiert Proband GeForMT2 seine vorletzte Geste mit „Das * muss ich nochmal nachschauen [...]; ich hab wieder vergessen, wofür es steht". Verwechslung von Operatoren kamen in der Gruppe GeForMT auch häufiger vor und sorgten für falsche Identifikation der Geste oder gelegentliches Zurücksetzen bzw. erneutes Nachschlagen.

Ein weiteres Thema für die Probanden der Gruppe GeForMT war die Bedeutung von Schlüsselwörtern. Zwar verwendet GeForMT eine Menge einfach lesbarer Schlüsselwörter und setzt sie auch gemäß ihrer Bedeutung ein (z. B. `LINE` oder `CIRCLE`), dennoch griffen Probanden selbst bei vermeintlich eindeutigen Begriffen häufig zum Spickzettel – Proband GeForMT4 schaut z. B. die konkrete Bedeutung von `LINE` direkt zu Beginn nach. Demgegenüber stehen z. B. die Probanden GC1 und GC2, die bei der Geste Spread ohne Verwendung des Spickzettels die Bedeutung der Eigenschaft `Direction: Invariant` – die Bewegungsrichtung ist unabhängig von der graphischen Repräsentation – sofort nachvollziehen konnten.

Ein verwandtes wiederkehrendes Problem war die fehlerhafte Interpretation von Schlüsselwörtern, selbst wenn sie im Spickzettel nachgeschaut wurden. So schaut Proband GeForMT3 bei der Geste Pfeil die Bedeutung des Schlüsselworts `ASIDE` im Spickzettel nach und kommentiert anschließend „aah, zwei Teile nebeneinander", führt jedoch dann beide Teilgesten untereinander statt nebeneinander aus. Mehrfach interpretierten Probanden der Gruppe GeForMT darüber hinaus die Richtungsangaben verkehrt.

Außerdem waren Abkürzungen in der Gruppe GeForMT problematisch, auch wenn sich einige Probanden diese schnell aus dem Kontext erklären konn-

ten. So rätselte Proband GeForMT1 zu Beginn seiner Sitzung bei der ersten Geste schon über eine Minute über die Bedeutung von „1F“ und somit sehr viel länger als z. B. Proband GeForMT3, der sich dessen sofort bewusst war. Vor allem die Angabe der Bewegungsrichtung (`CW` oder `CCW`) bei Kreisen und Halbkreisen war immer wieder Gegenstand längeren Nachdenkens bzw. Nachschlagens im Spickzettel. Den meisten Probanden der Gruppe GeForMT war außerdem nicht klar, ob die Richtungsangabe für den Start, das Ende (was korrekt wäre) oder gar die Ausrichtung des Halbkreises ist. So interpretierten die Probanden GeForMT4 und GeForMT5 die Richtungsangabe als die Richtung, in die die Öffnung des Halbkreises zeigt, Proband GeForMT6 hingegen als die Richtung, in die die Rundung des Halbkreises zeigt.

In der Gruppe GestureCards hingegen konnten die meisten Gesten ohne Blick in den Spickzettel korrekt identifiziert werden. Das gilt insbesondere für die Gesten, die ausschließlich die graphische Repräsentation einsetzten (wie z. B. Geste Y), aber auch für Gesten, die mit zusätzlichen Eigenschaften ausgestattet waren. Selbst vermeintlich schwierige Gesten wie die Geste Spread, bei der die in der graphischen Repräsentation dargestellte Richtung durch eine textuelle Eigenschaft überschrieben wurde (`Direction: Invariant`), wurden von den Probanden korrekt identifiziert. Die Probanden GC1, GC2 und GC3 wiesen sogar explizit darauf hin, dass die Richtung durch die genannte Eigenschaft egal ist. Proband GC6 geht an dieser Stelle noch einen Schritt weiter und beschreibt, dass er die Geste anders dargestellt hätte – konkret schlug er eine Aufteilung auf zwei parallele GestureCards vor. Tatsächlich kann diese Geste auf mehrere Arten dargestellt werden und der Vorschlag von GC6 wäre korrekt gewesen.

Zwei Probanden, GC3 und GC6, ordneten zu Beginn ihrer Sitzung allerdings die GestureCards den auf dem UI dargestellten Objekten zu. Beide Sitzungen begannen zufällig mit der Geste SSL, bei der drei Teilgesten zu sehen sind. Daraus leiteten die Probanden wohl einen Zusammenhang her und tippten für jede der drei Teilgesten unterschiedliche Objekte auf dem Bildschirm an. Damit führten sie zwar die Geste korrekt aus,[5] entwickelten aber wohl ein mentales Modell, das sich in den nachfolgenden Gesten als falsch herausstellen sollte. In beiden Fällen realisierten die Probanden ihre falsche Vorstellung innerhalb der nächsten Gesten.

[5] Die Geste SSL beschreibt lediglich eine rhythmische Geste bestehend aus drei Taps, grenzt aber den Ausführungsort nicht weiter ein.

Darüber hinaus fällt auf, dass die Probanden der Gruppe GestureCards sich manchmal dazu verleiten ließen, zusätzliche Eigenschaften zu ignorieren und sich rein auf die graphische Darstellung zu verlassen. Daraus folgten zwangsläufig einige Fehler. So ignorierte z. B. Proband GC5 bei der Geste X die Eigenschaft `Position: Crosses`, wodurch aus der X-Form eine V-Form wird. Dasselbe Phänomen lässt sich aber auch bei Probanden der Gruppe GeForMT beobachten. Hier ignorierte Proband GeForMT6 bei der gleichen Geste ebenfalls das Schlüsselwort `CROSS` und machte somit den gleichen Fehler wie Proband GC5.

Ein wiederkehrendes Problem in beiden Gruppen war die Verwendung von Namen zur Identifikation virtueller Objekte. So fragten sich sowohl Probanden der Gruppe GeForMT als auch der Gruppe GestureCards, was die Bedeutung von *Object* und *Canvas* bei Geste HoldMove sei.

Betrachtet man einzelne Gesten, so stechen die Gesten HoldMove und Pfeil in der Menge der Kommentare heraus. Neben der bereits angesprochenen Bedeutung von *Object* und *Canvas* hatten Probanden beider Gruppen (GeForMT2, GC3, GC5) Schwierigkeiten, die Geste HoldMove als Multi-Touch-Geste zu erkennen und führten die Teilgesten als Multi-Stroke-Gesten aus. Proband GC4 führte die Geste zwar korrekt aus, ignorierte aber die Bedingungen, die durch die Eigenschaft `Location` gegeben war. Bei der Geste Pfeil interpretierten einige Probanden beider Gruppen die jeweiligen Teilgesten und ihre Positionierung nebeneinander zwar korrekt, jedoch verschoben sie die Pfeilspitze nach oben oder unten. In diesem Falle liegen aber keine Fehler der Probanden vor, da beide Notationen lediglich die Positionierung auf der horizontalen genau definieren.

8.3 Diskussion

Ziel der Studie war es, zu vergleichen, ob hybride Notationen besser zu verstehen sind als textuelle Notationen. Zu diesem Zweck wurden GestureCards und GeForMT gegenübergestellt. Die qualitative Auswertung der Kommentare und Gedanken der Probanden zeigten, dass in der Gruppe GeForMT vor allem die Bedeutung von Schlüsselwörtern und Operatoren im Vordergrund stand. Dadurch mussten sich die Probanden zwangsläufig mit dem gesamten Wortschatz der Notation auseinandersetzen. Bei der Gruppe GestureCards hingegen war das nicht in derselben Intensität nötig.

Die statistische Auswertung der insbesondere von der verwendeten Anwendung erfassten Daten zeigt, dass die in Kap. 8.1.5 aufgestellten Alternativhypothesen angenommen werden müssen. Daraus ist zu schließen, dass die eindeutig besseren Ergebnisse der Gruppe GestureCards kein Zufall sind, sondern dass GestureCards tatsächlich dafür sorgen, dass mehr Gesten korrekt identifiziert (Gruppe GestureCards $M = 6,83$; Gruppe GeForMT $M = 4,17$) werden. Gleichzeitig griffen Probanden der Gruppe GestureCards seltener auf die in Form eines Spickzettels zur Verfügung gestellte Dokumentation der Notation zurück (Gruppe GestureCards $M = 0,1$; Gruppe GeForMT $M = 2,25$) und waren zudem etwas zuversichtlicher, die Gesten korrekt erkannt zu haben (Gruppe GestureCards $Mdn = 1$ (*sicher*); Gruppe GeForMT $Mdn = 2$ (*eher sicher*)). Darüber hinaus benötigten die Probanden der Gruppe GestureCards auch weniger Zeit, um alle Gesten zu bearbeiten, selbst wenn die im Spickzettel verbrachte Zeit abgezogen wird (Gruppe GestureCards $M = 28,6\ s$; Gruppe GeForMT $M = 55,88s$). Die Ergebnisse zeigen, dass die übergeordnete Forschungsfrage, die einleitend zur Studie gestellt wurde, für die beiden verglichenen Notationen bejaht werden kann: Die hybride Notation wurde von ungeübten Probanden schneller und besser verstanden als die textuelle Notation.

Insbesondere die Übersetzungen von Formen in Schlüsselwörter und Operatoren scheint den Probanden wesentliche Schwierigkeiten bereitet zu haben. Dafür ist ein gewisses Abstraktionsvermögen notwendig, das aber bei einem Großteil der Probanden gegeben sein müsste: Abstraktion ist eine der grundlegenden Eigenschaften, die in der Ausbildung im SE vermittelt wird. Dass

die Probanden dennoch Schwierigkeiten damit hatten, lässt den Schluss zu, dass es bei cross-funktionalen Teams deswegen zu größeren Schwierigkeiten kommen könnte und textuelle Notationen daher ungeeignet zur Kommunikation von Gesten sind. Während die rhythmischen Gesten SSL und LSLS, die keine Form haben und recht einfach strukturiert sind, da sie nur aus einer Sequenz von Berührungen bestehen, verhältnismäßig problemlos erkannt werden konnten, machten insbesondere die Gesten, deren Formen aus mehreren Elementen bestehen, Schwierigkeiten. Bei beiden rhythmischen Gesten wurde auch der Spickzettel von den Probanden der Gruppe GeForMT nicht so häufig aufgerufen wie bei den anderen Gesten (vier Mal bei der Geste LSLS, sechs Mal bei der Geste SSL und zehn Mal oder häufiger bei allen anderen Gesten) und die Probanden waren am schnellsten fertig, bei der Geste LSLS im Schnitt sogar schneller als die Probanden der Gruppe GestureCards. Bei allen anderen Gesten hingegen sind die Ergebnisse eindeutig andersherum: Probanden der Gruppe GestureCards waren schneller fertig und riefen den Spickzettel nur selten auf.

Auch scheint die Darstellung zeitlicher Beziehungen, die in GestureCards durch die räumliche Anordnung der Teilgesten definiert wird, kein wesentliches Problem zu sein. Die Probanden der Gruppe GestureCards verstanden weitgehend, dass die Anordnung eine Bedeutung hat und konnten sich diese auch merken, nachdem sie in der Einführung erläutert wurde, auch wenn hier durchaus Ausnahmen zu beobachten sind. Im Gegensatz dazu mussten die Probanden der Gruppe GeForMT die Bedeutung der einzelnen Operatoren regelmäßig nachschlagen.

Allerdings ist auffällig, dass bei einigen Probanden der Gruppe GestureCards die Bedeutung der graphischen Repräsentation die Interpretation der GestureCard dominierte. Ab und an wurden zusätzliche Eigenschaften ignoriert, die für die Geste wesentlich waren: So stellte z. B. die Eigenschaft `Direction: Invariant` bei der Geste Spread die Probanden nicht vor Schwierigkeiten und sie interpretierten korrekt ihre Bedeutung, jedoch wurde die Eigenschaft `Location` in der Geste HoldMove nicht immer beachtet. Dasselbe Phänomen – dass ab und an Schlüsselwörter ignoriert wurden – findet sich aber auch bei einigen Probanden der Gruppe GeForMT.

Die Ursachen für den Unterschied zwischen GeForMT und GestureCards können vielfältig sein. Sicher spielen kognitive Prozesse eine Rolle, da verschiedene Informationen auf graphischem Wege vielleicht einfacher übermittelt werden können. Eine detaillierte Analyse der unterschiedlichen Faktoren, die die Wahrnehmung von Notationen beeinflussen, übersteigt jedoch den Fokus dieser Arbeit. Eine Betrachtung aus Sicht der Kognitionswissenschaften oder der Semiotik kann hier vielleicht weiterführende Erkenntnisse bringen.

GestureCards können aber vielleicht aufgrund ihrer Struktur einfacher zu erlernen sein als textuelle Notationen, die mehr Schlüsselwörter und Operatoren verlangen. Dass gerade die Bedeutung von Operatoren schwierig für die Probanden war, zeigen vor allem die Ergebnisse der qualitativen Auswertung. Außerdem müssen sich Nutzer textueller Notationen mit der Syntax und zusätzlichen, strukturierenden Elementen – z. B. unterschiedliche Klammern – auseinandersetzen und ihre nicht immer eindeutige Bedeutung verstehen. So-

mit müssen bei textuellen Notationen auch zusätzliche Regeln für die Syntax der Ausdrücke definiert werden.

Gleichzeitig werfen die Ergebnisse die Frage auf, warum die meisten Notationen zur Beschreibung von Gesten vorwiegend auf rein textuelle Elemente zurückgreifen. Insbesondere, weil es zahlreiche Algorithmen gibt, deren Grundlage keine Beschreibung, sondern aufgezeichnete Gesten – also im Wesentlichen eine graphische Darstellung – sind. Die Gründe können auch hier vielfältig sein. Software-Entwickler arbeiten in erster Linie mit Text und eine Notation wie GeForMT hat z. B. den Vorteil, dass Gesten direkt im Quellcode – also dem wesentlichen Artefakt – definiert werden können (Kammer et al., 2015). Sie benutzt auch im Kern die gleichen Strukturen, die Software-Entwickler aus ihrer täglichen Arbeit gewohnt sind.

8.4 Risiken für die Validität

Im Folgenden werden mögliche Limitationen der Studie und Bedrohungen für die Gültigkeit der Ergebnisse diskutiert. Es werden der Aufbau und die Durchführung der Studie analysiert und potentielle Bedrohungen für die interne, externe und allgemeine Gültigkeit (Kitchenham et al., 2002) aufgelistet.

8.4.1 Allgemeine Gültigkeit

Wie in vielen Studien, die an Universitäten durchgeführt werden, nahmen auch Studierende teil, wodurch unter Umständen ein Übertrag der Ergebnisse in ein tatsächliches Projekt nicht möglich ist, da der Einfluss des universitären Umfelds nicht klar ist. Dieses Risiko kann im vorliegenden Versuch als minimal betrachtet werden: Gemischte Teams mit unterschiedlichem Wissensstand und Hintergrund sind Alltag in der Software-Entwicklung. Darüber hinaus zeigen Studien, dass der Unterschied zwischen Probanden aus einem professionellem Umfeld und Studierenden eher gering ist (Höst et al., 2000). Die Probanden waren allerdings vorwiegend männlich; ein eventueller Einfluss dieses Umstands kann nicht ausgeschlossen werden.

Eine weitere Gefahr ist die Auswahl der Gesten. Auch hier kann die Auswirkung als minimal angesehen werden, da unterschiedliche Arten von Gesten im Versuch eingesetzt wurden (Single-Stroke-, Multi-Stroke-, Multi-Touch- und rhythmische Gesten) und diese auch unterschiedliche Ergebnisse brachten, insbesondere im direkten Vergleich beider Notationen.

Probanden können außerdem persönliche Vorlieben haben, im gegebenen Fall z. B. für textuelle oder hybride Notationen. Diese wurden in der Studie nicht berücksichtigt, sodass ein eventueller Einfluss unklar bleibt.

8.4.2 Interne Gültigkeit

Die Studie wurde mithilfe einer speziellen Anwendung durchgeführt, die auf einem digitalen Whiteboard lief. Zwar sind multi-touch-fähige Hardware

und Gestensteuerung durch die Verbreitung von Smartphones und Tablets alltäglich, dennoch bleibt der Einfluss der verwendeten Hardware und der Anwendung unklar. Die Interaktion mit Smartphones, Tablets und digitalen Whiteboards baut zwar auf demselben Paradigma auf, unterscheidet sich aber durch Haptik und Interaktionsmöglichkeiten. Es ist unklar, ob diese vielleicht ungewohnte Hardware Einfluss auf das Ergebnis und insbesondere die Vorstellung der Probanden hatte, wie Gesten auszuführen sind. Darüber hinaus ergeben sich verschiedene Darstellungsoptionen für die verwendete Anwendung. Dieses Risiko sollte minimiert werden, indem Probanden vor Beginn des eigentlichen Experiments bereits die Möglichkeit gegeben wurde, mit dem digitalen Whiteboard zu interagieren: zum einen durch die Einführung in die Notation, die bereits auf dem digitalen Whiteboard lief und vom Probanden selbst gesteuert werden konnte, zum anderen durch die Möglichkeit, nach der Einführung und vor Beginn des eigentlichen Experiments das UI auszuprobieren. Demgegenüber steht, dass der Einsatz eines Smartphones oder Tablets die Menge denkbarer Gesten eingeschränkt hätte, da diese Geräte typischerweise in einer Hand oder gar beiden Händen gehalten werden. Der Einsatz eines digitalen Whiteboards erlaubt also größere Freiheitsgrade.

Im Laufe einer Sitzung setzt zwangsläufig ein Lerneffekt ein: Probanden können nach einiger Zeit besser mit der jeweiligen Notation umgehen und diese schneller verstehen. Das kann Auswirkungen auf das Ergebnis der Studie haben, insbesondere wenn die verwendeten Gesten in einer festgelegten Reihenfolge angezeigt werden. Das war in der vorliegenden Studie nicht der Fall: Die Gesten wurden in einer zufälligen Reihenfolge angezeigt. Die Auswirkungen des Lerneffekts auf einzelne Gesten und somit auf das Gesamtergebnis werden so vermindert. Gänzlich ausschließen ließe sich ein Lerneffekt aber nur, wenn jeder Proband nur genau eine Geste bearbeitet. An der vorliegenden Studie hätten dann 96 Probanden teilnehmen müssen (zzgl. der beiden Pilotsitzungen) – ein Aufwand, der in keinem Verhältnis zum Nutzen steht, da der Lerneffekt auch wie beschrieben vermindert werden kann.

8.4.3 Externe Gültigkeit

Die Gesten wurden anonymisiert und nicht mit sprechenden Namen versehen. Dies war nötig, um eventuelle Ablenkung vom eigentlichen Ausdruck in der jeweiligen Notation zu vermeiden. In einem echten Projekt würden Gesten mutmaßlich mit sprechenden Namen versehen werden. Der Einfluss, den sprechende Namen auf die Identifikation der Geste haben, bleibt somit unklar.

Für die Studie wurde eine Menge selbstdefinierter Gesten verwendet und nicht auf die üblichen De-Facto-Standardgesten zurückgegriffen, die von Frameworks wie Android oder iOS unterstützt werden (vgl. Kap. 4.2). Somit bleibt der Einfluss der Unbekanntheit der Gesten unklar bzw. die folgende Konsequenz für bekannte Standardgesten. Es kann sein, dass Probanden bekannte Gesten besser erkannt hätten. Dem steht gegenüber, dass das Ziel der Studie die Identifikation unbekannter Gesten war.

9 Prägnanz und Genauigkeit verschiedener Paradigmen zur Beschreibung von Gesten

GestureCards sind als Werkzeug zur Beschreibung von Gesten konzipiert, das unterschiedliche Stakeholder eines Teams als Basis für die Kommunikation und Definition einsetzen können. Entsprechend ist wichtig, dass die Notation von unterschiedlichen Nutzern auch akzeptiert wird. Im Rahmen dieser Studie wird das subjektive Empfinden der Probanden für zwei[1] Faktoren untersucht:

- *Prägnanz*: Die Probanden prüfen die Darstellung der Gesten in unterschiedlichen Notationen. Dabei sollen sie unklare oder mehrdeutige Elemente erkennen oder Elemente identifizieren, die nicht zur Beschreibung der Geste, sondern zur Strukturierung der Notation dienen.

- *Genauigkeit*: Die Probanden prüfen, wie exakt eine vorgeführte Geste durch eine Notation dargestellt wird. Insbesondere sollen sie feststellen, ob alle notwendigen Informationen zur korrekten Beschreibung der Geste vorhanden sind.

[1] Die Studie enthielt ursprünglich noch zwei weitere Faktoren, *Komplexität* und *Lesbarkeit*, bei deren Erfassung jedoch ein Fehler unterlief. Im weiteren Verlauf werden beide daher nicht berücksichtigt.

Dieser Arbeit liegt die Annahme zugrunde, dass hybride Notationen besser zur Gestenbeschreibung geeignet sind als rein textuelle oder graphische Notationen. In Kap. 8 wurde gezeigt, dass GestureCards einen messbaren Vorteil in Bezug auf Verständnis und Auffassungsgeschwindigkeit im Vergleich mit einer textuellen Notation haben. Diese objektive Messung sagt aber nichts über das subjektive Empfinden potentieller Nutzer bei der Arbeit mit unterschiedlichen Notationen aus. In der nachfolgend beschriebenen Studie soll der Eindruck, den GestureCards im Vergleich zu anderen Ansätzen auf Nutzer machen, geprüft werden. Die zugrundeliegende Fragestellung ist:

Werden hybride Notationen wie GestureCards durch den Einsatz unterschiedlicher Darstellungsweisen für Gesteneigenschaften als prägnanter und genauer wahrgenommen als rein textuelle oder rein graphische Notationen? Erhalten sie daher bei einer subjektiven Bewertung mehr Punkte?

In dieser Studie werden GestureCards als hybride Notation mit einer textuellen und einer graphischen Notation verglichen, um o. g. Fragestellung zu untersuchen. Dazu wurden 38 Probanden mithilfe eines Fragebogens befragt. GestureCards erhielten in Summe mehr Punkte als die beiden anderen Notationen. Darüber zeigen die Ergebnisse, dass, vor allem bei der Bewertung von Prägnanz und Genauigkeit, die Wahrnehmung textueller und graphischer Notationen der Probanden von der jeweiligen Geste beeinflusst wird: Die Bewertungen der einzelnen Gesten schwanken bei diesen Variablen erheblich. Für die Prägnanz wurde, je nach Geste, entweder die textuelle oder die graphische Notation u. U. besser bewertet als GestureCards, jedoch nie beide

gemeinsam. Bei der Genauigkeit hingegen erhielten GestureCards die besten Bewertungen. Sowohl für die Prägnanz als auch für die Genauigkeit bekamen GestureCards durchgängig gute Bewertungen und auch für alle Gesten auf einem gleichbleibend hohen Niveau – im Gegensatz zu den textuellen und graphischen Vergleichsnotationen.

Die Studie wird analog zur Studie in Kap. 8 beschrieben. So werden zunächst die Grundlagen der Studie dargelegt und der allgemeine Aufbau beschrieben (Kap. 9.1). Daraufhin werden die Ergebnisse der Befragung analysiert (Kap. 9.2) und anschließend diskutiert (Kap. 9.3). Abschließend werden Risiken betrachtet, die die Validität der Ergebnisse potentiell gefährden können, und es wird aufgezeigt, wie versucht wurde, diesen durch den Aufbau der Studie entgegenzuwirken (Kap. 9.4).

9.1 Aufbau der Studie

Die Beschreibung des Studienaufbaus folgt erneut – analog zur in Kap. 8 beschriebenen Studie – der Struktur, die Jedlitschka et al. (2008) vorschlagen. Im Folgenden werden somit die Ziele der Studie, die teilnehmenden Probanden, die zu lösenden Aufgaben – also die verwendeten Gesten –, der eingesetzte Fragebogen, die erfassten Variablen, die Vorgehensweise während der Studie und die Analyse der erfassten Daten beschrieben.

9.1.1 Ziele

In der Studie wurden Probanden nach ihrer subjektiven Einschätzung zu unterschiedlichen Notationen befragt. Mithilfe der erfassten Daten soll die Wahrnehmung der Probanden mit Blick auf die den Notationen zugrundeliegenden Paradigmen untersucht werden. Entsprechend wurde versucht, Probanden zu gewinnen, die über grundlegende Erfahrung mit Gestensteuerung und Software-Entwicklung im Allgemeinen verfügen, aber nicht mit der Notation von Gesten im Speziellen (vgl. Kap. 9.1.2).

Die Probanden sollten im Wesentlichen zwei Faktoren bewerten: die Prägnanz und die Genauigkeit der Notationen. Um den Probanden eine möglichst einheitliche Vorstellung dieser Begriffe zu vermitteln, wurden sie mit Fragen umschrieben, die festlegten, welche Antworten sich die Probanden zur Bewertung der Notationen geben sollten.

Der Begriff Prägnanz wurde mit folgenden zwei Fragen umschrieben:

Wie umständlich wird die vorgeführte Geste dargestellt? Werden in der Notation Elemente verwendet, die unklar sind oder nicht zu den Kernelementen der Geste gehören?

Der Begriff der Genauigkeit wurde mit den folgenden Fragen umschrieben:

Wie exakt wird die vorgeführte Geste dargestellt? Sind alle notwendigen Informationen enthalten oder fehlen welche, die für das Verständnis der Geste wesentlich sind?

Darüber hinaus wurden die Probanden gebeten, die drei Notationen für jede Geste in eine absteigende Reihenfolge zu bringen, beginnend mit dem persönlichen Favoriten. Vorgaben, nach welchen Kriterien der Favorit bestimmt werden sollte, wurden nicht gemacht – die Probanden sollten subjektiv entscheiden, wie sie die Notationen für welche Geste einschätzen.

GestureCards wurden in dieser Studie wieder GeForMT (vgl. Kap. 6.2.4) gegenübergestellt, da Kammer et al. (2015) für die zu prüfenden Faktoren über gute Ergebnisse für GeForMT im Vergleich zu anderen textuellen Notationen berichten. Darüber hinaus sind auch die bereits für die erste Studie in Kap. 8.1.1 gelisteten Gründe gültig. Außerdem wurde für diese Studie noch Monox (vgl. Kap. 6.3.3) hinzugezogen, um GestureCards auch mit einem Vertreter rein graphischer Notationen zu vergleichen. Im Gegensatz zu anderen Notationen, deren Ziel auch die Konfiguration eines Gestenerkenners ist, fokussiert Monox die Dokumentation von Gesten. Dadurch können die Kompromisse, die andere – vor allem textuelle – Notationen in Bezug auf Maschinenlesbarkeit eingehen müssen, ignoriert und stattdessen ein Fokus auf die Kommunikation der eigentlichen Geste gelegt werden.[2] Monox wurde gewählt, da es die einzige graphische Notation ist, zu der im Rahmen dieser Arbeit Validierungsdaten idenzifiziert werden konnten, die zeigen, dass die Notation bei Anwendern positiv aufgenommen wird.

[2] Vgl. hierzu auch Kap. 2.1.3 und insbesondere Abb. 2.7, bei der die meisten Notationen eher auf der linken Seite (Maschine) zu finden sind und Monox eindeutig auf der rechten Seite (Mensch) positioniert werden kann.

9.1.2 Probanden

An der Studie nahmen 38 Probanden anonym teil, die einen Fragebogen zur Bewertung der Gesten erhielten. Außerdem wurden einige demographische Daten erfasst und die Probanden nach ihrer Erfahrung in der Nutzung und Entwicklung von Anwendungen mit Multi-Touch-Gesten befragt.

Alle Probanden waren Studierende der Universität Duisburg-Essen, besuchten im Rahmen ihres Studiums die Veranstaltung *Mobile Software Engineering* und strebten einen Bachelor-Abschluss an. 21 Probanden studierten Angewandte Informatik, 16 Probanden Wirtschaftsinformatik und ein Proband ein anderes Fach. Im Schnitt waren die Probanden im siebten Semester ihres Studiums, wobei die Angaben zwischen dem vierten und 21. Semester schwanken (33 Probanden gaben an, in welchem Semester sie aktuell studierten; fünf machten keine Angabe). Außerdem waren die Probanden im Schnitt 23,5 Jahre alt. Die Altersangaben schwanken zwischen 20 und 31 Jahren (32 Probanden gaben ihr Alter an und sechs machten keine Angabe). Auf eine Erfassung des Geschlechts wurde verzichtet, da zu Beginn der Studie festgestellt wurde, dass nur vier Frauen anwesend waren und somit die Anonymität der weiblichen Teilnehmer zu leicht aufgehoben werden könnte.

Tab. 9.1 zeigt, dass die Probanden überwiegend Erfahrung mit Multi-Touch-Anwendungen hatten, jedoch größtenteils klassische WIMP-UIs bevorzugten. Darüber hinaus wird deutlich, dass die meisten Probanden keine Erfahrung mit der Entwicklung multi-touch-basierter Anwendungen hatten. Es ist somit anzunehmen, dass bei der überwiegenden Zahl der Probanden ein grundlegendes Verständnis für Gestensteuerung vorhanden war und sie auch Erfahrungen im praktischen Einsatz hatten, sie jedoch keinen oder nur sehr wenig Kon-

Tabelle 9.1: Erfahrung in der Bedienung und Entwicklung von Multi-Touch-Anwendungen aller Probanden

Bevorzugtes Interaktionsparadigma	
Paradigma	Anzahl
Intensiv WIMP	3
Eher WIMP	22
Eher Multi-Touch	12
Intensiv Multi-Touch	1

Erfahrung mit Multi-Touch-Anwendungen	
Erfahrung	Anzahl
Gar keine	3
Gelegentlich	9
Regelmäßig	26
Vorwiegend	0

Erfahrung in der Entwicklung von Multi-Touch-Anwendungen	
Erfahrung	Anzahl
Keine	33
Etwas (1–2 Anwendungen)	2
Viel (3–5 Anwendungen)	1
Sehr viel (>5 Anwendungen)	2

takt mit Notationen zur Beschreibung von Gesten hatten. Die Daten zeigen außerdem eine leichte Korrelation zwischen dem bevorzugten Interaktionsparadigma und der Erfahrung mit Multi-Touch-Anwendungen: Je häufiger Probanden Multi-Touch-Steuerung einsetzten, desto eher bevorzugten sie diese auch (Spearmans Rangkorrelationskoeffizient[3]; $\rho = 0,346$; $p = 0,033$).

[3] Nach Bourier (2012, S. 218) wird – wie im vorliegenden Fall – Spearmans Rangkorrelationskoeffizient zur Ermittlung von Korrelationen zwischen ordinalskalierten Daten eingesetzt.

9.1.3 Eingesetzte Gesten

Für die Studie wurden vier[4] Gesten eingesetzt, die im Folgenden erläutert werden. Jede Geste wurde – im Rahmen der Möglichkeiten der jeweiligen Notation – in GestureCards, GeForMT und Monox beschrieben. Analog zu der in Kap. 8 beschriebenen Studie wurden die Gesten anonymisiert, um eine Beeinflussung der Probanden durch die Namensgebung zu vermeiden. Die Namen der eingesetzten Gesten wurden in allen Notationen entweder durch die neutrale Bezeichnung *Gesture* gefolgt von einer Nummer oder alphabetisch sortierte Buchstaben ohne erkennbare Relation zum Namen ersetzt.

[4] In der Studie wurden eigentlich fünf Gesten eingesetzt, jedoch wurde eine Geste aufgrund eines Darstellungsfehlers (die Teilgesten wurden bei einer Notation nicht anonymisiert) bei der Auswertung der Studie nicht berücksichtigt.

Geste Two-Finger-Swipe

Die Geste beschreibt eine schnell ausgeführte Linie nach rechts mit zwei Fingern, also eine Multi-Touch-Geste. Die Geschwindigkeit ist in GeForMT nicht darstellbar; in GestureCards kann sie durch die Eigenschaft `Speed` und in Monox durch die doppelte Pfeilspitze abgebildet werden. Darüber hinaus zeigt die Geste, wie unterschiedlich die Information, dass zwei Finger parallel verwendet werden, in allen Notationen dargestellt wird: in GeForMT durch die Angabe `2F`, in GestureCards durch die beiden nebeneinander angeordneten Teilgesten und in Monox durch die beiden Striche, die den Schaft des Pfeils kreuzen. Tab. 9.2 zeigt die entsprechenden Beschreibungen in allen drei Notationen.

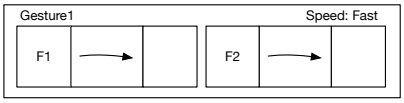
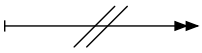
Notation	Darstellung
GeForMT	Gesture1 = 2F(LINE_E)
GestureCards	
Monox	

Tabelle 9.2: Die Geste Two-Finger-Swipe ausgedrückt in GeForMT, GestureCards und Monox

Geste Lang-Kurz-Lang-Kurz (LSLS)

Eine rhythmische Geste, bei der mit einem Finger im Rhythmus Lang-Kurz-Lang-Kurz die Oberfläche berührt wird. Die Geste kann in allen drei Notationen vollständig beschrieben werden, wie Tab. 9.3 zeigt. Auch an diesem Beispiel wird deutlich, wie die drei verglichenen Notationen völlig unterschiedlichen Paradigmen folgen und sich somit die jeweiligen Ausdrücke wesentlich in Aufbau und Struktur unterscheiden. Diese Geste wurde schon für die in Kap. 8 beschriebene Studie verwendet.

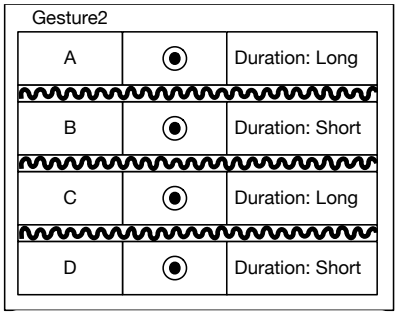
Notation	Darstellung
GeForMT	Gesture2 = 1F(HOLD;POINT;HOLD;POINT)
GestureCards	
Monox	

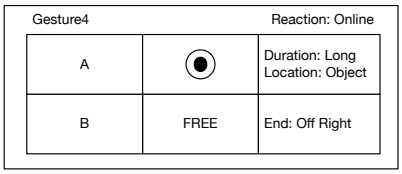
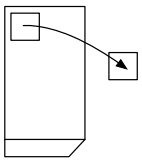
Tabelle 9.3: Die Geste LSLS ausgedrückt in GeForMT, GestureCards und Monox

Geste Drag-Out

Die Geste stellt eine klassische Drag'n'Drop-Aktion dar, die häufig zur Verschiebung virtueller Objekte eingesetzt wird. Im vorliegenden Fall soll das Objekt aber über den Rand des Bildschirms hinausgezogen werden (z. B., um es zu löschen). In GestureCards wird dies durch die Eigenschaft `End: Off Right` beschrieben und in Monox durch das Ende des Pfeils außerhalb der dargestellten Hardware. GeForMT hat hierfür kein vorgegebenes Schlüsselwort. Wesentlich ist, dass die Reaktion zur Laufzeit erfolgt (Online-Geste, vgl. Kap. 5.1.11), was in GeForMT durch das angehängte Schlüsselwort `[ON]`, in GestureCards durch die Eigenschaft `Reaction` und in Monox

durch die Kästchen am Anfang und Ende des Pfeils dargestellt wird. Interessant ist auch, dass der Bewegungspfad der Geste dynamisch ist; in GeForMT wird dazu das Schlüsselwort `MOVE` und in GestureCards das Schlüsselwort `FREE` anstelle der graphischen Repräsentation verwendet. Monox hat kein entsprechendes Symbol, sondern lediglich Pfeile; der variable Bewegungspfad fehlt hier also, wie in Tab. 9.4 zu sehen ist.

Tabelle 9.4: Die Geste Drag-Out ausgedrückt in GeForMT, GestureCards und Monox

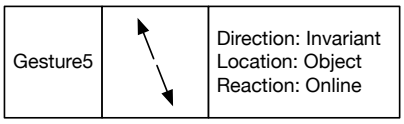
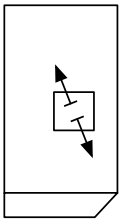
Notation	Darstellung
GeForMT	Gesture4 = 1F(HOLD(Object),MOVE(Object)):[ON]
GestureCards	
Monox	

Geste Spread

Den Abschluss bildete eine typische Multi-Touch-Geste: die *Spread*-Geste, die oft zur Vergrößerung virtueller Objekte oder zum Zoomen eingesetzt wird und häufig Teil des standardmäßig unterstützten Gestensatzes von Frameworks ist. Die Herausforderung ist, dass sowohl in GeForMT als auch in GestureCards spezifiziert ist, dass die Bewegungsrichtung dynamisch ist (in GeForMT durch Auslassen der Richtungsangabe nach dem Schlüsselwort `LINE` und in GestureCards durch die Eigenschaft `Direction: Invariant`[5]). In Monox geht diese Information verloren, wie Tab. 9.5 zeigt.

[5] Die Studie wurde durchgeführt, bevor `Direction: Invariant` gegen `Orientation: None` ausgetauscht wurde, daher wird für die Beschreibung dieser Studie an der veralteten Variante festgehalten.

Tabelle 9.5: Die Geste Spread ausgedrückt in GeForMT, GestureCards und Monox

Notation	Darstellung
GeForMT	Gesture5 = SPREAD[1F(LINE)*1F(LINE)](Object):[ON]
GestureCards	
Monox	

9.1.4 Fragebogen

Alle Daten wurden mithilfe eines Fragebogens erfasst, den die Probanden eigenständig ausfüllten. Der Fragebogen war aufgeteilt in drei Bereiche: *Einführung*, *Gestennotationen* und *Abschluss*. In Anh. C ist der vollständige Fragebogen mit allen Seiten zu finden.

Im Bereich *Einführung* wurden grundlegende demographische Daten wie Alter und Studiengang erfasst, die bereits in Kap. 9.1.2 angesprochen wurden. Darüber hinaus wurden Erfahrungswerte mit Gestensteuerung erfragt, um einen Überblick über den Kenntnisstand der Probanden zu gewinnen. Die Probanden wurden gebeten, ihre Erfahrungen mit der Bedienung und der Entwicklung multi-touch-basierter Anwendungen auf einer Likert-Skala zu klassifizieren. Außerdem sollten sie ihr bevorzugtes Interaktionsparadigma – Multi-Touch oder WIMP – angeben.

Der Bereich *Gestennotationen* erläuterte zunächst die grundlegenden Begriffe (Prägnanz und Genauigkeit) anhand der Fragen, die zu Beginn des Kapitels vorgestellt wurden, und fasste das Vorgehen während der Studie zusammen. Die Probanden wurden hier auch aufgefordert, ihre Gedanken zu jeder Notation auf dem Fragebogen zu notieren. Anschließend begann der eigentlich zur Erfassung der Daten vorgesehene Teil der Studie. Für jede Geste war ein separater Teil im Fragebogen verfügbar, der die Beschreibung aller Gesten in den drei Notationen und die zur Bewertung verwendeten Likert-Skalen enthielt. Die Likert-Skalen waren in sechs Stufen (aufsteigend von −−− bis +++) unterteilt. Zu guter Letzt konnten die Probanden für jede Geste auch die Notationen in eine persönliche Reihenfolge bringen.

Abschließend konnten die Probanden im Bereich *Abschluss* ihren Eindruck von der Komplexität und der Lesbarkeit der Notationen im Allgemeinen bewerten. Dieser Teil wurde jedoch aus zwei Gründen letztlich nicht in die Ergebnisse der Studie aufgenommen:

- Die Likert-Skala zur Lesbarkeit wurde falsch benannt, sodass – trotz mündlicher Erläuterung – unklar ist, ob die Probanden die Lesbarkeit bewertet haben oder sich nach der falschen Beschriftung richteten.
- Zu jeder Notation wurde eine Geste als Beispiel gezeigt, jedoch ist unklar, ob die Probanden tatsächlich ihren allgemeinen Eindruck oder lediglich diese Geste bewerteten.

Die Zweifel an der Gültigkeit der Ergebnisse für diese beiden Punkte überwiegen, daher wurden sie nicht in der Studie berücksichtigt.

9.1.5 Hypothesen und Variablen

Jeder Proband bewertete mithilfe des in Kap. 9.1.4 beschriebenen Fragebogens alle Gesten, die in den drei Notationen GeForMT, GestureCards und Monox definiert wurden. Ebenso wie bei der in Kap. 8 beschriebenen Studie handelt es sich daher bei dieser Studie um eine zweifaktorielle Studie, jedoch in diesem Falle mit den zwei Innersubjektfaktoren *Notation* und *Geste*, da die

Faktor	Nullhypothese H_0	Alternativhypothese H_1
Notation	Die Probanden bewerten die Genauigkeit aller Notationen gleich	Die Probanden bewerten die Genauigkeit von GestureCards besser als die der anderen Notationen
Geste	Die Probanden bewerten die Genauigkeit der Notationen für alle Gesten gleich	Die Probanden bewerten die Genauigkeit von GestureCards für alle Gesten besser als die der anderen Notationen
Interaktion	Die Interaktion der Faktoren Geste und Notation hat keinen Einfluss auf die Bewertung der Probanden	Die Interaktion der Faktoren Geste und Notation beeinflusst die Bewertung der Probanden

Tabelle 9.6: Überblick über alle Hypothesen der zweiten Studie für die Variable Genauigkeit

Probanden nicht einem Faktor zugeteilt werden, sondern alle identische Aufgaben bearbeiten. Die Faktoren können außerdem miteinander interagieren, somit muss die *Interaktion* zwischen den beiden Faktoren selbst als Faktor betrachtet werden. Da jeder Proband alle Gesten in allen drei Notationen bewertete, handelt es sich außerdem um eine Studie mit Messwiederholung.

Die Probanden bewerteten die Prägnanz und Genauigkeit der drei Notationen und brachten sie in eine persönlich favorisierte Reihenfolge. Somit ergeben sich folgende drei Variablen: Prägnanz, Genauigkeit und Favorit. Die der Analyse zugrundeliegenden Hypothesen müssen die Faktoren und die Variablen berücksichtigen, wodurch sich je neun Null- und Alternativhypothesen ergeben. Die einzelnen Hypothesen werden im Folgenden durch Zusatz der Variablen und des Faktors identifiziert: $H_{0GestePrägnanz}$ beschreibt also für den Faktor Geste die Nullhypothese zur Variable Prägnanz usw. Tab. 9.6 zeigt eine Übersicht der Null- und Alternativhypothesen für die Variable Genauigkeit, die beiden Faktoren sowie deren Interaktion. Die Hypothesen für die Variablen Prägnanz und Favorit sind analog zu formulieren, werden aber der besseren Übersicht halber an dieser Stelle nicht detailliert beschrieben.

9.1.6 Durchführung

Im Rahmen der Studie wurden den Probanden die in Kap. 9.1.3 beschriebenen Gesten als Video vorgeführt, das auf einem Projektor für alle sichtbar abgespielt wurde. Zu jedem Video wurde den Probanden eine Beschreibung der Geste in den drei Notationen zur Verfügung gestellt. Alle Beschreibungen waren – im Rahmen der Möglichkeiten der einzelnen Notationen – korrekt, d. h. sie definierten alle Gesten mit den verfügbaren Mitteln. Dies hat zwangsläufig zur Folge, dass manche Eigenschaften nicht abgebildet werden können. So definieren z. B. weder Monox noch GeForMT notwendige Notationselemente, um den Druck auf die Oberfläche zu beschreiben. Von einer Erweiterung der Notationen, mit der ohne Weiteres die fehlenden Eigenschaften hätten hinzugefügt werden können, wurde abgesehen, da eine Erweiterung von Monox und GeForMT durch den Autor von GestureCards die Objektivität gefährdet hätte und als potentielles Risiko für die Validität eingestuft wurde.

Die Studie war eingebettet in eine Lehrveranstaltung über Gestensteuerung auf mobilen Geräten. Zunächst wurden den Probanden die Grundlagen von

Gestensteuerung sowie ihre Implementierung unter Android vermittelt. Anschließend wurde auf die Notwendigkeit von Spezifikation eingegangen, insbesondere wenn keine von Android vorgegebenen Gesten verwendet werden. An dieser Stelle wurde bewusst auf die detaillierte Einführung in unterschiedliche Notationen verzichtet, um die Probanden unvorbelastet zu lassen.

Im nächsten Schritt wurden den Probanden der Reihe nach alle Videos vorgespielt und die passenden Beschreibungen in den drei gewählten Notationen gezeigt. Die Probanden hatten drei Minuten Zeit, um die gezeigte Geste und die Beschreibungen zu vergleichen und Prägnanz und Genauigkeit auf einer sechsstufigen Likert-Skala zu bewerten. Das Video der Geste wurde dabei in einer Endlosschleife gezeigt. Da wichtige Eigenschaften von Gesten im Video u. U. nur schwer zu erkennen waren, wurden alle Informationen über die Geste auch textuell als einfache Beschreibung eingeblendet. Abb. 9.1 zeigt ein Beispiel. Darüber hinaus wurden die Probanden gebeten, die drei Notationen für jede Geste nach persönlicher Vorliebe zu gewichten und ggf. persönliche Kommentare zu jeder Notation hinzuzufügen.

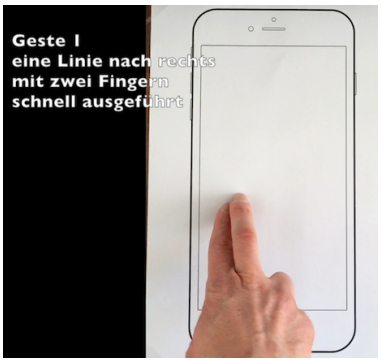


Abbildung 9.1: Szene aus dem vorgeführten Video für die Geste Two-Finger-Swipe.

9.1.7 Analyse

Zur Analyse wurde auf eine zweifaktorielle Varianzanalyse (ANOVA) mit Messwiederholung zurückgegriffen. Dieser Test vergleicht als *Omnibustest* die Ergebnisse mehrerer Stichproben hinsichtlich ihrer zentralen Tendenz, macht jedoch keine Aussage über die konkrete Ausprägung (Janssen & Laatz, 2013). Vereinfacht ausgedrückt – und auf diese Studie übertragen – sagt der Test lediglich aus, dass es zwischen den Ergebnissen der einzelnen Notationen Unterschiede gibt, aber nicht, wie diese ausgeprägt sind. Zur Erfassung der konkreten Ausprägungen müssen paarweise Vergleiche der Notationen durchgeführt werden (auch als *Post-Hoc-Tests* bezeichnet). Dabei wird – wenn nicht anders beschrieben – auf *Fisher's Least Significant Difference* zurückgegriffen. In einigen anderen Fällen – insbesondere zur Analyse der Interaktion zwischen Faktoren – kommen hingegen paarweise t-Tests zum Einsatz.

Zunächst wurde eine statistische Analyse der Variablen Prägnanz, Genauigkeit und Favorit mit den o. g. Tests durchgeführt. Diese wurden über den Fragebogen erfasst und zur besseren Auswertung in eine numerische Repräsentation übertragen. Dazu wurden Werte von 1 (für die schlechteste Beurteilung; auf den zugehörigen Likert-Skalen immer der Wert ganz links) bis 6 (für die beste Beurteilung; auf den zugehörigen Likert-Skalen immer der Wert ganz rechts) vergeben. Zur Bestimmung der Variablen Favorit wurde außerdem die Rangfolge, die die Probanden gebildet haben, in Punkte übersetzt: Rang 1 (Favorit) bekam drei Punkte, Rang 2 zwei Punkte und Rang 3 einen Punkt. Durch die Bestimmung der persönlichen Rangfolge konnten für jeden Probanden so 24 Punkte vergeben werden (für vier Gesten jeweils sechs Punkte; maximal 12 Punkte für eine Notation), die dann in Summe eine Gesamtpunktzahl sowie eine direkte Gegenüberstellung der drei Notationen ergeben.

Im Folgenden soll erneut das Signifikanzniveau von 5 % bzw. 1 % verwendet werden. Außerdem wird wieder in kleine, mittlere und große Effektstärken eingeteilt, wie Cohen (1988, S. 25f) vorgeschlagen hat.

9.2 Ergebnisse

Im Folgenden werden die Ergebnisse der Studie beschrieben. Dazu werden die drei Variablen Prägnanz, Genauigkeit und persönliche Präferenz – in Gestalt der gewählten Favoriten – einzeln betrachtet und die identifizierten Faktoren analysiert. Auf die Unterschiede zwischen den einzelnen Notationen wird dabei explizit eingegangen.

9.2.1 Prägnanz

Abb. 9.2 zeigt, wie sich die Bewertung der Prägnanz nach Geste und Notation verteilt. Unterschiede sind je nach Geste deutlich; insbesondere GeForMT und Monox schwanken in den Bewertungen zwischen den einzelnen Stufen, während GestureCards weniger zwischen den – im Verhältnis – hohen Bewertungen wechseln und eher auf einem Level bleiben.

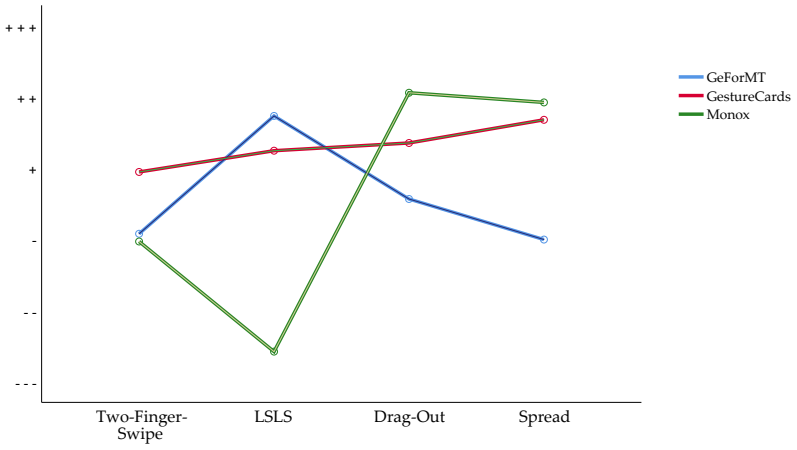


Abbildung 9.2: Interaktionsdiagramm der durchschnittlichen Bewertung der Prägnanz für alle Notationen je Geste.

Aggregiert man die Ergebnisse der einzelnen Gesten, so erhält man die Gesamtergebnisse der einzelnen Notationen. Diese unterscheiden sich voneinander ($F(2,72) = 11,522$; $p < 0,001$; ${\eta_p}^2 = 0,242$) mit einer großen Effektstärke, somit ist $H_{0NotationPrägnanz}$ abzulehnen. Im paarweisen Vergleich der drei Notationen ist für GeForMT ($M = 3,595$; 95 % CI $[3,269; 3,893]$) und Monox ($M = 3,595$; 95 % CI $[3,416; 3,773]$) kein Unterschied festzustellen.[6] GestureCards ($M = 4,304$; 95 % CI $[4,007; 4,601]$) hingegen wurden im Durchschnitt $0,709$ Punkte besser bewertet als GeForMT ($p = 0,001$) und $0,709$ Punkte besser als Monox ($p < 0,001$).

[6] Der identische Durchschnittswert für GeForMT und Monox ist Zufall. Die detaillierten Ergebnisse sind in Anh. E zu finden.

Der Einfluss der Geste ist ein messbarer Faktor ($F(3,108) = 15,57$; $p < 0,001$; ${\eta_p}^2 = 0,302$), bei dem eine große Effektstärke zu erkennen ist (${\eta_p}^2 > 0,14$). $H_{0GestePrägnanz}$ ist also abzulehnen. Der Einfluss der Geste auf die Bewertungen der Probanden wird für die einzelnen Notationen mit einer einfaktoriellen ANOVA mit Messwiederholung überprüft. Im Kern kommt also das gleiche Verfahren wie bei der Gesamtanalyse zum Einsatz, jedoch lediglich unter Berücksichtigung der Ergebnisse einer Notation. Die Analyse zeigt sowohl für GeForMT ($F(3,111) = 16,141$; $p < 0,001$; ${\eta_p}^2 = 0,304$)

als auch für Monox ($F(3,108) = 93,431$; $p < 0,001$; ${\eta_p}^2 = 0,722$) einen erkennbaren Einfluss des Faktors Geste mit großer Effektstärke. Das schlägt sich in der paarweisen Betrachtung der Gesten je Notation nieder: Während sich bei GeForMT lediglich die Geste LSLS von den anderen unterscheidet, sind bei Monox zwischen allen Gesten statistische Unterschiede zu sehen, außer zwischen Drag-Out und Spread. Für GestureCards hingegen zeigt die Überprüfung der Varianz mit der einfaktoriellen ANOVA über alle Gesten keine signifikanten Unterschiede.

Bei den Faktoren Geste und Notation ist auch eine Interaktion ($F(6,216) = 44,611$, $p < 0,001$, ${\eta_p}^2 = 0,553$) mit einer großen Effektstärke sichtbar. Auch $H_{0InteraktionPrägnanz}$ ist daher abzulehnen. Die Interaktion der beiden Faktoren hat außerdem eine größere Effektstärke als einzelne Faktoren. Im Folgenden sollen die Ergebnisse der Notationen paarweise bei ausgewählten Gesten miteinander verglichen werden – immer dann, wenn zwei Notationen nach Abb. 9.2 höher bewertet wurden als eine andere und diese beiden auch nahe beieinanderliegen. So wurden bei der Geste LSLS GeForMT ($M = 4,74$; 95 % CI $[4,36; 5,11]$) und GestureCards ($M = 4,18$; 95 % CI $[3,67; 4,7]$) höher bewertet als Monox ($M = 1,55$; 95 % CI $[1,17; 1,94]$), jedoch ist zwischen GeForMT und GestureCards kein signifikanter Unterschied festzustellen. Bei der Geste Spread wurden Monox ($M = 4,92$; 95 % CI $[4,52; 5,32]$) und GestureCards ($M = 4,68$; 95 % CI $[4,31; 5,04]$) höher bewertet als GeForMT ($M = 3$; 95 % CI $[2,52; 3,48]$), jedoch kann zwischen GestureCards und Monox für diese Geste auch kein signifikanter Unterschied festgestellt werden. Bei den anderen zwei Gesten ist der direkte Vergleich der bestbewerteten Notation mit den beiden anderen signifikant. GestureCards erhielten im Schnitt die beste Bewertung.

9.2.2 Genauigkeit

Abb. 9.3 zeigt, wie die Probanden die Genauigkeit der Notationen aufgeteilt nach Gesten bewerteten. Im Wesentlichen bestätigt sich das Bild, das auch die Auswertung der Prägnanz zeigt: Während GeForMT und Monox abhängig von der jeweiligen Geste stark schwanken, wurden GestureCards durchgehend gut und für alle Gesten auf fast gleichem Niveau bewertet.

Zwischen den Gesamtergebnissen der Notationen besteht ein sehr signifikanter Unterschied ($F(2,72) = 32,272$, $p < 0,001$, ${\eta_p}^2 = 0,473$) mit einer großen Effektstärke. $H_{0NotationGenauigkeit}$ ist demnach abzulehnen. Im paarweisen Vergleich ist zwischen GeForMT ($M = 3,527$; 95 % CI $[3,291; 3,763]$) und Monox ($M = 3,358$; 95 % CI $[3,149; 3,567]$) kein signifikanter Unterschied festzustellen. GestureCards ($M = 4,480$; 95 % CI $[4,223; 4,736]$) hingegen wurden – analog zur Prägnanz – besser bewertet: um $0,953$ Punkte besser als GeForMT ($p < 0,001$) und um $1,122$ Punkt besser als Monox ($p < 0,001$).

Betrachtet man die Unterschiede für den Faktor Geste, so zeigen die Ergebnisse, dass sehr signifikante Unterschiede ($F(3,108) = 11,184$, $p < 0,001$, ${\eta_p}^2 = 0,237$) und eine große Effektstärke vorliegen. Auch $H_{0GesteGenauigkeit}$ ist also abzulehnen. Analog zur Betrachtung der Prägnanz soll auch für die Genauigkeit je Notation eine einfaktorielle ANOVA über alle Gesten Aufschluss

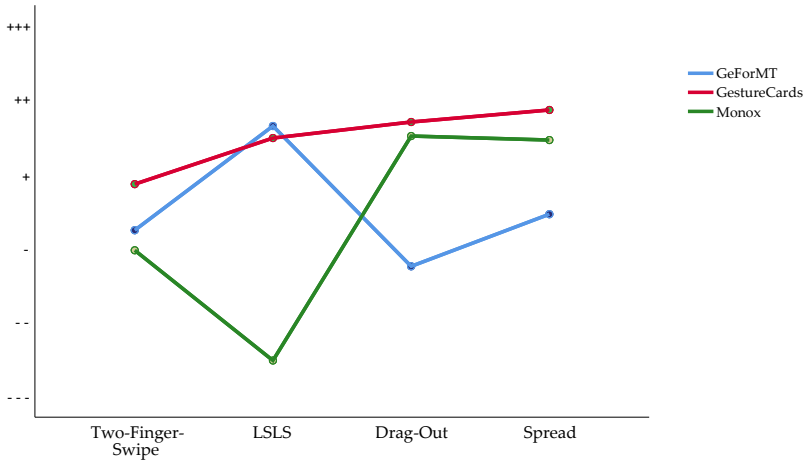


Abbildung 9.3: Interaktionsdiagramm der durchschnittlichen Bewertung der Genauigkeit für alle Notationen je Geste.

über den Einfluss der einzelnen Gesten geben. Für alle drei Notationen sind signifikante Unterschiede sichtbar, jedoch mit schwankenden Effektstärken. Während für GeForMT ($F(3,111) = 17,486$, $p < 0,001$, ${\eta_p}^2 = 0,321$) und Monox ($F(3,108) = 57,01$, $p < 0,001$, ${\eta_p}^2 = 0,586$) eine große Effektstärke vorliegt, kann für GestureCards ($F(3,108) = 4,611$, $p = 0,004$, ${\eta_p}^2 = 0,114$) nur eine mittlere Effektstärke beobachtet werden. Die paarweise Betrachtung der Gesten je Notation zeigt, dass sich bei Monox mit Ausnahme der Gesten Drag-Out und Spread alle Gesten signifikant in der Bewertung unterscheiden, während sich bei GeForMT insbesondere für Geste LSLS ein signifikanter Unterschied zu allen anderen Gesten beobachten lässt. Ansonsten gibt es bei GeForMT nur vereinzelte signifikante Unterschiede zwischen den Gesten. Bei GestureCards hingegen unterscheidet sich aussschließlich die Geste Two-Finger-Swipe signifikant von allen anderen.

Beide Faktoren interagieren mit großer Effektstärke ($F(6,216) = 32,416$, $p < 0,001$, ${\eta_p}^2 = 0,474$). $H_{0InteraktionGenauigkeit}$ ist demnach ebenfalls abzulehnen. Die Ergebnisse der Notationen für einzelne Gesten zeigen ein ähnliches Bild wie für die Prägnanz, entsprechend werden paarweise Vergleiche wieder nur zwischen nahe beieinander liegenden Ergebnissen durchgeführt. Das ist erneut bei der Geste LSLS der Fall: GeForMT ($M = 4,61$; 95 % CI $[4,2;5,01]$) und GestureCards ($M = 4,5$; 95 % CI $[4,08;4,92]$) werden ähnlich gut bewertet, im Gegensatz zu Monox ($M = 1,5$; 95 % CI $[1,17;1,83]$); der paarweise Vergleich von GeForMT und GestureCards zeigt aber keinen messbaren Unterschied. Bei der Geste Drag-Out wurden Monox ($M = 4,55$; 95 % CI $[4,19;4,92]$) und GestureCards ($M = 4,74$; 95 % CI $[4,29;5,19]$) besser bewertet als GeForMT ($M = 2,74$; 95 % CI $[2,42;3,06]$), allerdings zeigt ein paarweiser Vergleich von Monox und GestureCards wieder keinen Unterschied. Geste Spread macht keine Ausnahme: Monox ($M = 4,46$; 95 % CI $[4,03;4,89]$) und GestureCards ($M = 4,86$; 95 % CI $[4,5;5,23]$) wurden besser bewertet als GeForMT ($M = 3,46$; 95 % CI $[3,05;3,86]$), ein statistischer Unterschied liegt jedoch nicht vor.

GestureCards wurden im Vergleich zu GeForMT und Monox gut bewertet, jedoch wurde in Abhängigkeit der Gesten immer eine weitere Notation ebenfalls gut bewertet: für die Geste LSLS GeForMT und für die Gesten Drag-Out

und Spread Monox. Im Gesamtergebnis erhielten GestureCards von allen drei Notationen jedoch die beste Bewertung.

9.2.3 Persönliche Präferenz

Im letzten Schritt werden die persönlichen Präferenzen der Probanden untersucht. Die Analyse der zweifaktoriellen ANOVA zeigt keinen signifikanten Einfluss durch die Geste, deshalb muss $H_{0GesteFavorit}$ angenommen werden. Für die Notation ($F(2,74) = 13,664$, $p < 0,001$, ${\eta_p}^2 = 0,270$) hingegen kann ein Einfluss mit einer großen Effektstärke festgestellt werden. Zwischen beiden Faktoren herrscht aber eine sehr signifikante Interaktion ($F(6,222) = 20,305$, $p < 0,001$, ${\eta_p}^2 = 0,354$), ebenfalls mit einer großen Effektstärke. $H_{0NotationFavorit}$ und $H_{0InteraktionFavorit}$ müssen also angenommen werden.

Die Betrachtung der Punkte für einzelne Gesten zeigt, dass GestureCards für die Geste Two-Finger-Swipe am besten bewertet wurde, und das sehr signifikant ($p < 0,001$). Für die anderen drei Gesten sind GestureCards knapper Zweiter und werden entweder von GeForMT (Geste LSLS) oder Monox (Gesten Drag-Out und Spread) überholt. Allerdings kann im direkten Vergleich kein signifikanter Unterschied festgestellt werden.

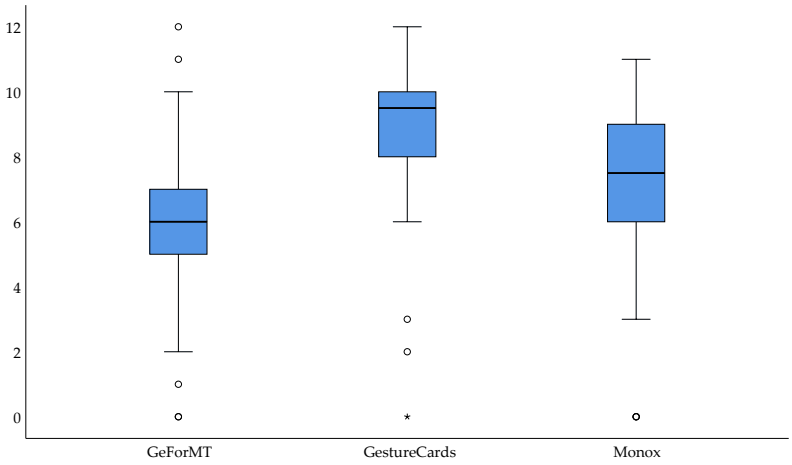


Abbildung 9.4: Aggregierte Punktzahl der persönlichen Präferenz aller Probanden.

Summiert man alle Bewertungen der persönlichen Präferenz zu einem Gesamtergebnis auf, so erhalten GestureCards die meisten Punkte ($M = 8,61$; 95 % CI $[7,75; 9,46]$), gefolgt von Monox ($M = 6,92$; 95 % CI $[6,04; 7,8]$) und GeForMT ($M = 6,05$; 95 % CI $[5,21; 6,9]$). Abb. 9.4 stellt das summierte Gesamtergebnis für alle Gesten pro Notation gegenüber. Zum direkten Vergleich der Ergebnisse wurden paarweise t-Tests durchgeführt. Zwischen Monox und GeForMT ist kein signifikanter Unterschied sichtbar, jedoch zwischen GestureCards und Monox ($t(37) = -3,93$; $p < 0,001$; $d = 0,64$) mit mittlerer und zwischen GestureCards und GeForMT ($t(37) = -4,674$; $p < 0,001$; $d = 0,99$) mit großer Effektstärke.

9.3 Diskussion

Die Ergebnisse zeigen, dass die Probanden die Notationen nicht nur sehr unterschiedlich bewerteten, sondern dass diese Bewertung auch unter Einbezug der aktuell bearbeiteten Geste gesehen werden muss. Insbesondere GeForMT und Monox fluktuierten in ihren Ergebnissen je nach Geste. Die Analyse zeigt, dass der Einfluss durch die Gesten für die Prägnanz bei den Notationen unterschiedlich ausgeprägt ist: Die Bewertung von Monox ist stärker von der jeweiligen Geste abhängig als die Bewertung von GeForMT. Die Bewertung der Prägnanz von GestureCards hingegen ist – zumindest für die verwendeten Gesten – unabhängig. Für die Bewertung der Genauigkeit zeigt sich ein ähnliches Bild. Zwar ist hier auch bei den Bewertungen von GestureCards ein signifikanter Einfluss der Gesten zu erkennen, jedoch hat die detaillierte Analyse gezeigt, dass dieser nur von der ersten Geste – Two-Finger-Swipe – ausgeht; es kann sich also um ein statistisches Artefakt handeln, das z. B. durch fehlende Randomisierung verursacht wurde. Für GeForMT und insbesondere für Monox sind stärkere Einflüsse durch die Geste erkennbar. Auf die Wahl des persönlichen Favoriten hingegen hat die Geste allerdings keinen Einfluss, sodass hier wohl die jeweilige Notation entscheidend ist.

Eine mögliche Schlussfolgerung ist, dass manche Notationen – aus Sicht der Probanden – eher geeignet sind als andere, bestimmte Gesten prägnant und genau zu beschreiben. Dies wird vor allem dadurch betont, dass die Interaktion von Geste und Notation bei der Bewertung der Prägnanz die größte Effektstärke aufwies. Die Konsequenz ist ein Dilemma bei der Auswahl einer Notation für ein Projekt: Es besteht die Gefahr, dass eine Notation gewählt wird, die nicht alle für das Projekt notwendigen Gesten gut beschreiben kann. GestureCards hingegen fluktuierten nicht so stark wie die anderen Notationen in ihrer Bewertung, bekamen aber recht hohe Noten. Für die Variable Prägnanz bewerteten die Probanden Monox und GeForMT für manche Gesten besser, jedoch bevorzugten sie niemals beide. Für die Genauigkeit bevorzugten die Probanden GestureCards. Das zeigt, dass mit GestureCards ein guter Mittelweg möglich ist und die Notation eine Alternative darstellt, wenn es um die Beschreibung von Gesten geht. Die Wahl der persönlichen Favoriten bestätigt dies: GestureCards erhielten in Summe die meisten Punkte.

Die Fluktuationen in den Bewertungen von GeForMT und Monox werfen die Frage auf, warum die Probanden die gleiche Notation für verschiedene Gesten so unterschiedlich bewerteten. Die Betrachtung der Kommentare, Gedanken und Fragen, die die Probanden während der Sitzungen auf den Fragebogen notierten, kann hilfreich sein. Einige Beobachtungen der in Kap. 8 beschriebenen Studie werden dabei bestätigt. Die Probanden waren häufiger unsicher, welche Bedeutung sich hinter Schlüsselwörtern in GeForMT und Symbolen in Monox verbarg. Während sich für GeForMT die Fragen im Prinzip im Vergleich zur vorigen Studie wiederholten, klingen sie für Monox ähnlich: z. B. fragten sich Probanden, was die Bedeutung einer doppelten Pfeilspitze sei oder wie schwarz und weiß gefärbte Kreise zu verstehen sind. Bei GestureCards war insbesondere die geschlängelte Linie, die zur Darstellung von Pausen verwendet wird, Gegenstand von Fragen. Ungeachtet der konkreten Bedeutung von Schlüsselwörtern und Symbolen zeigt die Auswertung der

Variable Genauigkeit jedoch, dass die Probanden fehlende Informationen, die für die exakte Beschreibung der Geste notwendig sind, identifizieren konnten.

Die Ergebnisse – vor allem die Ausreißer – in Abb. 9.4 zeigen auch, dass einige Probanden eine Notation entweder favorisierten oder völlig ablehnten. Mit Blick darauf, dass manche Notationen messbar schwerer zu verstehen sind (vgl. Kap. 8), stellt sich die Frage, wie Entwicklungsteams die Entscheidungen für oder gegen eine bestimmte Notation treffen und wie sie ihre Werkzeuge auswählen. Vor allem für cross-funktionale Teams, die in der Software-Entwicklung an der Tagesordnung sind, ist somit der Einfluss persönlicher Präferenzen unklar und kann ggf. zu Schwierigkeiten im Projektverlauf führen. Obwohl einzelne Gesten für die Studie verwendet wurden, für die die Probanden bessere Noten für GeForMT und Monox vergaben, zeigt die Studie jedoch, dass die Probanden GestureCards im Allgemeinen bevorzugten, vor allem, wenn Gesten unterschiedlicher Art verwendet werden.

9.4 Risiken für die Validität

Diese Studie teilt einige Risiken für die Validität mit der in Kap. 8 beschriebenen Studie: Die Probanden waren (vorwiegend männliche) Studierende und es wurden individuelle, anonymisierte Gesten eingesetzt. Darüber hinaus werden nachfolgend einige für diese Studie spezifische Bedrohungen gelistet.

Im Gegenzug zur Studie aus Kap. 8 wurden in dieser Studie die eingesetzten Notationen nicht im Vorhinein eingeführt und kein Spickzettel mit weiterführenden Informationen zur Verfügung gestellt. Diese Entscheidung erschien notwendig, um eine Beeinflussung der Probanden durch vorab zur Verfügung gestellte Informationen zu vermeiden, die in ein ungewolltes mentales Modell hätten führen können. Es bleibt aber dennoch unklar, ob und wie diese Entscheidung die Ergebnisse evtl. anderweitig beeinflusst hat.

Die Probanden bewerteten die Notationen nach ihrer subjektiven Meinung. Es ist möglich, dass persönliche Präferenzen von unterschiedlichen Faktoren beeinflusst werden, die im Rahmen der Studie nicht kontrollierbar sind. So könnten Probanden mehr Erfahrung mit graphischen Notationen haben (z. B., weil sie vorwiegend graphische Notationen wie UML verwenden) und daher tendenziell graphische Notationen besser bewerten als textuelle. Darüber hinaus kann aus den Ergebnissen nicht abgeleitet werden, ob eine hohe oder geringe Präferenz für oder gegen eine Notationen von kleinen Designentscheidungen abhängt. Es ist denkbar, dass einzelne Schlüsselwörter oder graphische Elemente die Bewertung sehr stark beeinflussen und zu einer guten oder schlechten Bewertung führen. In diesem Falle könnte eine kleine Änderung an der Notation die Bewertung verändern.

Letztlich sind die gewählten Notationen lediglich – wenn auch sorgsam – ausgewählte Repräsentanten ihres jeweiligen Paradigmas. Andere Notationen, die dem gleichen Paradigma folgen, führen vielleicht zu anderen Ergebnissen.

10 Einsatz in der Praxis

Die beiden Studien, die in Kap. 8 und Kap. 9 dargelegt werden, fokussieren den direkten Vergleich unterschiedlicher Notationen zur Beschreibung von Gesten auf Basis verschiedener Kriterien, die für das Verständnis und die Verwendung der Notationen relevant sind. Beide Studien betrachten jedoch die Beschreibung einzelner Gesten ohne den Kontext einer konkreten Anwendung bzw. ohne Berücksichtigung der für die Entwicklung interaktiver Applikationen notwendigen weiteren Elemente. Neben der Definition von Gesten müssen sich Projektteams bei der Entwicklung einer gestenbasierten Anwendung noch mit den konkreten Anwendungsfällen sowie der Gestaltung des UI auseinandersetzen. Ein weiterer Aspekt, der in den Studien aus Kap. 8 und Kap. 9 nicht berücksichtigt wird, jedoch in einem praktischen Kontext von entscheidender Bedeutung ist, ist neben dem Verständnis auch die Definition von Gesten mittels GestureCards. Im Folgenden soll berichtet werden, welchen Eindruck Probanden von GestureCards bei der Lösung praktischer Aufgaben haben.

Mit Blick auf den in Kap. 2 definierten Einsatzzweck von GestureCards soll im Rahmen dieser Studie von einem existierenden Entwurf einer Anwendung sowie bereits definierten Gesten ausgegangen werden. Die Probanden wurden aufgefordert, in zwei Szenarien folgende Aufgabenstellungen zu bearbeiten, die ihnen in einem echten Projekt so oder so ähnlich begegnen können:

- Einen bestehenden Entwurf einer Anwendung um eine neue Funktionalität erweitern und passende Gesten definieren.
- Bestehende Gesten auf ihre Verwendbarkeit beim Einsatz bestimmter Hardware prüfen und ggf. durch passende Alternativen ersetzen.

Es wird angenommen, dass GestureCards als hybride Notation ihre Vorteile, die in den Studien aus Kap. 8 und Kap. 9 gezeigt wurden, auch in einem praktischen Kontext ausspielen können. Die erste Forschungsfrage, die in der Studie beantwortet werden soll, lautet somit:

Können GestureCards bei der Entwicklung gestenbasierter Anwendungen unterstützen?

Darüber hinaus soll die Frage betrachtet werden, wie GestureCards für den praktischen Einsatz weiter aufgewertet werden können. Eine interessante Option ist die Digitalisierung von GestureCards bzw. die damit verbundenen Möglichkeiten. Für die vorliegende Studie wurde daher der Augmentierte Interaction Room (AugIR) (Kleffmann, 2014) um die Möglichkeit erweitert, Gesten mit GestureCards definieren zu können. Diese Integration bietet zwei wesentliche Funktionen, die nicht mit analogen GestureCards und Whiteboards umgesetzt werden können:

- Die zur graphischen Darstellung der Geste verwendeten Pfeile werden als Animation wiedergegeben.
- Die definierten Gesten können mithilfe eines passenden Gestenerkenners (vgl. Kap. 13.1 sowie Hesenius et al. (2017)) in einem Simulationsmodus direkt ausprobiert werden.

Die Probanden bearbeiteten je ein Szenario im AugIR und eines auf klassischen, analogen Whiteboards, um einen Vergleich beider Varianten erfahren zu können. Daraus ergibt sich die zweite Forschungsfrage für diese Studie:

Können GestureCards durch die Möglichkeiten, die die Digitalisierung von Sketching-Umgebungen bietet, aufgewertet werden?

An der Studie nahmen 14 Probanden teil. Die Ergebnisse deuten auf eine starke Präferenz der meisten Probanden für die Arbeit mit dem AugIR hin. Mehrere Probanden hoben insbesondere die Animation der Gesten sowie die Möglichkeit, diese direkt testen zu können, hervor. Jedoch kann eine gewisse Polarisierung beobachtet werden: Eine Minderheit der Probanden sprach sich klar für die analoge Version aus. Allen Probanden ist jedoch gemein, dass sie die Arbeit mit GestureCards als nützlich für die Entwicklung gestenbasierter Anwendungen empfanden.

Die Beschreibung der Studie folgt im Wesentlichen wieder der bereits in Kap. 8 und Kap. 9 verwendeten Struktur. Zunächst werden die Grundlagen der Studie dargelegt und der allgemeine Aufbau, insbesondere die Integration in den AugIR, die eingesetzten Szenarien und der Fragebogen zur Erfassung der Ergebnisse, beschrieben (Kap. 10.1). Anschließend werden die Ergebnisse des Fragebogens sowie verschiedene Beobachtungen während der einzelnen Sitzungen analysiert (Kap. 10.2) und daraufhin diskutiert (Kap. 10.3). Zum Abschluss werden Risiken betrachtet, die die Validität der Ergebnisse potentiell gefährden können, und gezeigt, wie diesen durch den Aufbau der Studie entgegengetreten wurde (Kap. 10.4).

10.1 Aufbau der Studie

Die Beschreibung des Studienaufbaus folgt erneut den Vorschlägen von Jedlitschka et al. (2008). Es werden die Ziele der Studie, die teilnehmenden Probanden, die Integration von GestureCards in den AugIR, die zu lösenden Aufgaben – in der vorliegenden Studie die eingesetzten Szenarien und die zugehörigen Probleme –, der verwendete Fragebogen, die Vorgehensweise während der Studie sowie die Analyse der erfassten Daten beschrieben.

10.1.1 Ziele

In der Studie wurden Probanden nach ihrer subjektiven Einschätzung bzgl. der Arbeit mit einer digitalen und einer analogen Sketching-Umgebung im Allgemeinen und zur Arbeit mit GestureCards im Besonderen befragt. Mithilfe der erfassten Daten soll die Wahrnehmung der Probanden mit Blick auf den Einsatz von GestureCards zur Lösung praktischer Aufgaben untersucht werden.

Es wurde versucht, Probanden zu gewinnen, die über grundlegende Erfahrung mit Software-Entwicklung verfügen (vgl. Kap. 10.1.2). Eine besondere Präferenz bzgl. der Erfahrung in der Entwicklung gestenbasierter Anwendungen wurde nicht vorausgesetzt, jedoch davon ausgegangen, dass alle Probanden Erfahrung in der Bedienung Multi-Touch-basierter Anwendungen haben.

Die Probanden bearbeiteten zwei Szenarien: eines analog und eines digital (vgl. Kap. 10.1.4). Für den analogen Teil der Studie bekamen die Probanden die Möglichkeit, an klassischen Whiteboards sowie mit Stift und Papier zu arbeiten. Zur Umsetzung der digitalen Umgebung wurden GestureCards in den AugIR integriert (vgl. Kap. 10.1.3).

10.1.2 Probanden

Die teilnehmenden Probanden wurden für die Studie zufällig in zwei Gruppen eingeteilt. Zwischen beiden Gruppen gab es keine Überschneidungen, d. h. jeder Proband war Mitglied genau einer Gruppe. Beide Gruppen bearbeiteten beide in Kap. 10.1.4 beschriebenen Szenarien, starteten jedoch entweder mit der analogen oder der digitalen Variante. Diese Maßnahme hat für den weiteren Verlauf sowie die Auswertung der Studie keine Bedeutung, sondern sollte lediglich eine denkbare Beeinflussung der Probanden durch die zuerst bearbeitete Variante vermeiden.

An der Studie nahmen 14 Probanden teil; 12 waren männlich und zwei weiblich.[1] Die Probanden wurden in der Reihenfolge ihrer Teilnahme an der Studie abwechselnd auf die beiden Gruppen aufgeteilt, sodass die Zuordnung im Wesentlichen zufällig und ohne Berücksichtigung weiterer Faktoren geschah. Mit einer Ausnahme hatten alle Probanden einen Hintergrund im SE auf unterschiedlichem Niveau (vom Bachelor- bis zum Promotionsstudium; teilweise mit mehrjähriger Berufserfahrung). Einige wenige Probanden hatten bereits Erfahrung mit GestureCards, da sie an einer der vorigen Studien teilgenommen hatten, wurden jedoch ebenso wie die unerfahrenen Probanden in die Notation eingewiesen.

[1] Dass die Anzahl der Probanden und das Geschlechterverhältnis identisch zu denen der in Kap. 8 beschrieben Studie sind, ist rein zufällig; es haben nicht genau dieselben Probanden an beiden Studien teilgenommen.

Die Probanden nahmen in individuell vereinbarten Sitzungen an der Studie teil. Jede Sitzung lief in dem in Kap. 10.1.6 beschriebenem Schema ab. Während der Sitzungen war außer dem aktuellen Probanden und den Studienleitern niemand anwesend. Nach Abschluss jeder Sitzung bekamen die Probanden den in Kap. 10.1.5 beschriebenen Fragebogen ausgehändigt, den sie anonym ausfüllten. In die spätere Auswertung flossen alle Fragebögen ein.

10.1.3 Integration von GestureCards in den AugIR

Der AugIR ist ein physischer Teamraum, dessen Wände mit großen, digitalen Whiteboards ausgestattet sind. Diese verfügen über eine berührungsempfindliche Oberfläche, die mit den Fingern, speziellen Stiften und einem Wischer bedient werden kann. Der AugIR erweitert das Konzept des (analogen) Interaction Room (IR) (Book et al., 2017) um die Möglichkeiten digitalisierter Whiteboards; so können z. B. durch automatisierte Worterkennung Verknüp-

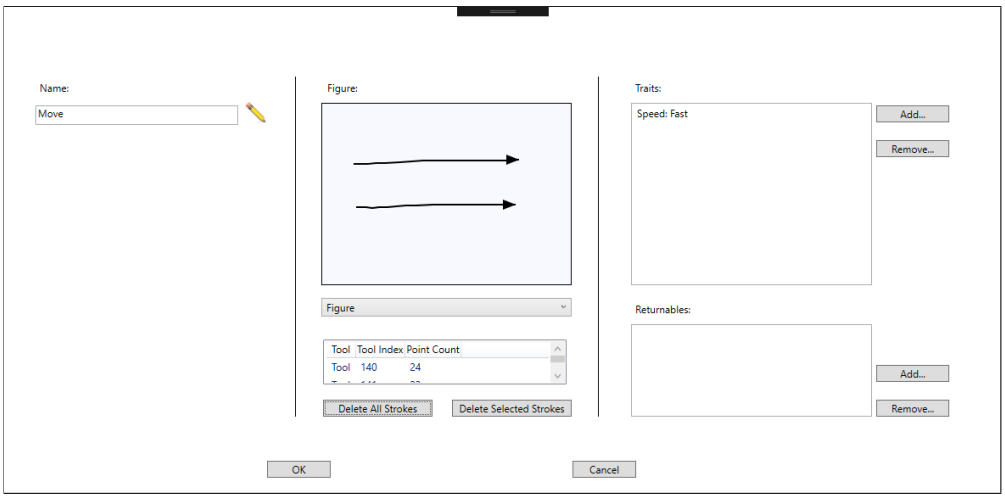


Abbildung 10.1: Dialog im AugIR zur Definition einfacher GestureCards

fungen zwischen unterschiedlichen Prozess-, UI- und sonstigen Sketches hergestellt werden (Kleffmann et al., 2015a,b, 2018). Der AugIR wird hauptsächlich in frühen Phasen der Entwicklung eingesetzt, in denen die schnelle Erstellung von Diagrammen und Sketches ein wesentlicher Bestandteil der Tätigkeiten ist. Ein besonderer Vorteil ist das schnelle Umschalten zwischen unterschiedlichen Perspektiven, die im Kontext des IR *Canvases*[2] genannt werden – so können Entwicklungsteams z. B. direkt zwischen UI-Sketches und dem zugehörigen Geschäftsprozess hin- und her wechseln (Kleffmann et al., 2015a). Durch den Einsatz mehrerer digitaler Whiteboards können auch unterschiedliche Perspektiven auf den gleichen Sachverhalt zur gleichen Zeit betrachtet werden. Im AugIR wurden bereits Möglichkeiten zur Definition von Gesten in UI-Sketches im Rahmen eigener Arbeiten getestet (Hesenius et al., 2016), die im Folgenden durch GestureCards ersetzt werden.

Für die Entwicklung gestenbasierter Anwendungen ist der *Interaction Canvas* interessant. Er erlaubt die Erarbeitung von UI-Sketches, die zu einem Storyboard verknüpft werden. Bei der Verbindung zweier UI-Sketches wurde die Möglichkeit eingefügt, eine Geste mithilfe von GestureCards zu definieren. Dazu wurde eine Reihe von Dialogen entwickelt, mit denen verschiedene Aspekte von GestureCards beschrieben werden können. AugIR-Nutzer werden so in die Lage versetzt, einfache (Abb. 10.1) und zusammengesetzte (Abb. 10.2) GestureCards sowie verschiedene Eigenschaften und Rückgaben zu definieren. Das Storyboard wurde mit den definierten GestureCards an den Transitionen annotiert; ein Beispiel ist in Abb. 10.3 zu sehen.

Darüber hinaus wurde dem AugIR die Möglichkeit hinzugefügt, definierte Storyboards in einem Simulationsmodus durchspielen zu können. Dazu wurde ein Gestenerkenner entwickelt, der mit GestureCards konfiguriert werden kann (vgl. Kap. 13.1 und Hesenius et al. (2017)). Benutzer konnten im Simulationsmodus durch Ausführung der Geste von einem Sketch zum jeweils verknüpften wechseln.

[2] Eine Auflistung der Canvases sowie eine Erläuterung ihres Einsatzzwecks ist bei Book et al. (2017) zu finden.

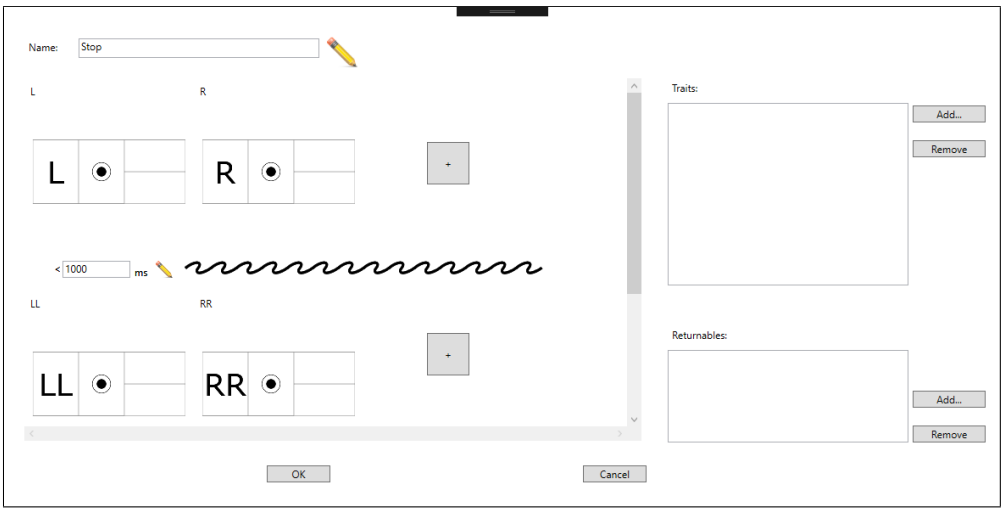


Abbildung 10.2: Dialog im AugIR zur Definition zusammengesetzter GestureCards

10.1.4 Szenarien

Jeder Proband bearbeitete zwei Szenarien, die aus je zwei Aufgaben bestanden. Beide Szenarien wichen in Details voneinander ab, jedoch ist das zugrundeliegende Schema identisch. Im Folgenden wird zunächst dieses Schema erläutert und dann auf die beiden Szenarien im Detail eingegangen. Für jedes Szenario wurde eine analoge Variante auf klassischen Whiteboards, in der die definierten GestureCards auf vorgedrucktem Papier handschriftlich eingetragen wurden, sowie eine digitale Variante im AugIR erstellt.

Die Ausgangssituation der Szenarien war, dass die Probanden gemeinsam mit einem Kollegen Anforderungen an eine gestenbasierte Anwendung erfasst und als User Stories dokumentiert hatten. Anschließend ist der Proband in den Urlaub gefahren und der Kollege hat aus den erfassten User Stories einen ersten Entwurf der Anwendung abgeleitet. Dazu hat er UI-Sketches entwickelt, in einem Storyboard zusammengefasst, die notwendigen Gesten mit GestureCards dokumentiert und diese an die Transitionen zwischen den UI-Sketches platziert. Leider haben der Proband und der Kollege sich nicht abgestimmt: Als der Proband aus dem Urlaub zurückkehrt, tritt der Kollege seinen Urlaub an, sodass keine Übergabe stattfinden konnte. Allerdings muss der Proband morgen einen Zwischenstand der Entwicklung präsentieren.

Den Probanden wurde keine Zielplattform oder Geräteklasse vorgegeben, sondern lediglich mitgeteilt, dass die Anwendung gestenbasiert sein soll. Jedes Szenario existierte außerdem in einer digitalen Variante im AugIR und einer analogen Variante auf klassischen Whiteboards. Den Probanden wurden außerdem die dokumentierten User Stories zur Verfügung gestellt; dafür wurden für die analoge Variante Papierkärtchen an ein Flip Chart gehängt und in der digitalen Version der *Feature Canvas* des AugIR verwendet.

Daraus folgt die erste Aufgabe des Probanden: Auseinandersetzung mit dem Storyboard und den definierten User Stories sowie Prüfung, ob alle definierten User Stories integriert wurden. Sollten User Stories fehlen, so mussten die Probanden sie hinzufügen. Dazu mussten die Probanden zwei Teilaufgaben lösen: notwendige UI-Sketches entwickeln und zum Storyboard hinzufügen sowie passende Gesten mit GestureCards definieren.

Nr.	User Story
1	Als Benutzer möchte ich wissen, welcher Song als nächstes gespielt wird
2	Als Benutzer möchte ich die Lautstärke regeln können
3	Als Benutzer möchte die Wiedergabe kontrollieren können (starten, stoppen, pausieren)
4	Als Benutzer möchte ich ein Lied vor-/zurückschalten können
5	Als Benutzer möchte ich Playlisten mit Freunden teilen können

Tabelle 10.1: User Stories für das Szenario MP3-Player

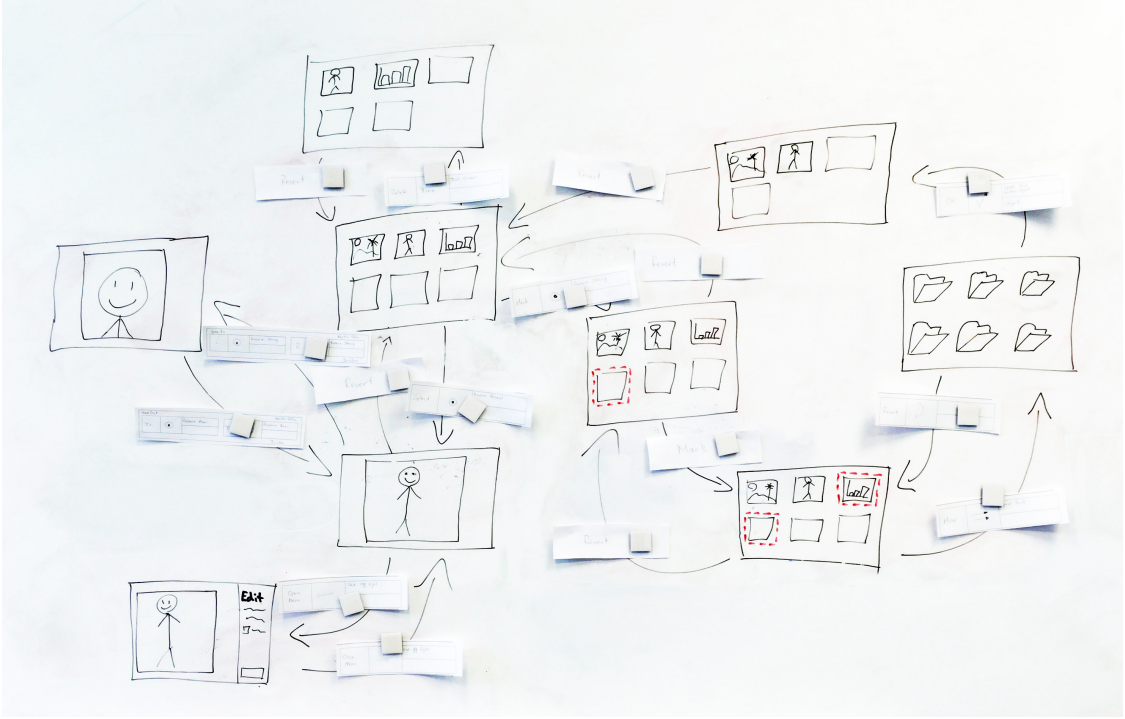


Abbildung 10.3: Analoges Storyboard für das Szenario Foto-App

[3] Die notwendigen Informationen wurden den Probanden seitens der Studienleitung mitgeteilt; die Analyse der Hardware-Spezifikation war nicht Teil der Aufgabe.

Nr.	User Story
1	Als Benutzer möchte ich Bilder löschen können
2	Als Benutzer möchte ich Aktionen rückgängig machen können
3	Als Benutzer möchte ich Bilder vergrößern/verkleinern können
4	Als Benutzer möchte ich Bilder bearbeiten können
5	Als Benutzer möchte ich (mehrere) Bilder in Ordner einsortieren können
6	Als Benutzer möchte ich Bilder mit Freunden teilen können

Tabelle 10.2: User Stories für das Szenario Foto-App

Zur Einführung der zweiten Aufgabe wurde den Probanden mitgeteilt, dass sie soeben eine E-Mail erhalten haben, in der steht, dass die angedachte Hardware leider nicht beschafft werden kann und daher auf bereits vorhandene zurückgegriffen werden muss. An die E-Mail war die Spezifikation der Hardware angehangen, aus denen die Probanden je nach Szenario unterschiedliche Informationen ableiten konnten.[3] Die Probanden waren nun angehalten, alle Gesten zu identifizieren, die nicht mit der zur Verfügung stehenden Hardware erkannt werden können und gegen passende Gesten zu ersetzen. Die vorhandenen Gesten waren bewusst so gewählt, dass die Probanden durch die offensichtliche Lösung Konflikte mit bestehenden Gesten provozierten, die sie aufdecken und beheben mussten.

Das erste Szenario für die Probanden war eine App zur Bearbeitung und Verwaltung von Fotos, im Folgenden als *Foto-App* bezeichnet. Initial wurden den Probanden die in Tab. 10.2 definierten User Stories und das in Abb. 10.3 definierte Storyboard zur Verfügung gestellt. Im Storyboard fehlte die User Story Nr. 6. Außerdem wurde den Probanden nach Erledigung der ersten Aufgabe mitgeteilt, dass die verwendete Hardware keinen Druck erkennen konnte. Entsprechend mussten sie alle Gesten, die mit der Eigenschaft `Pressure` versehen waren, identifizieren und gegen Alternativen austauschen.

Im zweiten Szenario arbeiteten die Probanden an einer gestenbasierten App zum Abspielen von Musik, im Folgenden als *MP3-Player* bezeichnet. Den Probanden wurden die in Tab. 10.1 gelisteten User Stories und das in Abb. 10.4 gezeigte Storyboard zur Verfügung gestellt. Im Storyboard fehlte die User Story Nr. 5. Darüber hinaus war im Storyboard eine Funktionalität integriert, die nicht als User Story definiert war (Vor- und Zurückspulen). Für die zweite Aufgabe wurde den Probanden mitgeteilt, dass die Hardware keine parallelen Berührungen verarbeiten konnte, also nicht Multi-Touch-fähig war.

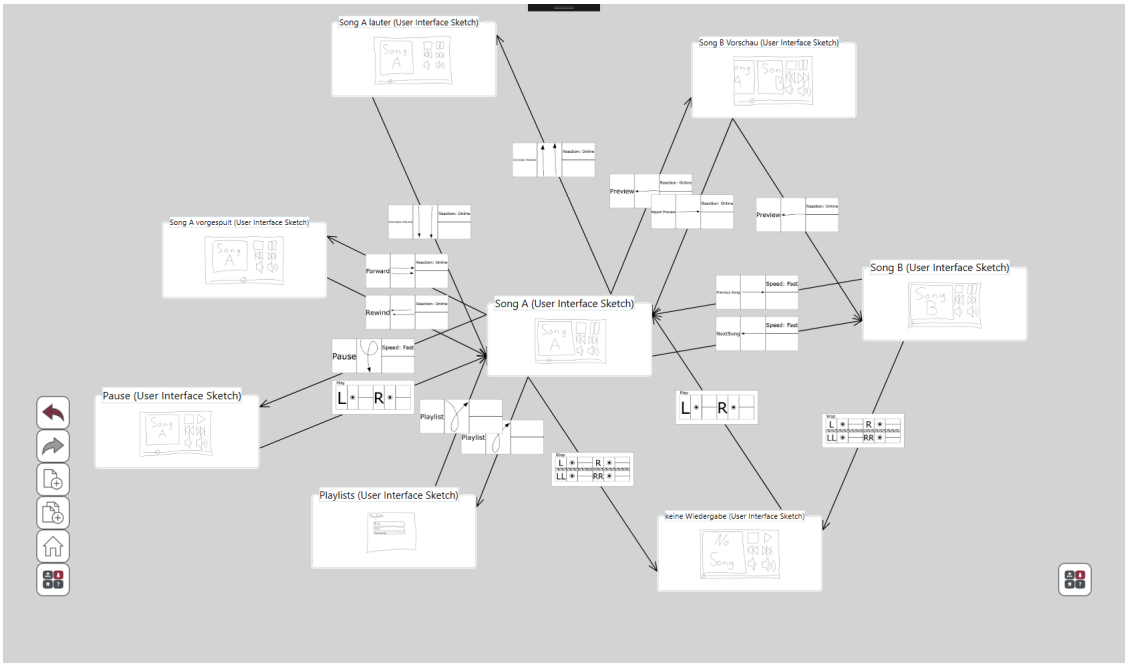


Abbildung 10.4: Digitales Storyboard im AugIR für das Szenario MP3-Player

10.1.5 Fragebogen

Der Fragebogen bestand aus 30 Fragen mit Antwortmöglichkeiten in Form von fünfstufigen Likert-Skalen und wurde von den Probanden anonym ausgefüllt. Außerdem wurde in drei Freitextfeldern nach individuellem Feedback zum AugIR und zur Integration von GestureCards gefragt. Die verschiedenen Fragen deckten unterschiedliche Aspekte von GestureCards, dem AugIR sowie der Integration von GestureCards in den AugIR ab. Im Folgenden wird auf die Fragen, die keinen Bezug zu GestureCards haben und lediglich die Verwendung des AugIRs thematisieren, nicht eingegangen. Der vollständige Fragebogen ist in Anh. D zu finden.

Zu jeder Frage wurden Likert-Skalen mit fünf Stufen definiert, auf denen die Benutzer ihre Erfahrung während der Bearbeitung der Aufgaben dokumentierten. Die Likert-Skalen stellten Wortpaare gegenüber (z. B. *Leicht* und *Schwierig*), anhand derer die Probanden einschätzen sollten, wie sie die Arbeit mit dem AugIR oder an den analogen Whiteboards empfanden. Dabei wurden die Werte 1–5 verwendet, um Abstufungen zwischen den Wortpaaren zu definieren, wobei die Werte 1 und 5 den jeweiligen Extremwerten zugeordnet wurden (im o. g. Beispiel 1 für *Leicht* und 5 für *Schwierig*). Bei den Fragen 1–28 handelt es sich bei den Antwortmöglichkeiten um gegenteilige Paare (z. B. *Schnell* und *Zeitlich aufwendig*) und bei den Fragen 29 und 30 um die Gegenüberstellung von *analog* und *digital*. Einige Fragen wurden doppelt, aber mit unterschiedlichem Fokus in den Antworten gestellt. Die Fragen 31–33 enthielten die eingangs erwähnten Freitextfelder, in denen die Probanden explizit Dinge erwähnen konnten, die ihnen gut oder gar nicht gefallen haben und welche Funktionalitäten sie sich für eine zukünftige Version wünschen.

10.1.6 Aufbau der Sitzungen

Jeder Proband nahm in einer individuellen Sitzung teil, die ohne Publikum stattfand und etwa eine Stunde dauerte. Es wurde seitens der Studienleitung versucht, dieses Zeitlimit weitgehend einzuhalten und ggf. steuernd in den Ablauf einzugreifen. Die Probanden starteten immer mit dem Szenario Foto-App, jedoch abwechselnd in der digitalen und analogen Variante. Der grundlegende Ablauf einer Sitzung war dabei immer identisch:

- Einführung in den Forschungskontext durch den Studienleiter
- Einführung in GestureCards durch den Studienleiter
- Einführung in das Szenario Foto-App
- Erläuterung und Durchführung der ersten Aufgabe
- Erläuterung und Durchführung der zweiten Aufgabe
- Einführung in das Szenario MP3-Player
- Erläuterung und Durchführung der ersten Aufgabe
- Erläuterung und Durchführung der zweiten Aufgabe

Der AugIR wurde jeweils vor der Verwendung in einem Szenario anhand eines Beispiels erläutert. Im Falle, dass der Proband mit der digitalen Variante startete, wurde der AugIR direkt nach der Einführung von GestureCards eingeführt. Startete der Proband mit der analogen Variante, dann wurde der AugIR erst nach dem ersten Szenario eingeführt.

Zunächst wurden den Probanden die grundlegenden Konzepte von GestureCards mithilfe einer kurzen Präsentation nahegebracht, in der die Struktur sowie einige Beispiele vermittelt wurden. Die vollständige Präsentation ist in Anh. D zu finden. Die Probanden wurden im Rahmen der Einführung darauf hingewiesen, dass GestureCards zahlreiche Eigenschaften unterstützen, die sie ggf. bei der Studienleitung erfragen konnten.

Der AugIR wurde mithilfe eines kurzen Beispiels eingeführt. Dazu wurde vom Studienleiter in einem Demo-Projekt ein neuer UI-Sketch angelegt, auf dem beispielhaft drei Checkboxen mit angedeutetem Beschreibungstext gezeichnet wurden. Am Beschreibungstext wurde gezeigt, wie im AugIR UI-Sketches manipuliert werden können (Markieren, Verschieben und Löschen des Textes sowie das Rückgängigmachen der Änderungen). Anschließend wurde der UI-Sketch geklont und hier zwei der Checkboxen markiert. Dann wurde exemplarisch eine GestureCard für eine Geste angelegt, mit der zwei der Checkboxen durch eine Linie nach unten, die mit der Eigenschaft `Tool` auf ein Werkzeug beschränkt wurde, aktiviert werden konnten. Mit dieser Geste wurde auch der Simulationsmodus des AugIR demonstriert, bei dem das Storyboard interaktiv bedient werden kann. Anschließend wurde demonstriert, wie bereits definierte Gesten wiederverwendet werden und im Storyboard navigiert wird (Herein- und Herauszoomen sowie Verschieben).

Die Szenarien wurden wie in Kap. 10.1.4 beschrieben eingeführt. Für die Arbeit an den analogen Storyboards wurden den Probanden die notwendigen Hilfsmittel in Form von Stiften und Vordrucken für GestureCards zur Verfügung gestellt. Die Probanden konnten außerdem zu jeder Zeit Fragen zu allen Aspekten stellen und wurden aufgefordert, ihre Gedanken laut zu formulieren. Jede Sitzung wurde – das Einverständnis des Probanden vorausgesetzt – auf Video aufgenommen. Nach Durchführung beider Szenarien wurde den Probanden der Fragebogen ausgehändigt, den sie eigenständig ausfüllten.

10.1.7 Analyse

Zur Auswertung wurde eine deskriptive Analyse der Fragebögen durchgeführt und alle Antworten zu einem Gesamtbild aggregiert. Dabei wurde lediglich auf die Fragen eingegangen, die einen direkten Bezug zu GestureCards haben. Fragen, die die inhaltliche Ausgestaltung des AugIR (z. B. zur Erstellung von UI-Sketches) zum Inhalt haben, wurden nicht berücksichtigt.

Die Antworten aller Probanden wurden aggregiert und in die Kategorien *Verständnis*, *Definition* und *Gesamteindruck* unterteilt, um einen Überblick über ihre Einschätzung der Arbeit mit GestureCards und dem AugIR zu gewinnen. Außerdem wurden die Hinweise der Probanden in den Freitextfeldern sowie ihre Kommentare nach Ähnlichkeit und Thema gruppiert.

10.2 Ergebnisse

Im Folgenden werden die Ergebnisse der Studie dargelegt. Dazu werden zunächst die aggregierten Ergebnisse jeder Kategorie vorgestellt. Anschließend wird gezielt auf Beobachtungen und Kommentare der Probanden während der Sitzungen und im Fragebogen eingegangen.

10.2.1 Verständnis

Die Fragen 1–7 beschäftigten sich allgemein mit dem Verständnis der vordefinierten Gesten in den verwendeten Storyboards, sowohl analog auf klassischen Whiteboards als auch digital im AugIR. Es wird deutlich, dass die Probanden die analoge Definition von GestureCards als intuitiv (Frage 1; $Mdn = 2$) betrachteten und den zeitlichen Aufwand als eher gering einschätzten (Frage 2; $Mdn = 2$). Allerdings schätzten sie die digitale Variante im AugIR als intuitiver (Frage 3; $Mdn = 1$) und zeitlich weniger aufwendig (Frage 4; $Mdn = 1$) ein. Das gilt auch für den direkten Vergleich, bei dem die Probanden den AugIR sowohl als intuitiver (Frage 5; $Mdn = 2$) als auch zeitlich weniger aufwendig (Frage 6; $Mdn = 2$) bewerteten. Außerdem fühlten sich die Probanden vom AugIR beim Verständnis der mit GestureCards vordefinierten Gesten unterstützt (Frage 7; $Mdn = 2$).

Die im AugIR eingesetzten Erweiterungen wurden von den Probanden positiv aufgenommen. Der Simulationsmodus wurde ebenso als hilfreich empfunden (Frage 23; $Mdn = 1$) wie die Animation der graphischen Repräsen-

Das Verstehen vordefinierter Gesten auf analogen Whiteboards war ...					
Frage	1	2	3	4	5
1	Intuitiv ... Nicht intuitiv				
	4	7	2	1	–
2	Schnell ... Zeitlich aufwendig				
	4	7	2	1	–

Das Verstehen vordefinierter Gesten im AugIR war ...					
Frage	1	2	3	4	5
3	Intuitiv ... Nicht intuitiv				
	8	4	2	–	–
4	Schnell ... Zeitlich aufwendig				
	9	3	2	–	–

Im Vergleich zu analogen Whiteboards war das Verstehen vordefinierter Gesten im AugIR ...					
Frage	1	2	3	4	5
5	Intuitiver ... Weniger intuitiv				
	5	5	4	–	–
6	Schneller ... Zeitlich aufwendiger				
	4	4	6	–	–

Im Vergleich zu analogen Whiteboards hat der AugIR mich beim Verstehen vordefinierter Gesten ...					
Frage	1	2	3	4	5
7	Unterstützt ... Behindert				
	5	5	4	–	–

Für das Verstehen der Gesten war der Simulationsmodus im AugIR ...					
Frage	1	2	3	4	5
23	Hilfreich ... Nicht hilfreich				
	10	2	2	1	–

Für das Verstehen der Gesten war die Animation der Gestenbewegung im AugIR ...					
Frage	1	2	3	4	5
24	Hilfreich ... Nicht hilfreich				
	10	3	1	–	–

Tabelle 10.3: Aggregierte Ergebnisse der Fragen 1–7, 23 und 24

tation (Frage 24; $Mdn = 1$). Diese Bewertung spiegelt sich auch in den Kommentaren der Probanden wider: Sechs Probanden erwähnten explizit, dass ihnen die Animationen gut gefallen haben, und drei lobten den Simulationsmodus, in dem Gesten direkt ausprobiert werden können. Die Ergebnisse aller Fragen der Kategorie Verständnis sind in Tab. 10.3 gelistet.

10.2.2 Definition

In den Fragen 8–14 beschrieben die Probanden ihren Eindruck von der Definition von Gesten. Tab. 10.4 und Tab. 10.5 geben einen Überblick: Zwischen der analogen Definition von GestureCards auf (vorgedrucktem) Papier und der digitalen im AugIR gibt es nur wenige Unterschiede. So empfanden die meisten Probanden die Definition neuer Gesten sowohl im Analogen als auch im Digitalen als eher leicht (Fragen 8 und 10; jeweils $Mdn = 2$). Lediglich ein Proband empfand die analogen GestureCards als eher schwierig. Auch beim zeitlichen Aufwand sind nur marginale Unterschiede zwischen beiden Varianten sichtbar (Fragen 9 und 11; jeweils $Mdn = 2$).

Beim direkten Vergleich jedoch empfanden einige Probanden scheinbare Widersprüche. Während die meisten Probanden die Definition neuer Gesten im AugIR als leichter oder mindestens genauso leicht wie auf analogen Whiteboards empfanden (Frage 12; $Mdn = 2$), nivelliert sich der Eindruck bei Betrachtung des gefühlten zeitlichen Aufwands (Frage 13; $Mdn = 2,5$). Drei Probanden empfanden die Definition von Gesten im AugIR als zeitlich eher aufwendiger und zwei gar als zeitlich aufwendiger. Allerdings fühlten sich die meisten Probanden durch den AugIR bei der Definition neuer Gesten eher unterstützt (Frage 14; $Mdn = 2$).

Im Vergleich zu analogen Whiteboards hat der AugIR mich bei der Definition neuer Gesten ...					
Frage	1	2	3	4	5
14	Unterstützt ... Behindert				
	3	7	2	2	–

Tabelle 10.4: Ergebnisse der Frage 14.

Tabelle 10.5: Aggregierte Ergebnisse der Fragen 8–13

Die Definition neuer Gesten auf analogen Whiteboards war ...					
Frage	1	2	3	4	5
8	Leicht ... Schwierig				
	6	4	3	1	–
9	Schnell ... Zeitlich aufwendig				
	4	5	2	3	–

Die Definition neuer Gesten im AugIR war ...					
Frage	1	2	3	4	5
10	Leicht ... Schwierig				
	5	4	5	–	–
11	Schnell ... Zeitlich aufwendig				
	4	6	3	1	–

Im Vergleich zu analogen Whiteboards war die Definition neuer Gesten im AugIR ...					
Frage	1	2	3	4	5
12	Leichter ... Schwieriger				
	4	4	2	4	–
13	Schneller ... Zeitlich aufwendiger				
	4	3	2	3	2

Insgesamt halte ich GestureCards zur Beschreibung von Gesten für ...					
Frage	1	2	3	4	5
27	Nützlich ... Nicht nützlich				
	9	5	–	–	–

Insgesamt würde ich Folgendes präferieren ...					
Frage	1	2	3	4	5
29	Analoge GCs ... Digitale GCs				
	1	2	–	7	4

Insgesamt würde ich Folgendes präferieren ...					
Frage	1	2	3	4	5
30	Analoge Whiteboards ... AugIR				
	1	3	–	6	4

Tabelle 10.6: Aggregierte Ergebnisse der Fragen 27, 29 und 30

10.2.3 Gesamteindruck

Der Gesamteindruck der Probanden, zu sehen in Tab. 10.6, zeigt, dass sie überwiegend den Einsatz von GestureCards für die gestellten Aufgaben in beiden Szenarien als nützlich empfanden (Frage 27; $Mdn = 1$). Die Antworten auf die Fragen nach der persönlichen Präferenz zwischen der Arbeit auf analogen Whiteboards und im digitalen AugIR sind etwas polarisierter. Zwar bevorzugen die meisten Probanden eher sowohl digitale GestureCards (Frage 29; $Mdn = 4$) als auch den AugIR (Frage 30; $Mdn = 4$), jedoch spricht sich eine Minderheit deutlich für die analogen Varianten aus. Allerdings zeigt ein Blick in die einzelnen Ergebnisse, dass diese Trennung nicht pauschal für alle Probanden gilt. Zwar gaben die meisten Probanden (elf an der Zahl) bei den Fragen 29 und 30 dieselbe Bewertung an, drei Probanden jedoch wichen in Details ab. So bevorzugte ein Proband analoge GestureCards, bewertete aber den AugIR besser als analoge Whiteboards. Zwei Probanden bevorzugten zwar die Arbeit mit digitalen GestureCards gegenüber analogen, arbeiteten jedoch lieber auf analogen Whiteboards als im AugIR.

10.2.4 Beobachtungen

Neben den Ergebnissen des Fragebogens, der die subjektive Einschätzung der Probanden enthält, ergeben sich einige Erkenntnisse aus der Beobachtung der Probanden bei der Arbeit sowie durch ihre Gedanken und Kommentare, die sie im Laufe einer Sitzung formulierten. Die Probanden waren alle in der Lage, die gestellten Aufgaben weitgehend zu lösen, auch wenn sich einige länger an manchen Details aufhielten als andere, sodass seitens der Studienleitung eingegriffen werden musste. So konnten alle Probanden in beiden Szenarien größtenteils problemlos die fraglichen Gesten identifizieren, die aufgrund der beschriebenen Hardware-Beschränkungen nicht verwendet werden können. Außerdem konnten alle Probanden die definierten Storyboards um eine neue Funktionalität erweitern und passende Gesten dazu definieren. Sie identifizierten außerdem erfolgreich potentielle Probleme durch neue Gesten, wenn diese zu ähnlich zu bestehenden waren.

Bei der Erstellung und Änderung von Gesten legten die Probanden unterschiedliche Strategien an den Tag: Manche orientierten sich eher an bekannten Gesten aus gängigen Frameworks (vgl. Kap. 4.2) und ignorierten die bereits definierten Gesten, während andere versuchten, die neuen Gesten an die bestehenden anzupassen oder zumindest eine gewisse Ähnlichkeit zu gewährleisten. Interessant ist dabei, dass die meisten Probanden bestrebt waren, eher neue Gesten zu definieren, anstatt bestehende wiederzuverwenden. Ein Proband ging sogar so weit, bereits bestehende Gesten im Nachhinein zu verändern, weil er sie nicht für geeignet hielt.

Obwohl den Probanden keine Vorgaben gemacht wurden, wie sie die Gesten zu definieren hatten, wiederholen sich einige Lösungen. So tauschten mehrere Probanden die Eigenschaft `Pressure` im Szenario Foto-App gegen die Eigenschaft `Duration` aus. Einige Probanden erkannten auch, dass dies bei zwei Gesten zu einem Konflikt durch die Eigenschaft `Reaction: Online` führen würde, und passten die Gesten entsprechend an. Manche Probanden allerdings ignorierten diese Eigenschaft und konnten diese Feinheit nicht aufdecken. Im Szenario MP3-Player entfernten die meisten Probanden zunächst einen Gestenzug aus den Multi-Touch-Gesten, um die Aufgabe zu lösen, und identifizierten auch die daraus folgenden Konflikte in den Gesten größtenteils direkt im Anschluss.

Mit allen anderen Eigenschaften konnten die Probanden weitgehend problemlos arbeiten, obwohl sie kein Nachschlagewerk zur Verfügung hatten. Außerdem definierten einige Probanden auch passende Rückgaben, um die für die Anwendung wichtigen Informationen der Interaktion festzulegen. Einige Probanden erkannten auch, dass manche ihrer definierten GestureCards Schwächen hatten und fragten explizit nach, wie sie sie lösen könnten. Eine wiederkehrende Frage war, wie die Richtung der Ausführung aufgehoben werden kann, ohne die Form zu verlieren (die Lösung in GestureCards ist, die Eigenschaft `Orientation: None` zu definieren).

Auffällig ist, dass die Probanden sowohl auf analogen Whiteboards als auch im digitalen AugIR die Aufgaben erfolgreich lösen konnten. Allerdings kamen einige Probanden auf analogen Whiteboards in die Situation, dass Teile von GestureCards durch andere GestureCards überdeckt wurden, sodass vielleicht wichtige Informationen nicht auf Anhieb erkennbar waren. Einige Probanden mussten deshalb einmal getroffene Entscheidungen revidieren und sich eine andere Lösung überlegen.

Mehrere Probanden lobten explizit die Animation der Gesten im AugIR sowie den Simulationsmodus und die Möglichkeit, definierte Gesten direkt im Kontext der Anwendung auszuprobieren. Drei Probanden hoben außerdem hervor, dass ihnen die Struktur von GestureCards – also die Trennung von graphischer Präsentation und textuell definierten Eigenschaften – gut gefiel. Dem steht gegenüber, dass mehrere Probanden die Menustruktur bei der Erstellung von GestureCards im AugIR kritisierten und sich hier eine Vereinfachung wünschten. Außerdem kritisierten mehrere Probanden, dass die Lesbarkeit auf der verwendeten Hardware im AugIR nicht optimal war. Ein Proband erwähnte explizit, dass die Erstellung der graphischen Repräsentation auf den analogen GestureCards einfacher war als im AugIR. Ein weiterer kritisierte, dass er im AugIR eher den Überblick über das Storyboard verliere.

10.3 Diskussion

Die Ergebnisse zeigen, dass die Probanden GestureCards als Bereicherung und nützliche Unterstützung während ihrer Arbeit mit den Aufgaben der Studie empfanden. Somit haben GestureCards nicht nur messbare (vgl. Kap. 8) und subjektive (vgl. Kap. 9) Vorteile gegenüber anderen Notationen, sondern werden auch im praktischen Einsatz als hilfreich wahrgenommen und

unterstützen somit bei der Entwicklung gestenbasierter Anwendungen. Das gilt sowohl für den analogen Einsatz auf klassischen Whiteboards mithilfe von Stift und Papier, als auch für den Einsatz auf digitalen Whiteboards im AugIR. Somit ist die eingangs gestellte erste Forschungsfrage dieser Studie zu bejahen: GestureCards unterstützen bei der Entwicklung gestenbasierter Anwendungen. Außerdem zeigen die Ergebnisse, dass auch die zweite eingangs gestellte Forschungsfrage zu bejahen ist: GestureCards können durch Digitalisierung aufgewertet werden.

Insbesondere der Einsatz animierter GestureCards im AugIR fand großen Zuspruch unter den Probanden – eine Funktion, die nur mit technischen Mitteln umgesetzt werden kann und auf analogen Whiteboards nicht zur Verfügung steht. Hier ist mutmaßlich noch mehr Potential zur Optimierung vorhanden, das der AugIR durch Automatisierung verschiedener Vorgänge bietet. So könnten GestureCards auf ähnliche Interaktionsmuster geprüft (vgl. Kap. 13.2) und Entwickler bei der Identifikation schwieriger oder zu ähnlicher Gesten unterstützt werden.

Auch die Möglichkeit, definierte Gesten im Simulationsmodus des AugIR direkt auszuprobieren, wurde von den Probanden weitgehend positiv aufgenommen. An dieser Stelle besteht aber auch die Gefahr, durch technische Einschränkungen den kreativen Fluss bei der Erstellung von UI-Sketches zu unterbrechen oder die Entwicklung der Interaktion an den technischen Möglichkeiten der Anwendung zum Sketching auszurichten. Außerdem entsteht quasi ein Henne-Ei-Problem: Die definierten Gesten sind Grundlage für die technischen Entscheidungen (vgl. Kap. 4.3), die aber zur Ausführung des Simulationsmodus bereits getroffen bzw. implementiert sein müssten, außer der AugIR deckt alle denkbaren Möglichkeiten ab, woraus wiederum folgt, dass die technischen Entscheidungen gar nicht notwendig wären. Eine denkbare Alternative wäre, dass UI-Designer und Software-Ingenieure an dieser Stelle enger zusammenarbeiten und der AugIR durch ein Plug-in-System um die technischen Komponenten erweitert werden kann, die zur Erkennung der definierten Gesten notwendig sind. Die erforderliche technische Grundlage ist im verwendeten Erkenner bereits vorhanden (vgl. Kap. 13.1). Allerdings droht hier die Grenze zwischen Sketching und Implementierung zu verschwimmen und es muss bezweifelt werden, ob das zielführend ist; schließlich ist eines der Merkmale von Sketching, dass man eben nicht durch technische Fragestellungen in seiner Arbeit gestört wird.

Auffällig ist, dass die Eigenschaft `Reaction` und insbesondere ihre Auswirkungen auf die Anwendung von einigen Probanden nicht beachtet wurde. Zur Entwurfszeit der Anwendung ist hier eine gewisse Vorstellungskraft notwendig, da in den UI-Sketches weder auf analogen Whiteboards noch im AugIR die notwendige Logik zur Animation vorhanden ist. Bei analogen UI-Sketches könnten einige Anwendungsfälle durch den Einsatz von Büromaterial (vgl. Kap. 3.2.1.2) umgesetzt werden, jedoch ist auch hier eine Kreativität gefragt, die mutmaßlich nur mit einer gewissen Erfahrung erlangt werden kann. Im AugIR könnten hierfür die Definition von Schlüsselszenen und die automatische Berechnung des Zustands zwischen den Schlüsselszenen hilfreich sein (vgl. Kap. 3.3), obwohl vermutlich irgendwann eine technische

Grenze erreicht wird, an der zu viel Dynamik verlangt wird, die nur noch mit expliziter Programmierung gelöst werden kann. Spätestens an diesem Punkt taucht aber wieder das Problem auf, dass die Grenze zwischen dem Entwurf und der Implementierung verschwimmt.

10.4 Risiken für die Validität

Wie die vorigen Studien sind auch die Ergebnisse dieser Studie Risiken ausgesetzt, die die Validität gefährden und im Folgenden diskutiert werden. So waren die Probanden vorwiegend männlich und ein potentieller Einfluss dieser Tatsache kann nicht ausgeschlossen werden. Die Probanden hatten außerdem größtenteils weiterführende Kenntnisse der Programmierung. Eine Fähigkeit von Programmierern ist die Abstraktion und Aufteilung von Konzepten. Es ist nicht auszuschließen, dass diese Fähigkeit Einfluss auf die Art und Weise hat, wie die Probanden mit Gestenspezifikationen arbeiten.

Weitere Risiken sind die vorgegebenen Szenarien und eingesetzten Gesten. Zwar wurde darauf geachtet, auch eher ungewöhnliche Gesten in den vorgefertigten Szenarien einzusetzen, jedoch ist der Einfluss typischer Gesten aus bekannten Frameworks unklar. Das gilt auch für den Einfluss, den eventuell bekannte Anwendungen, die die gleichen Funktionen aufweisen wie die beiden in den Szenarien beschriebenen, auf die Arbeit der Probanden haben. Allerdings hätte der Einsatz einer unbekannten und auf einen spezielleren Anwendungsfall abgestimmte Anwendung eine tiefere Auseinandersetzung der Probanden mit dem Anwendungskontext notwendig gemacht und mutmaßlich vom eigentlichen Zweck der Studie abgelenkt.

Ein weiterer Faktor, dessen Einfluss ungewiss ist, ist die verwendete Hardware für den AugIR. Einige Probanden gingen explizit darauf ein, dass die Auflösung der digitalen Whiteboards zu gering gewesen sei; andere ließen diesen Aspekt unkommentiert. Es ist nicht klar, ob sie dieser Umstand nicht weiter störte bzw. sie sich damit zufrieden gaben, ständig die Ansicht vergrößern zu müssen, oder ob bessere Hardware ihre Bewertung beeinflusst hätte.

Teil IV

Vorschlag zur graphischen Repräsentation räumlicher Gesten

11 Einsatz von Handmodellen und animierten Pfeilen

GestureCards können so, wie sie in Kap. 7 definiert wurden, sowohl zur Beschreibung von Oberflächengesten als auch von räumlichen Gesten eingesetzt werden. Allerdings ist die Menge der räumlichen Gesten, die so beschrieben werden können, begrenzt. Dafür gibt es zwei Gründe:

- Räumliche Gesten erlauben Bewegung im dreidimensionalen Raum, die in einer zweidimensionalen Notation nur schwer dargestellt werden kann. In Kap. 7 wurde die dritte Dimension insbesondere bei der Definition der Form in der graphischen Repräsentation nicht berücksichtigt.

- Räumliche Gesten erlauben feingliedrige Posen, bei denen es auf die Positionierung z. B. einzelner Finger im Raum ankommt. In Kap. 7 wird auf Posen als ausschließlich bewegungslose Aktionen eingegangen, jedoch nicht beschrieben, wie die unterschiedlichen Teile einer Pose in ein Verhältnis zueinander gesetzt werden können.

Die Einführung einer dritten Dimension äußert sich in erster Linie in der graphischen Repräsentation von Gesten, aber auch verschiedene Eigenschaften sind betroffen und können weitere Werte annehmen. In Kap. 7 wurde diesbezüglich schon auf einige eingegangen. Für die graphische Repräsentation sind unterschiedliche Optionen denkbar und einige verwandte Arbeiten (vgl. Kap. 6 und insbesondere Kap. 6.5.2) zeigen auch die Vor- und Nachteile verschiedener Varianten. Zu beachten ist aber vor allem der kommunikative Aspekt während der Arbeit in einem interdisziplinären Projektteam; die von vielen verwandten Arbeiten angestrebte Exaktheit steht hier u. U. im Widerspruch.

Moderne Sensoren ermöglichen die exakte Bestimmung einzelner Körperteile. Selbst feingliedrige Elemente wie einzelne Finger und ihre Gelenke können einwandfrei identifiziert werden. Gleichzeitig ist die menschliche Hand ein sehr exaktes Werkzeug, dass eine Fülle unterschiedlicher Möglichkeiten bietet, die auch für Gestensteuerung eingesetzt bzw. für Posen verwendet werden können. Die Beschreibung von Posen erfordert z. T. andere Information als die Beschreibung von Bewegung: Neben den wesentlichen Körperteilen müssen u. U. andere Körperteile bzw. ihre Relation zueinander beschrieben werden. So müssen zur exakten Definition einer Pose mit der Hand sowohl die abgespreizten als auch die eingeklappten Finger definiert werden. Die Fülle an Informationen, die hier notwendig ist, macht selbst einfache Notationen schnell unübersichtlich (vgl. insbesondere Kap. 6.2.6).

GestureCards setzen zur Darstellung der Form einer Geste auf eine graphische Repräsentation, die mithilfe von Pfeilen auszuführende Bewegungen dokumentiert. Für Oberflächengesten ist der gewählte Lösungsansatz einfach

einzusetzen, denn eine Geste auf einer Oberfläche entspricht im Wesentlichen der Darstellung in GestureCards: Oberflächen sind ebenso wie GestureCards zweidimensional. Die Darstellung von räumlichen Gesten hingegen verlangt die Darstellung einer dritten Dimension, die dann zwangsläufig auf Einschränkungen durch das gewählte Medium trifft. Das zugrundeliegende Problem betrifft nicht nur die Spezifikation von Gestensteuerung, sondern findet sich in jeder Notation, die versucht, dreidimensionale Bewegungen auf einem zweidimensionalen Medium darzustellen und kann unterschiedlich gelöst werden (vgl. Kap. 6.5.2).

Im Folgenden werden zwei Erweiterungen für GestureCards vorgeschlagen, mit denen räumliche Gesten und Posen graphisch dargestellt werden können. Der Fokus wird dabei weiterhin auf den Händen und den unterschiedlichen Möglichkeiten liegen, auch wenn heute bereits Sensoren existieren, mit der der ganze Körper aufgenommen werden kann. Dazu wird diskutiert und veranschaulicht, wie räumliche Posen (Kap. 11.1) und Gesten (Kap. 11.2) in GestureCards definiert werden können.

11.1 Darstellung von Posen

Eine Pose ist nach der Definition in Kap. 4.1.2 eine statische Positionierung von Werkzeugen im Raum. Ein wesentlicher Aspekt dabei ist aber ihre Haltung und Beziehung zueinander. Zur Beschreibung einer Pose müssen also unterschiedliche Informationen definiert und in Bezug zueinander gesetzt werden können. Eine Möglichkeit ist z. B. die Definition einzelner Finger, die dann über die unterschiedlichen Eigenschaften zueinander in Relation gesetzt werden. Es ist aber davon auszugehen, dass eine Vielzahl von Informationen dazu beschrieben werden muss und GestureCards für Posen dadurch unnötig kompliziert werden. Eine einfachere Variante ist, die graphische Repräsentation durch Abbildungen der Hände zu ersetzen.

Eine erste, einfache Variante kann mit Fotos der Hand umgesetzt werden, die die definierte Pose zeigt. Dieses Vorgehen hat verschiedene Vor- und Nachteile. Ein wesentlicher Vorteil ist, dass die Pose in den meisten Fällen gut erkennbar ist, denn das Foto zeigt alle Teile der Hand. Ein weiterer Vorteil ist, dass ein Foto einer Geste schnell erstellt ist, denn aufgrund der Verbreitung von Smartphones hat man eine Kamera schnell zur Hand. Allerdings ist die Integration eines solchen Bildes in GestureCards mit unverhältnismäßig viel Aufwand verbunden. Zunächst muss das Foto aus der passenden Perspektive aufgenommen werden, wofür u. U. bereits eine gewissen Geschicklichkeit vonnöten ist. Anschließend muss der Hintergrund entfernt werden, denn die einzig notwendige Information ist die abgebildete Hand. Moderne Software zur Bildbearbeitung kann hier unterstützen. Letztlich muss das Bild der Hand noch passend skaliert werden. Abb. 11.1 zeigt ein Beispiel.

Eine Alternative dazu, die in ähnlicher Form auch oftmals zur Darstellung von Händen im Kontext HCI eingesetzt wird, ist, lediglich die Silhouette zu verwenden. Dazu kann wie oben beschrieben vorgegangen werden, nur dass die Abbildung der Hand zusätzlich noch geschwärzt wird. Die Silhouette hat den Vorteil, dass sie von von der aufgenommenen Hand weitgehend abstra-

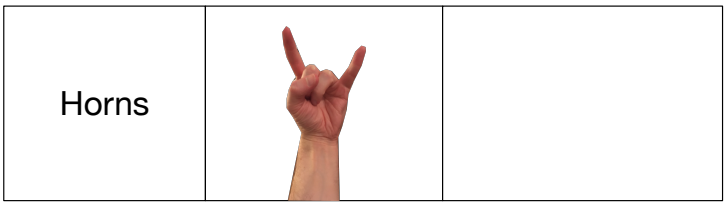


Abbildung 11.1: GestureCard einer Pose mit einem Foto als graphische Repräsentation

hiert und im Nachhinein auch leichter manipuliert werden kann, ohne dass die Manipulation sofort auffällt. Ein weiterer Vorteil ist, dass sich keiner der Mitarbeiter im Team als (u. U. identifizierbares) Mannequin zur Verfügung stellen muss, sondern wiedererkennbare Strukturen verfremdet werden. Allerdings hat die Verwendung einer Silhouette den Nachteil, dass sie lediglich die Konturen zeigt und somit zwar die Form der Pose betont, jedoch u. U. wichtige Details der Haltung verbirgt. Abb. 11.2 zeigt, wie das Beispiel aus Abb. 11.1 mit einer Silhouette dargestellt wird. Zwar bleibt die Grundform der Pose – die abgespreizten Finger – erhalten, jedoch gehen Informationen über die eingeknickten Finger verloren. Außerdem ist unklar, ob die rechte oder linke Hand von vorne oder von hinten betrachtet wird.

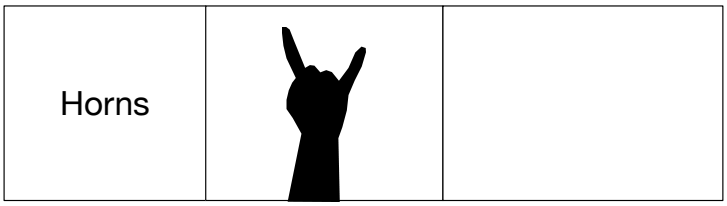


Abbildung 11.2: GestureCard einer Pose mit einer Silhouette als graphische Repräsentation

Eine dritte Option ist der Einsatz eines Handmodells, das die menschliche Hand nachbildet. Der Vorteil ist, dass für die Interaktion unwesentliche Details (z. B. unsichtbare Knochen der Mittelhand) abstrahiert werden können, wodurch die Darstellung auf das notwendige Minimum reduziert wird. Außerdem wird erneut der Mitarbeiter, dessen Hand für die Darstellung fotografiert wird, unkenntlich gemacht. Abb. 11.3 zeigt ein Beispiel. Die blauen Kugeln repräsentieren die Gelenke der Hand und der Finger. Die Verbindungen dazwischen stellen die Knochen dar. Alle weiteren Informationen (z. B. Fingernägel) sind für die Definition der Pose unwesentlich und daher entfernt worden. Die Auswahl eines passenden Handmodells ist natürlich geprägt von unterschiedlichen Geschmäckern, Vorlieben und dem jeweiligen Einsatzzweck. Alternativen sind somit denkbar – insbesondere, wenn nicht nur die Hand, sondern z. B. ganze Körperposen eingesetzt werden sollen.

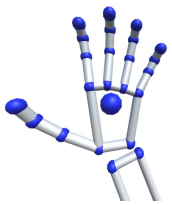


Abbildung 11.3: Das im Folgenden verwendete Modell der Hand (dem SDK des Leap Motion Controllers (Leap Motion Developer, oJ[e]) entnommen; Bildquelle: Darstellung aus eigener Software)

Das Handmodell kann in der graphischen Repräsentation eingesetzt werden, um Posen zu definieren. Während o. g. Schritte immer mit begleitendem Aufwand verbunden sind, lässt sich ein derartiges Handmodell mit technischer Unterstützung recht einfach generieren. Zahlreiche Beispiele finden sich z. B. im SDK des Leap Motion Controllers, das verschiedene vorgefertigte Handmodelle mit variierendem Detailgrad zur Verfügung stellt (Leap Motion Developer, oJ[e]) oder auch die Möglichkeit bietet, eigene Modelle zu definieren (Leap Motion Developer, oJ[a]). Im Folgenden wird für die Beispiele auf das Handmodell *Capsule* des Leap Motion SDKs zurückgegriffen (Leap Motion Developer, oJ[e]), das in Abb. 11.3 dargestellt ist. Denkbar ist aber

auch der Einsatz eines anderen Handmodells, wenn unterschiedliche Merkmale betont werden sollen oder das Team keine derart abstrakte Darstellung wünscht; z. B. wurde im Rahmen der Studie, die in Kap. 12 beschrieben wird, auf das Modell *Rigged Hands* (Leap Motion Developer, oJ[e]) zurückgegriffen. Mit Unterstützung durch passende Sensoren und Software kann das Handmodell auch generiert werden, indem die Hand aufgenommen und entsprechend übersetzt wird. Eine Bearbeitung wie im ersten Beispiel ist somit nicht mehr nötig und das Handmodell kann direkt in die GestureCard eingesetzt werden. Abb. 11.4 zeigt die Geste aus Abb. 11.1 als Handmodell.

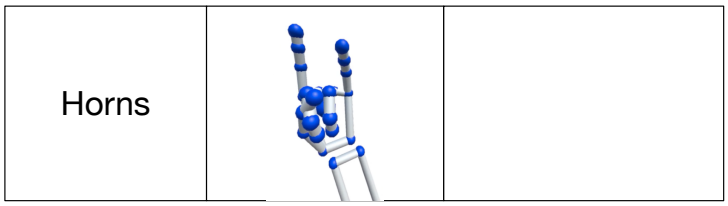


Abbildung 11.4: Eine Pose in GestureCards dargestellt mit einem Handmodell

Posen können genauso wie Gesten auch mit zusätzlichen Eigenschaften verfeinert werden. Dazu kann auf fast alle der in Kap. 7 beschriebenen Eigenschaften zurückgegriffen werden, da sie analog zu Gesten in einer Pose verwendet werden können. Abb. 11.5 zeigt, wie eine Pose um die Eigenschaft `Duration: Long` erweitert wird, um deutlich zu machen, dass die Pose über einen längeren Zeitraum eingenommen und gehalten werden muss. Somit entspricht die Verwendung also im Wesentlichen der in Kap. 7.2.1.6 definierten Bedeutung.

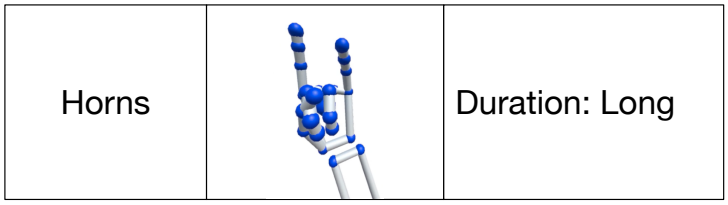


Abbildung 11.5: Eine Pose in GestureCards mit einer zusätzlichen Eigenschaft

Bei der Eigenschaft `Reaction` muss sich beim Einsatz in einer Pose jedoch die Bedeutung ein wenig ändern. Der Wert `Online` bedeutet beim Einsatz in einer Pose, dass die Anwendung die ganze Zeit, in der die Pose eingenommen wird, reagieren soll. Je länger die Pose gehalten wird, desto länger reagiert die Anwendung entsprechend. Der Wert `Offline` hingegen bedeutet, dass genau einmal auf die Pose reagiert wird. Das ist auch – analog zur Beschreibung in Kap. 7.2.1.12 – das Verhalten, das bei Fehlen der Eigenschaft erwartet wird. Somit ist die Bedeutung ist grundlegend wie in Kap. 7.2.1.12 beschrieben, mit dem (feinen) Unterschied, dass auf Offline-Posen sofort reagiert werden soll und nicht – wie bei Offline-Gesten – auf das Ende der Ausführung gewartet wird. Während bei Offline-Gesten jedoch bis zum Ende der Geste auf weitere Eingaben gewartet werden muss, ehe die Verarbeitung starten kann, sind bei Posen per Definition keine Eingaben mehr notwendig, sobald die Pose erreicht wird.

Einige Eigenschaften sind aufgrund der Natur einer Pose nur eingeschränkt bzw. gar nicht verwendbar:

- Anzahl der Werkzeuge (`Tool Count`): Die menschliche Anatomie legt für die Anzahl der verwendeten Werkzeuge eine natürliche Grenze fest, denn mehr als zwei Hände können nicht eingesetzt werden. Die Verwendung unterschiedlicher Finger würde bei Posen im Gegensatz zu Oberflächengesten in eine andere Form resultieren und daher über die graphische Repräsentation dargestellt werden. Es ist daher anzunehmen, dass die Anzahl der Werkzeuge bei räumlichen Posen eher implizit denn explizit definiert wird.

- Auflagefläche (`Contact Area`): Räumliche Posen sind nicht von einer Oberfläche abhängig, entsprechend gibt es auch keine Auflagefläche. Darüber hinaus definiert die Eigenschaft die Veränderung der Auflagefläche, also eine Bewegung, die bei Posen nicht gegeben ist.

- Geschwindigkeit (`Speed`): Setzt eine Bewegung voraus, räumliche Posen sind jedoch statisch.

- Rotation (`Rotation`): siehe Geschwindigkeit.

- Start- und Endpunkt (`Start`/`End`): siehe Geschwindigkeit.

Posen mit beiden Händen können analog zur Spezifikation einer unterschiedlichen Anzahl der Werkzeuge (vgl. Kap. 7.2.1.1) definiert werden (ungeachtet der o. g. Einschränkung). Beide Hände können in die graphische Repräsentation aufgenommen werden, so wie in Abb. 11.6 dargestellt. In diesem Falle gelten definierte Eigenschaften für die gesamte Pose, also beide Hände. Eine Alternative wäre, beide Hände als parallel ausgeführte GestureCards in einer zusammengesetzten Karte zu definieren. Dann wären unterschiedliche Eigenschaften, z. B. Begrenzung des Ortes der Ausführung, möglich, jedoch müsste u. U. die Position der beiden Posen zueinander beschrieben werden. Außerdem können auch für Posen Rückgaben definiert werden, die für die Anwendung zur weiteren Verarbeitung relevant sind. In Abb. 11.6 wird z. B. der Ort der Ausführung als Rückgabe definiert.

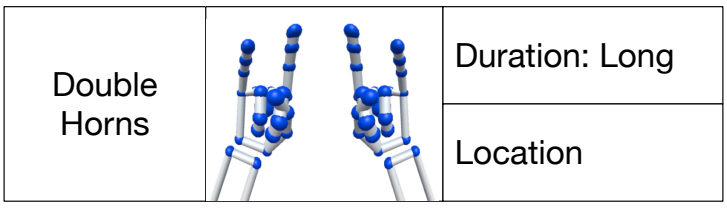


Abbildung 11.6: Eine Pose mit beiden Händen

Darüber hinaus können Posen mit beiden Händen auch verschiedene GestureCards aufgeteilt werden. Da die Posen gleichzeitig durchgeführt werden, wird die zeitliche Relation der Parallelität analog zu einer Multi-Touch-Geste verwendet. Abb. 11.7 verdeutlicht, wie die Geste aus Abb. 11.6 getrennt dargestellt wird. Die räumliche Relation muss in einer der beiden Teilgesten mit der Eigenschaft `Position` definiert werden, andernfalls wäre eine beliebige Positionierung der Hände möglich. Im gegebenen Beispiel sollen die Hände aber nebeneinander positioniert werden.

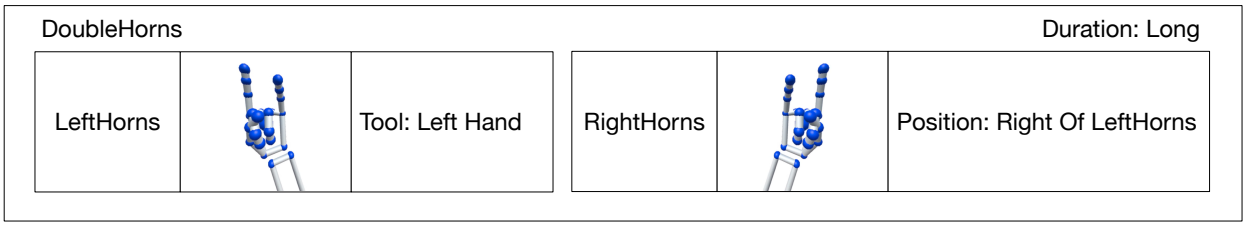


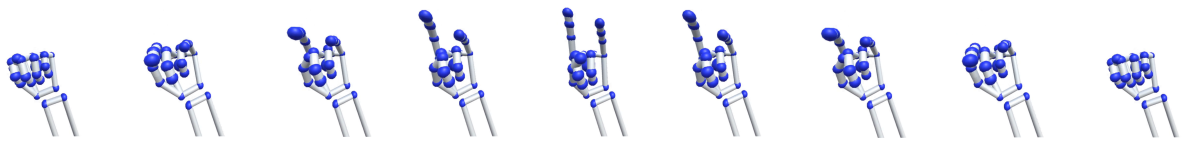
Abbildung 11.7: Eine Pose mit beiden Händen als getrennte Gesture-Cards

Die Verwendung technischer Unterstützung erlaubt eine weitere interessante Option, die u. U. die Lesbarkeit erhöht: Die Pose kann animiert und in einer kurzen Videosequenz wiedergegeben werden. Dabei wird die graphische Repräsentation durch die Animation ersetzt, die die Bewegung der Hand wie in Abb. 11.8 dargestellt zeigt. Allerdings verschwimmt dadurch die Grenze zwischen Pose und Geste, denn die Integration von Bewegung ist eigentlich gemäß der Definition in Kap. 4.1.2 der entscheidende Unterschied. Die Pose wird letztlich zur Geste, wenn die Bewegung als Teil der Pose (Geste) wahrgenommen wird und u. U. der Eindruck entsteht, dass die Bewegung eine semantische Bedeutung für die Anwendung hat. Anders ausgedrückt: In den vorher beschriebenen statischen Abbildungen ist letztlich irrelevant, *wie* die Pose erreicht wird; eine explizite Darstellung in der Animation gibt den Weg zur Pose jedoch vor. Daher sollte diese Variante nur gewählt werden, wenn die Animation eine deutliche Verbesserung der Lesbarkeit mit sich bringt (oder tatsächlich die Definition einer Geste notwendig ist). Eine Vereinfachung wäre u. U. schon, wenn das Handmodell nicht statisch, sondern in einer drehenden Animation angezeigt wird und somit von allen Seiten betrachtet werden kann. Mit passender Software-Unterstützung ließe sich dies auch vom Benutzer steuern und die eindeutige Trennung zwischen Pose und Geste wiederherstellen. Der Einsatz von Videos hat darüber hinaus den Nachteil, dass passende Software zum Abspielen vorhanden sein muss.

11.2 Darstellung von Gesten

Zur Beschreibung von Bewegung werden in GestureCards Pfeile eingesetzt, die den Bewegungspfad beschreiben. Diese erscheinen auf den ersten Blick intuitiv, vor allem beim Einsatz von Oberflächengesten: Die Dimensionen des Mediums zur Notation und der anschließenden Umgebung der Ausführung sind identisch. Bei räumlichen Gesten hingegen wird eine dritte Dimension hinzugefügt, die entweder durch Manipulation der optischen Wahrnehmung oder durch verschiedene Optionen der Notation dargestellt werden kann. So wird z. B. in Labanotation eine Kombination aus unterschiedlichen Farben, Schraffierungen und Formen je Dimension verwendet (Hutchinson Guest,

Abbildung 11.8: Darstellung von Schlüsselszenen in einem Video, das die Ausführung einer Pose zeigt



2005). Der Einsatz von Pfeilen zur Abbildung des Bewegungspfades erreicht somit eine natürliche Grenze, denn die fehlende dritte Dimension muss auf dem Medium zur Notation kenntlich gemacht werden.

Im Folgenden werden zwei unterschiedliche Ansätze vorgeschlagen, die je nach Einsatzzweck unterschiedliche Vor- und Nachteile haben. Beide Ansätze haben gemein, dass Animationen in unterschiedlichen Ausprägungen eingesetzt werden. So soll zunächst das Handmodell, das in Kap. 11.1 zur Darstellung von Posen eingesetzt wird, auch zur Beschreibung von Gesten verwendet werden (Kap. 11.2.1). Es eignet sich insbesondere, wenn die Haltung der Hand für die Geste relevant ist oder eine feingranulare Definition der Finger notwendig ist. Anschließend wird gezeigt, wie animierte Bewegungspfade in GestureCards integriert werden (Kap. 11.2.2). Sie eignen sich vor allem, wenn die Haltung von Hand und Fingern keine Rolle spielt.

11.2.1 Einsatz eines Handmodells

Die Verwendung eines Handmodells zur Definition einer Geste soll anhand des folgenden Beispiels dargelegt werden: Die rechte Hand wird zu Beginn in der gängigen Pose des Zeichens für *Victory* – Zeige- und Mittelfinger werden V-förmig abgespreizt – gehalten. Anschließend werden die beiden Finger zweimal aufeinander zu- und voneinander weggeführt. Die Geste wird passend als `Scissor`[1] betitelt. Abb. 11.9 zeigt, wie die Geste mit GestureCards definiert werden kann. Eine spezielle Anpassung für räumliche Gesten ist nicht nötig, da es sich zwar um eine räumliche Geste handelt, die Bewegung jedoch lediglich in einer Dimension durchgeführt wird.

[1] Englisch für *Schere*

Das Beispiel spezifiziert die Geste weitgehend korrekt. In den Teilgesten `Left1` und `Right1` werden zunächst der Zeige- und Mittelfinger (festgelegt durch die Eigenschaft `Tool` in beiden Teilgesten) der rechten Hand (von der übergeordneten, zusammengesetzten Geste ebenfalls durch die Eigenschaft `Tool` festgelegt) aufeinander zugeführt. Die Anordnung nebeneinander besagt, dass beide Teilgesten gleichzeitig ausgeführt werden. In den folgenden Teilgesten `Left2` und `Right2` werden die beiden Finger wieder voneinander weg geführt. Eine erneute Definition der Eigenschaft `Tool` ist an dieser Stelle nicht notwendig, denn der direkte Anschluss der Teilgesten an ihre Vorgänge definiert, dass beide Gesten ohne Verzögerung durchgeführt werden und somit keine andere Möglichkeit besteht. Die erneute Definition des Werkzeugs hingegen betont diesen Umstand nochmal. Darüber hinaus definiert die übergeordnete Geste, dass die Geste zweimal ausgeführt

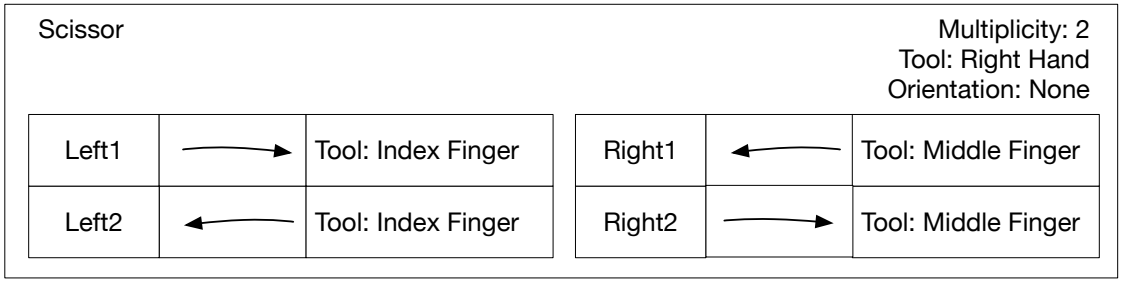


Abbildung 11.9: GestureCard für die Geste `Scissor`

werden muss (`Multiplicity: 2`) und dass die Ausrichtung nicht relevant ist (`Orientation: None`), d. h. die Hand unterschiedlich gehalten werden kann. Allerdings unterschlägt die Beschreibung ein wesentliches Merkmal: Die beiden aktiven Finger sollen abgespreizt werden, während die anderen Finger eingeknickt sind. GestureCards sehen in der Darstellung der Bewegung mit Pfeilen auch nicht vor, dass die inaktiven Gliedmaßen explizit spezifiziert werden – diese Entscheidung wurde bewusst getroffen, um die Notation einfach und auf das Wesentliche fokussiert zu halten.

An dieser Stelle wird ein weiterer Unterschied zwischen Oberflächengesten und räumlichen Gesten deutlich: Während bei Oberflächengesten eine Unterscheidung zwischen aktiven und inaktiven Teilen des verwendeten Werkzeugs nicht notwendig ist, da zur Geste ausschließlich die Berührungen der Oberfläche gehören, können räumliche Gesten die explizite Definition der inaktiven Teile des Werkzeugs verlangen. Alternativ kann diese Geste auch so betrachtet werden, dass sie in einer Pose startet und endet, die mit Pfeilen aber per Definition nicht beschrieben werden kann. Die Darstellung in Abb. 11.9 macht aber keine Aussage über die Haltung der Hand; die Durchführung der Geste wäre auch korrekt, wenn dabei die drei nicht aktiven Finger der Hand ebenfalls abgespreizt werden würden.

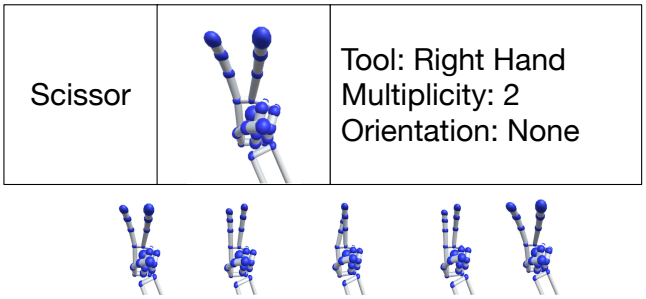


Abbildung 11.10: GestureCard für die Geste `Scissor` mit integrierter Animation der graphischen Repräsentation

Zur Beschreibung derartiger Gesten wird daher wieder auf das bereits in Kap. 11.1 zur Darstellung von Posen verwendete Handmodell zurückgegriffen. Zur Veranschaulichung der Bewegung wird das Modell animiert. Dadurch kann sowohl die vollständige Hand inklusive der passiven Finger, als auch die Bewegung der einzelnen Finger zueinander dargestellt werden. Abb. 11.10 zeigt die entsprechende GestureCard und die Schlüsselszenen einer passenden Animation, die anstelle der graphischen Repräsentation verwendet wird.

Aus den verschiedenen Informationen, die zur Beschreibung in Abb. 11.9 notwendig waren, sind die Informationen der Teilgesten aus den Eigenschaften in die graphische Repräsentation gewandert. Dadurch kann die Geste als einfache Geste und somit kompakter dargestellt werden, da die Trennung in einzelne Teilgesten nicht mehr notwendig ist. Insbesondere ist eine Definition der unterschiedlichen Werkzeuge – in Abb. 11.9 die verwendeten Finger – auf die verwendete Hand reduziert.

Die Einführung von Animationen zur Darstellung von Gesten wirft aber zwangsläufig die Frage auf, wie in zusammengesetzten Gesten mit verschiedenen Eigenschaften umgegangen werden soll, insbesondere wenn Eigenschaften voneinander abhängen. So könnte eine Teilgeste `S2` dort starten, wo

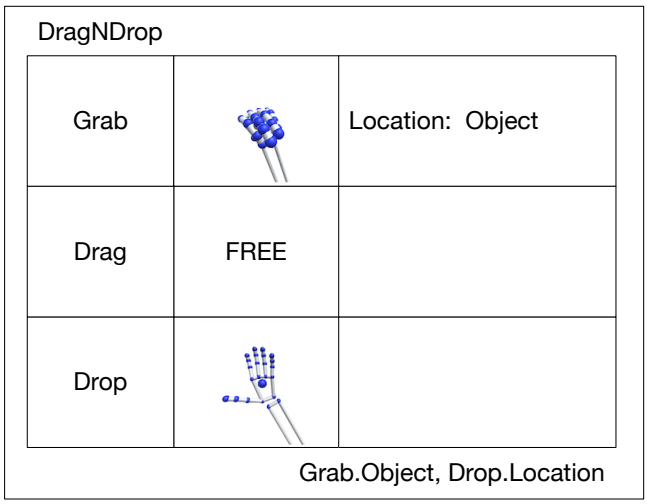


Abbildung 11.11: GestureCard für die Geste `DragNDrop`

eine Teilgeste `S1` endet.[2] Grundsätzlich findet keine Synchronisation zwischen Teilgesten statt, d. h. die graphische Repräsentation jeder Teilgeste ist autonom zu betrachten. Auch werden zusätzliche Eigenschaften nicht mit der graphischen Repräsentation im Video in Einklang gebracht. So wird eine Geste, in der die Eigenschaft `Speed: Slow` gesetzt ist, in der graphischen Repräsentation genauso schnell ausgeführt wie eine Geste, in der die Eigenschaft `Speed: Fast` gesetzt ist. Hintergrund dieser Entscheidung ist, dass die graphische Repräsentation analog zur Spezifikation von Gesten mit Pfeilen ausschließlich den definierten Eigenschaften vorbehalten ist und nicht mit anderen Eigenschaften vermischt werden sollte.

[2] Entsprechend müsste `Start: S1.End` in der Teilgeste `S2` gesetzt werden.

Einzig für die zeitliche Beziehung zwischen Teilgesten erscheint eine Synchronisation sinnvoll: Videos von Teilgesten, die sequentiell ausgeführt werden, sollen auch hintereinander gestartet werden; ebenso sollten Videos von Teilgesten, die parallel ausgeführt werden, gleichzeitig ablaufen. Abb. 11.11 zeigt ein Beispiel: Das räumliche Pendant zur klassischen Drag'n'Drop-Geste, mit der bei WIMP-UIs virtuelle Objekte verschoben werden. In diesem Fall wird ein virtuelles Objekt mit der Hand durch Zusammenführen der Finger *gegriffen*, an einen anderen Ort gebracht und hier durch Auseinanderführen der Finger wieder losgelassen. Entscheidend für die weitere Verarbeitung ist das gegriffene Objekt sowie der Ort, an dem es abgelegt wird, entsprechend sind die Rückgaben der GestureCard definiert. Die zugehörigen Schlüsselszenen der Animationen sind in Abb. 11.12 zu sehen.

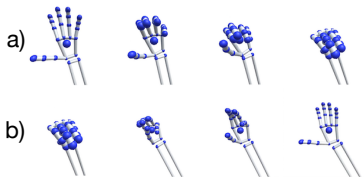


Abbildung 11.12: Schlüsselszenen des Videos für die Teilgesten a) `Grab` und b) `Drop`

11.2.2 Einsatz animierter Bewegungspfade

Es ist anzunehmen, dass die Darstellung als Handmodell nicht für jede Geste die gewünschte Lesbarkeit ermöglicht. Zwar können so die Bewegungen der unterschiedlichen Finger detailliert dargestellt werden, jedoch sind auch Gesten denkbar, die mit der ganzen Hand ausgeführt werden. Außerdem können komplexe geometrische Figuren ausgeführt werden. Für derartige Gesten kann die Darstellung des Pfades durch Pfeile analog zur Darstellung bei Oberflächengesten besser geeignet sein. In Kap. 10 wurde gezeigt, wie GestureCards mit animierten Bewegungspfaden im AugIR eingesetzt wurden

und dass die Probanden dies als nützliche Aufwertung empfanden. Der Einsatz eines Handmodells zur Spezifikation räumlicher Gesten und Posen ist bereits auf digitale Unterstützung angewiesen, daher soll im Folgenden das Konzept der Animationen aus dem AugIR erneut aufgegriffen und für die Spezifikation räumlicher Gesten aufgearbeitet werden. Die Herausforderung ist dabei, einen Weg zu finden, die Bewegungen auf der z-Achse treffend darzustellen.

Räumlichkeit lässt sich auf zweidimensionalen Medien mit unterschiedlichen Maßnahmen darstellen. Dafür notwendige Techniken existieren schon seit langem, z. B. auf Bildern bekannter Künstler. Ein anderer Bereich, bei dem die räumliche Darstellung auf Bildern weit verbreitet ist, sind technische Zeichnungen und Konzepte, die beim Entwurf neuer Produkte entwickelt und graphisch dargestellt werden. Dem liegt zugrunde, dass das menschliche Gehirn die räumliche Entfernung von Objekten aufgrund unterschiedlicher Reize bestimmt. Nagata (1987) gibt einen Überblick über die unterschiedlichen Möglichkeiten, auf die das menschliche Gehirn zur Erkennung von Tiefeninformationen in zweidimensionalen Graphiken zurückgreifen kann. Demnach können u. a. Veränderungen der Größe und Bewegung auf Räumlichkeit hindeuten. Sowohl die Bewegung des Betrachters als auch die Bewegung des betrachteten Objekts können dazu eingesetzt werden.

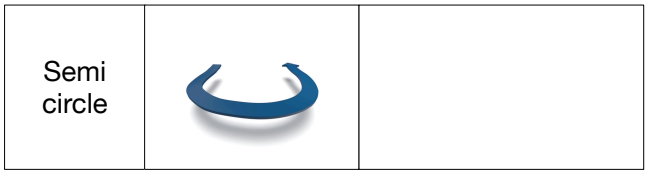


Abbildung 11.13: GestureCard für die Geste `Semicircle`

In GestureCards soll eine Kombination der oben genannten Varianten eingesetzt werden. Abb. 11.13 zeigt ein Beispiel für eine Geste. Dabei wird die Hand vor dem Körper in einem liegenden Halbkreis bewegt. Bereits bei der Betrachtung der statischen Abbildung entsteht durch die Veränderung der Größe über den Verlauf des Pfeils ein räumlicher Eindruck. Teile der Ausführung, die sich näher am Körper befinden, werden größer dargestellt als Teile, die sich weiter weg befinden. Dieser Eindruck wird durch die Animation des Pfeils, deren Schlüsselszenen in Abb. 11.14 dargestellt werden, noch verstärkt. Die Animation baut sich in Bewegungsrichtung der Geste auf, bis der komplette Pfad dargestellt ist.

Dynamik wird in diesem Falle genauso dargestellt wie für Oberflächengesten: Die Eigenschaft `Direction` kann mit den Werten `Towards` oder `Away from` sowie der Referenz auf eine andere Teilgeste oder ein virtuelles Objekt definiert werden. Auch die Unabhängigkeit von der Ausrichtung wird wie bei Oberflächengesten definiert (vgl. Kap. 7.2.1.4).

Abbildung 11.14: Schlüsselszenen des Videos für die Geste in Abb. 11.13

12 Vergleich graphischer Repräsentationen für räumliche Gesten

In Kap. 11.2 wurden verschiedene Variationen zur graphischen Repräsentation räumlicher Gesten vorgestellt. GestureCards unterstützen prinzipiell alle dargestellten Möglichkeiten, jedoch bleiben unterschiedliche Fragen offen, insbesondere über die Wirksamkeit der verschiedenen Varianten. Während die Darstellung als Handmodell für Posen offensichtliche Vorteile bietet, da die Positionierung der Finger im Raum detailliert dargestellt werden kann, ist bei Gesten – also der Beschreibung von Bewegung – nicht auf Anhieb deutlich, ob das Handmodell oder die Pfeile besser zu verstehen ist. In der nachfolgend vorgestellten Studie soll verglichen werden, wie Probanden eine Darstellung der Geste als Handmodell oder als Pfeil auffassen. In Kap. 10 wurden bei der Integration von GestureCards in den AugIR die Pfeile in der graphischen Repräsentation animiert. Die Mehrheit der Probanden empfand diese Erweiterung als hilfreich. In dieser Studie soll diese Erkenntnis aufgegriffen werden: Ebenso wie das Handmodell werden auch die Pfeile in einer kurzen Animation dargestellt.

Die in Kap. 8 beschriebene Studie deutete bereits an, dass die jeweilige Geste einen gewissen Einfluss auf das Ergebnis hat. Dieser Eindruck verstärkte sich in der Studie, die in Kap. 9 beschrieben wurde. Für die folgende Studie besteht die Frage nach dem Einfluss der Geste somit auch, denn während der Entwicklung einer gestenbasierten Anwendung müssen die verantwortlichen Teammitglieder eine Entscheidung treffen, wie die jeweiligen Gesten dargestellt werden sollen. Es wird davon ausgegangen, dass unterschiedliche räumliche Gesten mit verschiedenen Darstellungen unterschiedlich gut beschrieben werden können, allerdings sind die zu berücksichtigenden Faktoren unklar. Der Studie liegt daher folgende Forschungsfrage zugrunde:

Ist die Repräsentation mittels eines Handmodells oder räumlich gezeichneter Pfeile besser zur Beschreibung räumlicher Gesten geeignet?

Die Ergebnisse lassen eine eindeutige Beantwortung der Forschungsfrage nicht zu, denn für drei von vier erfassten Variablen sind für die beiden Gruppen keine statistisch signifikanten Unterschiede sichtbar. Der einzig messbare Unterschied ist, dass die Probanden, die mit dem animierten Handmodell arbeiteten, häufiger die Animation erneut abspielten. Außerdem legen die Ergebnisse nahe, dass vor allem die Interaktion zwischen der darzustellenden Geste und der Darstellungsform beachtet werden muss. Manche Gesten werden besser als Handmodell und manche besser mit animierten Pfeilen dargestellt. Über die Ursache kann im Rahmen dieser Studie nur spekuliert werden – hier ist weitere Forschung notwendig, die die unterschiedlichen Parameter ausarbeitet –, jedoch gibt sie erste Hinweise auf denkbare Faktoren. So ist eventuell die Form bei der Auswahl ebenso zu berücksichtigen wie die Anzahl zusätzlicher Eigenschaften.

Die Beschreibung der Studie folgt dem bei der Darstellung der anderen Studien verwendeten Muster. Zunächst werden die Grundlagen der Studie beschrieben und erläutert, wie die Studie aufgebaut wurde (Kap. 12.1). Anschließend werden die erfassten Daten ausgewertet (Kap. 12.2). Es folgt eine Diskussion der Ergebnisse, die in unterschiedliche Schlussfolgerungen und Implikationen für zukünftige Arbeiten mündet (Kap. 12.3). Abschließend werden potentielle Risiken betrachtet, die die Validität der Ergebnisse in Frage stellen können, und wie diese Risiken durch den Aufbau der Studie minimiert wurden (Kap. 12.4).

12.1 Aufbau der Studie

Diese Studie ist im Wesentlichen so aufgebaut wie die Studie, die in Kap. 8 beschrieben wurde. Nachfolgend werden daher die Unterschiede zwischen den beiden Studien fokussiert und identische Teile ggf. verkürzt wiedergegeben. Die Struktur, die Jedlitschka et al. (2008) vorgeschlagen haben, wird auch in dieser Studie zur Beschreibung des Aufbaus verwendet. Es werden die Ziele der Studie, die teilnehmenden Probanden, die zu lösenden Aufgaben (in Form der verwendeten Gesten), die eingesetzte Anwendung, die erfassten Variablen und zugehörigen Hypothesen, der Aufbau der Sitzungen mit Probanden sowie die Vorgehensweise zur Analyse der erfassten Daten beschrieben.

12.1.1 Ziele

In der Studie wird verglichen, ob und wie Probanden räumliche Gesten identifizieren können, die mit unterschiedlichen graphischen Repräsentationen definiert werden. Die erfassten Daten sollen zeigen, wie Probanden verschiedene Varianten verstehen und somit Aufschluss über die Verständlichkeit der eingesetzten graphischen Repräsentationen geben. Folgende Informationen werden während der Studie aufgenommen:

- Die Anzahl korrekt identifizierter Gesten
- Die Zuversicht, mit der Probanden Gesten in der jeweiligen graphischen Repräsentation identifizieren können
- Die Dauer, die Probanden benötigten, um definierte Gesten in der jeweiligen graphischen Repräsentation zu identifizieren
- Die Anzahl der Wiederholungen der Animation, die als graphische Repräsentation in der GestureCard gezeigt wird

Die erfassten Daten ähneln jenen, die in der Studie in Kap. 8 erfasst wurden. Es fehlt der Spickzettel, der den Probanden in dieser Studie nicht zur Verfügung stand. Zusätzlich wird erfasst, wie oft Probanden die Animation der Geste wiederholten.

12.1.2 Probanden

Die teilnehmenden Probanden wurden in der Reihenfolge ihrer Teilnahme abwechselnd der Gruppe Hands oder der Gruppe Arrows zugeordnet. Es ist davon auszugehen, dass die Zuweisung zu den Gruppen zufällig erfolgte, da für die einzelnen Sitzungen individuelle Termine vereinbart wurden und die Probanden nicht vorab nach ihrer persönlichen Präferenz für eine der graphischen Repräsentationen befragt wurden. Jeder Proband war Mitglied genau einer Gruppe; Überschneidungen gab es nicht. Jeder Gruppe wurde eine graphische Repräsentation für räumliche Gesten zugeordnet, mit der die die Probanden der Gruppe ausschließlich arbeiteten. Kein Proband arbeitete somit mit beiden verwendeten graphischen Repräsentationen.

An der Studie nahmen zehn Probanden teil, deren Sitzungen in das Gesamtergebnis einflossen; jede Gruppe bestand somit aus fünf Mitgliedern. Sechs Probanden waren männlich und vier weiblich. Sie verfügten über unterschiedliche Erfahrungswerte in der Software-Entwicklung und mit gestenbasierten Anwendungen. Einige Probanden hatten auch bereits ein grundlegendes Verständnis des Aufbaus von GestureCards. Weiterführende demographische Daten wurden nicht erfasst. Vor der Durchführung der Probandensitzungen wurde der Aufbau der Studie sowie insbesondere die verwendete Anwendung mit einem elften Teilnehmer getestet, jedoch wurden diese Ergebnisse verworfen und nicht in das Gesamtergebnis aufgenommen.

Mit den Probanden wurden individuelle Termine für die jeweilige Sitzung vereinbart. Alle Sitzungen liefen nach dem in Kap. 12.1.6 beschriebenem Schema ab. Abgesehen von Proband und Studienleiter war bei einer Sitzung keine weitere Person anwesend. Die verwendete Anwendung speicherte jede Sitzung unter einer eindeutigen, jedoch anonymen ID (z. B. dmfy), die zur Identifikation der Probanden verwendet wird.

12.1.3 Eingesetzte Gesten

In dieser Studie wurden neun[1] Gesten eingesetzt und als passende GestureCards mit den zu testenden graphischen Repräsentationen formuliert. Analog zu den in Kap. 8 und Kap. 9 beschriebenen Studie wurden die Gesten anonymisiert. So sollte eine Beeinflussung der Probanden durch die Namen der Geste vermieden werden. Alle Gesten wurden als *Gesture* gefolgt von einer Nummer benannt. Auf die Verwendung zusätzlicher Eigenschaften in den GestureCards wurde weitgehend verzichtet, allerdings war dies bei einzelnen Gesten nicht möglich. Die jeweiligen Beispiele veranschaulichen allerdings gut, welche Auswirkungen die Änderung der graphischen Repräsentation auf die Beschreibung der Gesten hat.

[1] Bei der ersten Konzeption der Studie wurde noch eine zehnte Geste angedacht, aber noch vor Durchführung der Studie entfernt. Daher fehlt im Folgenden eine Geste mit der Nummer 7.

Im Folgenden werden die Gesten und ihre Eigenheiten vorgestellt sowie eine Begründung für die Auswahl gegeben. Aus Gründen der Übersichtlichkeit wird bei Gesten, zu denen keine zusätzlichen Eigenschaften definiert werden mussten, auf eine Darstellung als GestureCard verzichtet und stattdessen lediglich die Schlüsselszenen der Animation abgebildet. Für alle anderen Fälle wird jeweils die komplette GestureCard dargestellt.

Geste Boxing

Bei dieser Geste sollen die Hände wie bei Boxschlägen abwechselnd nach vorne geführt werden. Es soll mit der rechten Hand gestartet und die linke Hand erst nach vorne geführt werden, während die rechte schon wieder zurückgezogen wird. In Kap. 7 wurde beschrieben, dass in der graphischen Repräsentation keine zeitliche Verzögerung definiert werden kann, allerdings wurde hier von der statischen Variante der Notation ausgegangen. Dieses Beispiel zeigt, wie Animationen eingesetzt werden können, um zeitliche Verzögerung darzustellen, und soll prüfen, wie Probanden darauf reagieren. Die Geste wird in Abb. 12.1 dargestellt.

Abbildung 12.1: Schlüsselszenen der Geste Boxing für die Gruppen Hands und Arrows

Geste Conductor

Diese Geste orientiert sich an der Bewegung, die ein Dirigent mit dem Taktstock zur Darstellung eines 4/4-Taktes durchführt. Dabei wird die Hand erst unten und dann nach links, rechts und wieder nach oben geführt, wobei die Bewegung nach rechts größer ausfällt als die Bewegung nach links und somit den Pfad, der bei der Bewegung nach unten ausgeführt wurde, wieder kreuzt. Die Geste ist in der Ausführung für beide Gruppen in Abb. 12.2 zu sehen.

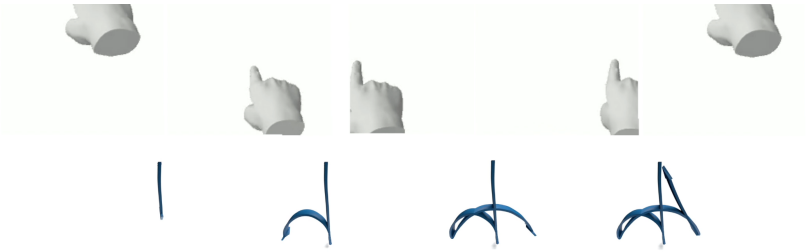


Abbildung 12.2: Schlüsselszenen der Geste Conductor für die Gruppen Hands und Arrows

Geste Gamma

Diese Geste soll in Form des griechischen Buchstabens Gamma ausgeführt werden, der wie ein Tropfen geformt ist. Abb. 12.3 stellt diese Geste in beiden Ausführungen dar.

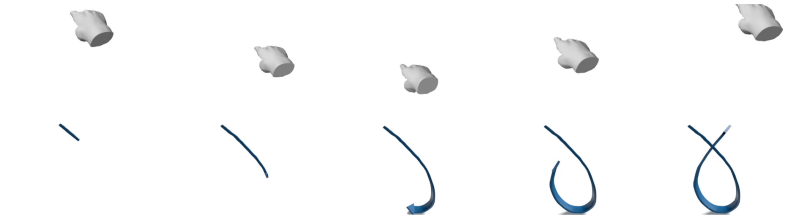


Abbildung 12.3: Schlüsselszenen der Geste Gamma für die Gruppen Hands und Arrows

Geste U

Diese Geste beschreibt einen liegenden Halbkreis mit einer Hand vor dem Körper. Die Hand muss zum Körper hin und anschließend wieder weggeführt werden. Sie ist in Abb. 12.4 dargestellt.

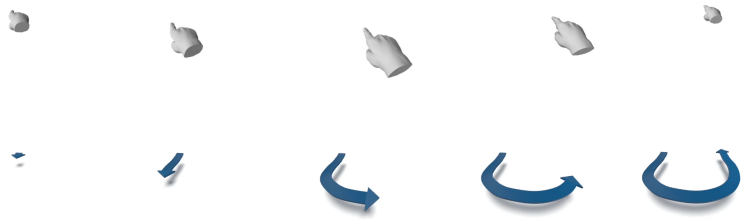


Abbildung 12.4: Schlüsselszenen der Geste U für die Gruppen Hands und Arrows

Geste Phi

Diese Geste orientiert sich wieder an einem griechischem Buchstaben: Sie soll in der Form eines Phi ausgeführt werden. Abb. 12.5 zeigt die Schlüsselszenen der zugehörigen Animationen.

Abbildung 12.5: Schlüsselszenen der Geste Phi für die Gruppen Hands und Arrows

Geste Present

Diese Geste beschreibt eine Bewegung nach rechts mit der Hand, die währenddessen um 180° gedreht wird. Grundlage ist die Bewegung, die beim Präsentieren von etwas verwendet wird. Die Umsetzung der Geste als GestureCard variiert für die getesteten graphischen Repräsentationen, denn die Darstellung mit Pfeilen erzwingt die Definition der Eigenschaft `Rotation`, um deutlich zu machen, dass die Hand bei der Bewegung gedreht werden soll. Bei der Darstellung als Handmodell ist die Eigenschaft nicht notwendig, denn die Drehung geht aus der graphischen Repräsentation hervor.

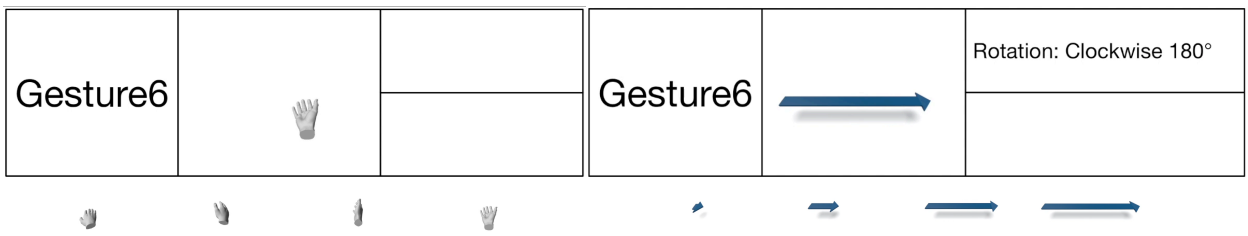


Abbildung 12.6: GestureCards der Geste Present für die Gruppen Hands und Arrows sowie zugehörige Schlüsselszenen

Geste X

Die Hand wird bei dieser Geste in Form eines X vor dem Körper geführt. Sie wird für beide Gruppen identisch definiert, lediglich die graphische Repräsentation wird ausgetauscht. Abb. 12.7 zeigt die Geste.

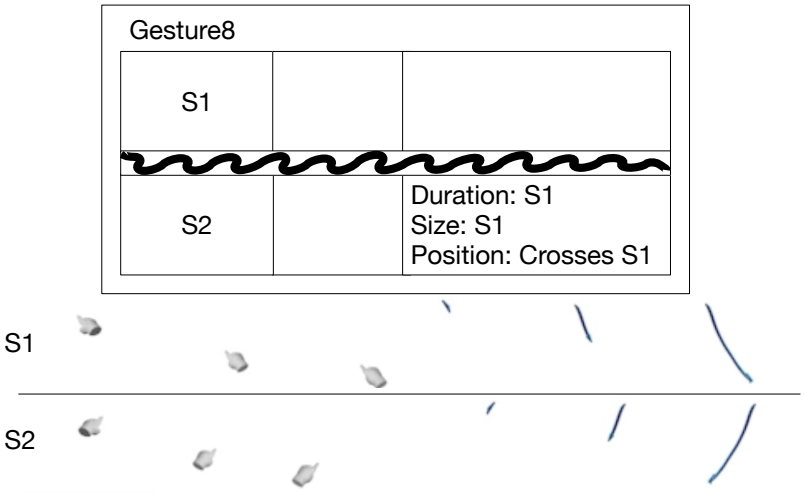


Abbildung 12.7: GestureCards der Geste X für die Gruppen Hands und Arrows sowie zugehörige Schlüsselszenen

Geste Drag'n'Drop

Diese Geste ist die räumliche Variante der klassischen Drag'n'Drop-Geste. Dazu wird mit der Hand gegriffen (dargestellt durch das schließen der Hand zur Faust), die Hand an einen anderen, beliebigen (angedeutet durch das Schlüsselwort *FREE*) Ort geführt und dann losgelassen (durch das Öffnen der Hand dargestellt). Da der wichtige Aspekt die Bewegung der Hand und der Finger ist, wurde auf eine genauere Definition verzichtet; eigentlich müsste die Geste noch mit `Location: Object` markiert werden. Die Geste wird in Abb. 12.8 dargestellt.

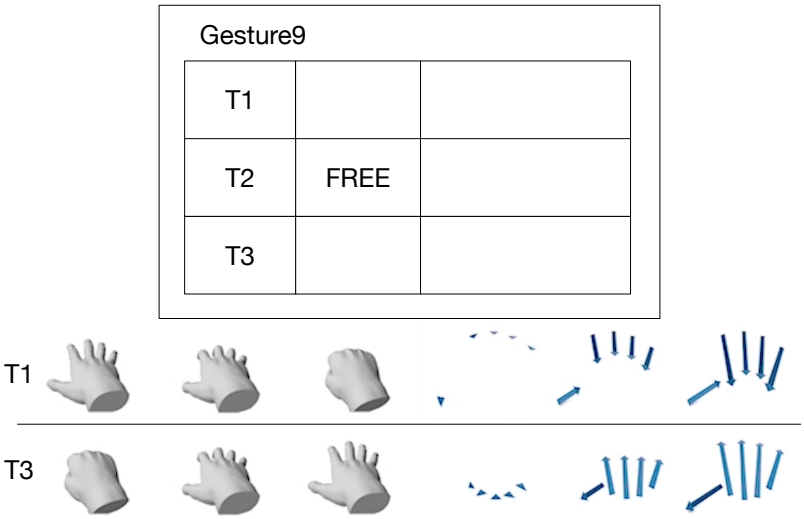


Abbildung 12.8: GestureCards der Geste Drag'n'Drop für die Gruppen Hands und Arrows sowie zugehörige Schlüsselszenen

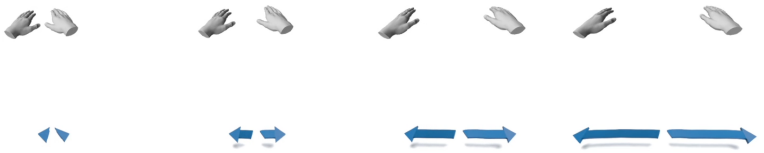


Abbildung 12.9: Schlüsselszenen der Geste Spread für die Gruppen Hands und Arrows

Geste Spread

Bei der letzten Geste müssen beide Hände auseinander gezogen werden. Sie ist das räumliche Pendant zur Spread-Geste auf Multi-Touch-Bildschirmen. Die Schlüsselszenen beider Animationen werden in Abb. 12.9 dargestellt.

12.1.4 Eingesetzte Anwendung

Zur Durchführung der Studie wurde eine Anwendung für ein Tablet entwickelt, mit der die Probanden durch die Studie geleitet wurden und die sie selbständig bedienen konnten. Auf dem Tablet lief das mobile Betriebssystem Android. Als Grundlage wurde die Anwendung, die bei der in Kap. 8 beschriebenen Studie eingesetzt wurde, genommen, jedoch für diese Studie vereinfacht und im Detail an die Zielstellung dieser Studie angepasst. So wurden der Spickzettel und die Möglichkeit, die aktuelle Geste zurückzusetzen, entfernt, da beide Teile für diese Studie nicht notwendig waren. Auch wurde die Einführung in GestureCards in dieser Studie nicht über die Anwendung, sondern durch die Studienleitung vorgenommen. Außerdem wurde auf die Darstellung eines Fensters einer potentiellen Anwendung, auf dem Gesten ausgeführt werden, verzichtet, da in den verwendeten Gesten kein Bezug zu UI-Elementen definiert wurde. Es war also ausschließlich die GestureCard der aktuellen Geste zu sehen. Dafür wurde die Möglichkeit hinzugefügt, den Probanden bei der Ausführung der Gesten zu filmen, sowie eine Vorschau für den Probanden direkt anzuzeigen.

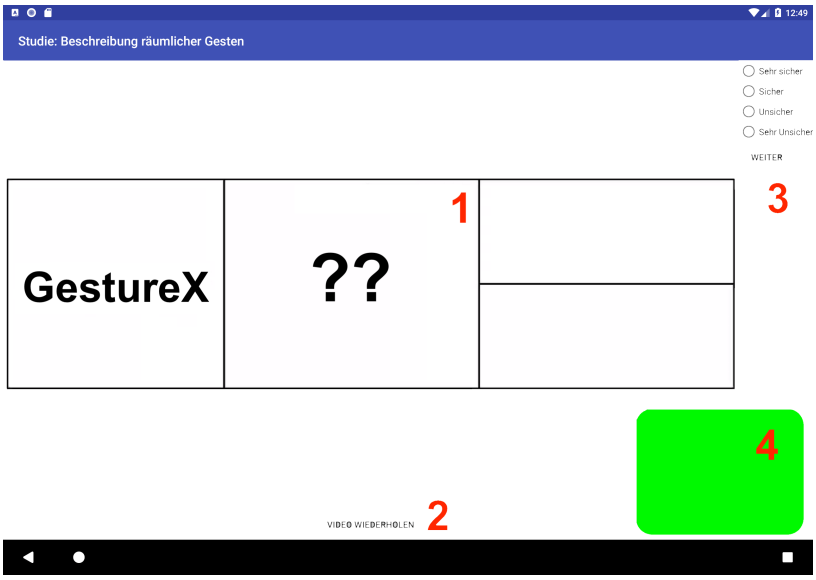


Abbildung 12.10: UI der für diese Studie verwendeten Anwendung. Die Darstellung wurde den Probanden während der Einführung als Beispiel gezeigt.

Abb. 12.10 zeigt das UI der Anwendung. Im linken, größeren Teil des Bildschirms wurde die aktuelle Geste als GestureCard mit einer passenden Animation als graphische Repräsentation dargestellt (Punkt 1 in Abb. 12.10). Benutzer hatten die Möglichkeit, die Animation der GestureCard über den unterhalb platzierten Button erneut zu starten (Punkt 2 in Abb. 12.10). Auf der rechten Seite konnten die Probanden ihre Zuversicht bewerten und zur nächsten Geste wechseln (Punkt 3 in Abb. 12.10). Die Wahl der Zuversicht war Voraussetzung, um die nächste Geste aufrufen zu können. Außerdem sahen die Probanden sich selbst in einer Vorschau des aufgenommenen Videos (Punkt 4 in Abb. 12.10). Die Anwendung protokollierte unterschiedliche Daten für jeden Probanden, die zur Auswertung der Studie verwendet wurden:

- Die Dauer der Bearbeitung jeder Geste
- Die Anzahl der Wiederholungen der Animation, die die graphische Repräsentation der Geste zeigt
- Das Maß an Zuversicht, das die Probanden nach Bearbeitung der jeweiligen Geste auf der Likert-Skala festlegten
- Ein Video des Probanden während der Durchführung der Studie

12.1.5 Hypothesen und Variablen

Im Folgenden werden die der statistischen Auswertung zugrundeliegenden Hypothesen und die notwendigen Variablen definiert, mit deren Hilfe die initial aufgeworfene Forschungsfrage beantwortet und die in Kap. 12.1.1 definierten Ziele erreicht werden sollen. Dazu werden die im Laufe der Studie erfassten Daten verwendet, die die Leistungen der Probanden widerspiegeln. Für jede Geste, die die Probanden bearbeiteten, ergeben sich vier Variablen, die zur Auswertung herangezogen werden. Tab. 12.1 listet alle abhängigen Variablen der Studie sowie ihre relevanten Eigenschaften.

Die Variablen wurden entweder durch die verwendete Anwendung oder die aufgenommenen Videos erfasst. Sowohl die Variable *Dauer* als auch die Variable *Replays* wurden während jeder einzelnen Sitzung durch die eingesetzte Anwendung gemessen. Die Variable *Zuversicht* wurde durch Selbsteinschät-

Tabelle 12.1: Überblick über alle abhängigen Variablen

Variable	Beschreibung	Einheit	Skala	Mögliche Werte
Korrekt	Die Geste wurde korrekt identifiziert	boolean	nominal	ja, nein
Zuversicht	Die Zuversicht der Probanden, die Geste korrekt identifiziert zu haben	Likert-Skala	ordinal	sehr sicher, sicher, unsicher, sehr unsicher
Dauer	Die Zeit, die die Probanden für jede Geste benötigen	Sekunden	metrisch	0..∞
Replays	Die Anzahl Wiederholungen der Animation	Numerisch	metrisch	0..∞

zung der Probanden erfasst, die ihre Zuversicht über eine in der Anwendung integrierte Likert-Skala bewerteten. Abweichend von diesem Vorgehen wurde die Variable *Korrekt* im Nachhinein mithilfe der Videoaufzeichnungen der Sitzungen durch die Studienleitung festgelegt. Probanden erkannten eine Geste korrekt, wenn die dargestellte Geste im Sinne der Beschreibung ausgeführt wurde. Dabei wurde ausschließlich die ausgeführte Bewegung bewertet und nicht die Handhaltung. Eine partielle Lösung war, analog zu der in Kap. 8 beschriebenen Studie, nicht möglich. Das Vorgehen während der Sitzungen wird in Kap. 12.1.6 beschrieben.

Für die Forschungsfrage soll beantwortet werden, welche der graphischen Darstellungen für räumliche Gesten die besseren Ergebnisse erzielt, entsprechend sind auch die Hypothesen für jede Variable zu definieren. Es wird erwartet, dass die Gruppe Hands bei der Auswertung der Variablen *Dauer*, *Zuversicht* und *Korrekt* bessere Ergebnisse erzielt: Die Probanden sind schneller, zuversichtlicher und erkennen mehr Gesten korrekt als die Gruppe Arrows. Für die Variable *Replays* hingegen wird die Gruppe Arrows im Vorteil gesehen: Die Probanden werden seltener auf eine Wiederholung der Animation zurückgreifen, da der Bewegungspfad durch die Pfeile eher nachvollzogen werden kann. Analog zum Vorgehen der Studie aus Kap. 8 werden im Folgenden den einzelnen Hypothesen die zugehörigen Variablen zur Identifikation hinzugesetzt; $H_{1Korrekt}$ bezeichnet also die Alternativhypothese der Variable *Korrekt* und $H_{0Zuversicht}$ die Nullhypothese der Variable *Zuversicht*.

Die Probanden wurden in der Studie gebeten, neun als GestureCard definierte Gesten zu identifizieren, in denen zwei verschiedene graphischen Repräsentationen verwendet wurden: ein animiertes Handmodell oder animierte Pfeile. Die Probanden arbeiteten mit genau einer graphischen Repräsentation, die ihnen in der Reihenfolge ihrer Teilnahme abwechselnd zugewiesen wurde. Somit können die Probanden in zwei Gruppen eingeteilt werden, die anhand der graphischen Repräsentation unterschieden werden: Gruppe Hands und Gruppe Arrows. Die graphische Repräsentation ist demnach ein Zwischensubjektfaktor bzw. eine unabhängige Variable der Studie. Die in Tab. 12.1 definierten Variablen können auf diesen Faktor reduziert werden, d. h. es ist anzunehmen, dass sich zwischen den Gruppen Unterschiede in den Ergebnissen der Variablen finden. Tab. 12.2 fasst die Hypothesen für jede Variable mit Blick auf den Faktor Gruppe (bzw. graphische Repräsentation) zusammen.

Jede Gruppe arbeitete mit denselben[2] Gesten, um die Vergleichbarkeit zu gewährleisten. Die Gesten wurden den Probanden in zufälliger Reihenfolge präsentiert, um zu verhindern, dass die Sequenz der Gesten das Ergebnis beeinflusst. Dennoch kann die jeweilige Geste Einfluss auf die Ergebnisse haben, da anzunehmen ist, dass manche Gesten mit einer der beiden gewählten graphischen Repräsentationen besser zu identifizieren sind. Die Geste ist also als Innersubjektfaktor zu betrachten. Bei der Studie handelt es sich somit um eine zweifaktorielle Studie mit den beiden unabhängigen Variablen graphische Repräsentation und Geste. Zusätzlich kann eine Interaktion zwischen beiden Faktoren auftreten. Für die Auswertung der Studie müssen diese beiden Faktoren – analog zum Vorgehen bei der Auswertung der Studie, die in Kap. 9 vorgestellt wurde – ebenfalls berücksichtigt werden.

[2] Die graphischen Repräsentationen basierten vorwiegend auf denselben Bewegungsdaten, die mithilfe des Leap Motion Controllers aufgezeichnet wurden. Es kann daher davon ausgegangen werden, dass es sich tatsächlich um unterschiedliche Darstellungen derselben Gesten handelte. Zur Generierung der Pfeildarstellung wurde ein markanter Punkt – typischerweise der Zeigefinger – ausgewählt. Ausnahmen sind die Gesten Present und Drag'n'Drop, für die die Pfeildarstellung separat erstellt wurde.

Variable	Nullhypothese H_0	Alternativhypothese H_1
Korrekt	Probanden der Gruppe Hands identifizieren höchstens so viele Gesten korrekt wie Probanden der Gruppe Arrows	Probanden der Gruppe Hands identifizieren mehr Gesten korrekt als Probanden der Gruppe Arrows
Zuversicht	Probanden der Gruppe Hands sind höchstens so zuversichtlich, Gesten korrekt identifiziert zu haben, wie Probanden der Gruppe Arrows	Probanden der Gruppe Hands sind zuversichtlicher, Gesten korrekt identifiziert zu haben, als Probanden der Gruppe Arrows
Dauer	Probanden der Gruppe Hands benötigen mindestens so lange, um Gesten abzuleiten, wie Probanden der Gruppe Arrows	Probanden der Gruppe Hands leiten Gesten schneller ab als Probanden der Gruppe Arrows
Replays	Probanden der Gruppe Arrows wiederholen die Animation mindestens so häufig wie Probanden der Gruppe Hands	Probanden der Gruppe Arrows wiederholen die Animation seltener als Probanden der Gruppe Hands

Tabelle 12.2: Überblick über alle Hypothesen der Studie zum Vergleich graphischer Repräsentationen für räumliche Gesten

12.1.6 Aufbau der Sitzungen

Die Probanden nahmen an der Studie in eigenen, individuell vereinbarten Sitzungen ohne Publikum teil, die zwischen zehn und 15 Minuten dauerten. Der Ablauf jeder Sitzung war in die folgenden Phasen gegliedert:

- Einführung in die Studie und Erläuterung des Vorgehens durch den Studienleiter
- Erläuterung der verwendeten Anwendung durch den Studienleiter
- Erläuterung der grundlegenden Konzepte von GestureCards durch den Studienleiter
- Start der Studie durch den Probanden

Für die Erläuterungen wurden Ausdrucke der notwendigen Informationen auf Papier verwendet. Diese bestanden aus einem Screenshot der Anwendung (vgl. Abb. 12.10) sowie Beispielen des grundlegenden Aufbaus von GestureCards für Oberflächengesten, die auch in Kap. 7 zu finden sind. Zur Durchführung der Studie wurde ein Tablet eingesetzt, auf dem die in Kap. 12.1.4 beschriebene Anwendung lief. Da die Probanden beide Hände frei haben sollten und von der Frontkamera des Tablets gefilmt wurden, wurde das Tablet von der Studienleitung in den Händen gehalten.

Zunächst wurde den Probanden erläutert, dass sie nachfolgend eine Menge von räumlichen Gesten sehen werden, die in GestureCards beschrieben sind. An dieser Stelle wurden die Probanden auch darauf hingewiesen, dass sie ausschließlich Gesten mit den Händen bearbeiten werden und nicht auf weitere Körperteile eingehen müssen. Der grundlegende Aufbau von GestureCards wurden anhand einer beispielhaften Darstellung in der eingesetzten Anwendung kurz erläutert.

Danach wurde das UI der Anwendung beschrieben und kurz umrissen, wie der Proband die Studie durchführt. An dieser Stelle wurden die Probanden

auch gebeten, ihre Gedanken im Sinne des *Think Aloud* laut zu formulieren, um weitere Einblicke in ihr mentales Modell der Beschreibung räumlicher Gesten zu gewinnen. Jede Sitzung wurde daher – das Einverständnis des Probanden vorausgesetzt – mit Bild und Ton aufgezeichnet. Dafür wurde die Frontkamera des verwendeten Tablets eingesetzt.

Nachdem die Einführung abgeschlossen war, wurden die Probanden aufgefordert, eventuelle Fragen zu stellen und konnten dann die Studie selbstständig starten. Den Probanden wurden während der Studie allgemeine Fragen sowie Fragen zum Aufbau und zu Konzepten von GestureCards gestattet, aber keine Fragen beantwortet, die sich auf Details der graphischen Repräsentation der Gesten bezogen. Zeitliche Vorgaben wurde nicht gemacht: Die Probanden konnten die Geschwindigkeit der Studie selber steuern, indem sie eine Geste selbstständig abschlossen und zur nächsten wechselten.

12.1.7 Analyse

Die Daten wurden in zwei Schritten analysiert. Zunächst wurde explizit auf den Unterschied zwischen den beiden Gruppen Arrows und Hands mit Blick auf die in Kap. 12.1.5 definierten Variablen und Hypothesen eingegangen. Je nach Art der Variablen wurden hierzu unterschiedliche statistische Methoden verwendet. In einem zweiten Schritt wurde dann mithilfe einer zweifaktoriellen ANOVA der Einfluss der Faktoren Gruppe (der gleichbedeutend mit der graphischen Repräsentation ist) und Geste sowie ihrer Interaktion untersucht.

Die Variable *Korrekt* wurde mithilfe der Videoaufzeichnungen ermittelt, die mit der Frontkamera des verwendeten Tablets aufgenommen wurden. Alle Gesten der Probanden waren deutlich sichtbar; als Kontrolle dienten außerdem ihre Schlussfolgerungen aus der dargestellten GestureCard. Alternativ hätte das Video auch mit einem Gestenerkenner analysiert werden können, der auf die verwendeten Gesten eingestellt ist. Dieses Vorgehen wäre mit dem Nachteil verbunden gewesen, dass eine Abhängigkeit von der technischen Funktionsfähigkeit des Gestenerkenners in den Aufbau der Studie eingeführt worden wäre, daher wurde stattdessen beschlossen, die Videos selbst auszuwerten. Zur Gewährleistung der Neutralität der Bewertung wurde ein Schiedsrichter[3] hinzugezogen, der die Ausführungen der Probanden ebenfalls auf Korrektheit untersuchte, und die Ergebnisse abgeglichen. Die Ergebnisse von Schiedsrichter und Studienleitung stimmten überein.

[3] Besonderer Dank gebührt Stefan Gries, der diese Aufgabe übernommen hat.

Zur Erfassung der Variable *Replays* wurde durch die Anwendung gezählt, wie oft die Probanden je Geste die Animation neu starteten. Um die Variable *Zuversicht* zu erfassen, wurde eine Likert-Skala in die Anwendung integriert, die die Probanden zwingend verwenden mussten, um zur nächsten Geste zu gelangen. Für die Auswertung wurden die möglichen Werte als Zahlen von 1 (sehr sicher) bis 4 (sehr unsicher) gespeichert. Die Variable *Gesamtzeit* wurde gemessen, indem Anfang und Ende der Bearbeitung jeder Geste von der Anwendung mit einem Zeitstempel versehen wurden. Die beiden Zeitstempel wurden dann subtrahiert, um die Dauer zu ermitteln. Die Daten wurden in Millisekunden erfasst und so auch für die Analyse verwendet, werden aber zur besseren Darstellung nachfolgend zu ganzen Sekunden gerundet.

Jeder Proband bearbeitete die neun Gesten, die in Kap. 12.1.3 beschrieben werden. Bei zehn Probanden wurden so 90 Datensätze, jeweils 45 für die Gruppe Arrows und die Gruppe Hands, zur Auswertung erfasst. Die statistische Auswertung wurde auf ebendiesen Daten durchgeführt, jedoch werden die Daten im Folgenden an verschiedenen Stellen zusammengefasst, um Unterschiede zwischen einzelnen Faktoren herauszustellen. Die Ergebnisse aller Probanden sind detailliert in Anh. E zu finden. Das Signifikanzniveau von 5 % bzw. 1 %, das bereits in den vorigen Studien verwendet wurde, ist auch in dieser Studie wieder die Grundlage. Auch die Einteilung der Effektstärken wird wieder so verwendet, wie Cohen (1988, S. 25f) sie vorgeschlagen hat.

12.2 Ergebnisse

Im Folgenden werden die erfassten Daten statistisch ausgewertet, die mithilfe der in Kap. 12.1.4 beschriebenen Anwendung erfasst wurden. Dazu werden die in Kap. 12.1.5 definierten Variablen und Faktoren mit den in Kap. 12.1.7 beschriebenen Methoden betrachtet.

12.2.1 Korrektheit

Die Probanden konnten die verwendeten Gesten nur korrekt oder nicht korrekt identifizieren. Partielle Lösungen, z. B. die korrekte Wiedergabe der Form, aber mit falscher Ausrichtung im Raum, wurden als nicht korrekt gewertet. Tab. 12.3 zeigt eine Übersicht über die kumulierten Ergebnisse der Probanden für alle Gesten. Zwischen den beiden Gruppen ist nur ein minimaler Unterschied zu sehen. Die Auswertung mit dem χ^2-Test bestätigt diese Beobachtung: Das Ergebnis ist nicht statistisch signifikant ($\chi^2(1, N = 90) = 0,058$; $p = 0,809$). $H_{0Korrekt}$ ist also anzunehmen.

Korrekt	Gruppe Hands	Gruppe Arrows
ja	34	33
nein	11	12
$p = 0,809$		

Tabelle 12.3: Korrekt und nicht korrekt identifzierte Gesten je Gruppe.

Die Analyse mit der zweifaktoriellen ANOVA[4] bestätigt das Ergebnis für den Faktor Gruppe ($F(1,72) = 0,074$; $p = 0,786$; ${\eta_p}^2 = 0,001$). Die Faktoren Geste ($F(8,72) = 2,519$; $p = 0,018$; ${\eta_p}^2 = 0,219$) sowie die Interaktion der Faktoren Geste und Gruppe hingegen zeigen einen Unterschied ($F(8,72) = 2,741$; $p = 0,011$; ${\eta_p}^2 = 0,233$), und zwar beide mit einer großen Effekstärke (${\eta_p}^2 > 0,14$). Tab. 12.4 summiert die korrekten Antworten je Geste und Gruppe auf.

[4] Zwar scheint auf den ersten Blick das notwendige Skalenniveau für eine Varianzanalyse nicht gegeben, aber nach Lunney (1970) können dichotome, abhängige Variablen mit einer ANOVA analysiert werden.

Geste	Gruppe Hands	Gruppe Arrows
Boxing	3	1
Conductor	2	5
Gamma	4	5
U	4	3
Phi	5	5
Present	4	1
X	3	5
Drag'n'Drop	5	3
Spread	4	5
Summe	34	33

Tabelle 12.4: Summe der korrekt identifizierten Gesten je Gruppe und Geste.

12.2.2 Zuversicht

Die Probanden waren in beiden Gruppen vorwiegend sicher und sehr sicher, die dargestellten Gesten korrekt identifiziert zu haben. Abb. 12.11 zeigt eine Übersicht, die auf den ersten Blick suggeriert, dass die Probanden der Gruppe Hands zuversichtlicher waren als die Probanden der Gruppe Arrows: In den Kategorien *Sicher* und *Sehr sicher* überwiegen die Probanden der Gruppe Hands, während es in den Kategorien *Unsicher* und *Sehr Unsicher* gegenteilig aussieht. Ein Mann-Whitney-U-Test zeigt aber, dass keine statistische Signifikanz vorliegt ($U = 864,0$; $p = 0,196$), sodass die Nullhypothese

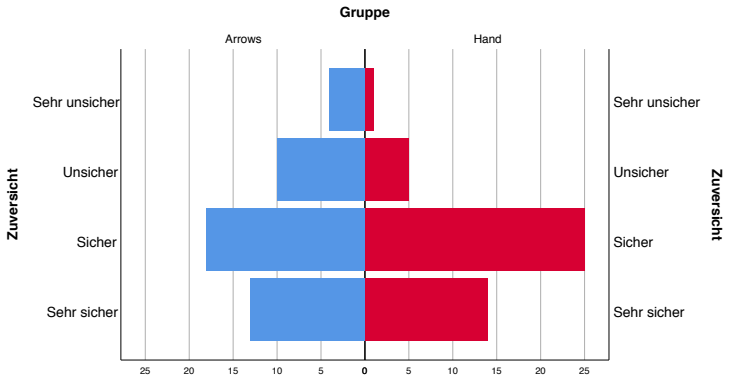


Abbildung 12.11: Vergleich der Zuversicht der Gruppen Arrows und Hands über alle Gesten

$H_{0Zuversicht}$ angenommen werden muss. Somit kann keine statistisch belastbare Aussage über die Zuversicht der beiden Gruppen gemacht werden.

Die Untersuchung mit der zweifaktoriellen ANOVA bestätigt dieses Ergebnis für den Faktor Gruppe ($F(1,72) = 3,512$; $p = 0,065$; ${\eta_p}^2 = 0,047$) sowie den Faktor Geste ($F(8,72) = 1,415$; $p = 0,205$; ${\eta_p}^2 = 0,136$): Beide sind statistisch nicht signifikant beim erwarteten Signifikanzniveau. Beide Faktoren interagieren jedoch miteinander und es wird ein Unterschied sichtbar ($F(8,72) = 6,146$; $p < 0,001$; ${\eta_p}^2 = 0,406$), der auch eine große Effektstärke aufweisen kann (${\eta_p}^2 > 0,14$).

12.2.3 Dauer

Probanden der Gruppe Arrows ($M = 22,66\ s$; 95 % CI $[19,39\ s; 25,92\ s]$) benötigten im Durchschnitt etwas länger als Probanden der Gruppe Hands ($M = 21,03\ s$; 95 % CI $[17,41\ s; 24,67\ s]$), um eine Geste zu bearbeiten. Abb. 12.12 stellt gegenüber, wie lange die Probanden der beiden Gruppen zur Lösung der einzelnen Gesten benötigten. Die graphische Darstellung zeigt, dass bei den Probanden der Gruppe Arrows die Bearbeitungszeit aller Gesten nicht so stark schwankt wie bei der Gruppe Hands: Zwar benötigten Sie im Schnitt länger, aber sowohl die Antennen als auch die beiden Quartile liegen näher beieinander. Ein t-Test zeigt, dass das Ergebnis statistisch nicht signifikant ist ($t(88) = -0,667$; $p = 0,506$). Die zugrundeliegende Nullhypothese H_{0Dauer} muss daher angenommen werden.

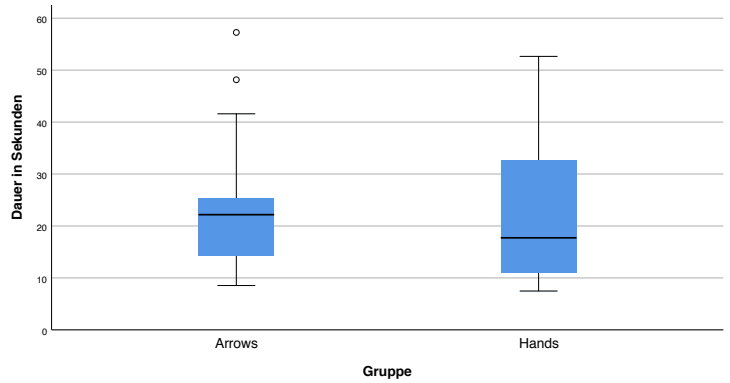


Abbildung 12.12: Vergleich der Dauer aller Gesten beider Gruppen

Die zweifaktorielle ANOVA bestätigt auch dieses Ergebnis für den Faktor Gruppe ($F(1,72) = 0,745$; $p = 0,391$; ${\eta_p}^2 = 0,01$), offenbart jedoch Unterschiede sowohl für den Faktor Geste ($F(8,72) = 4,348$; $p < 0,001$; ${\eta_p}^2 = 0,326$), als auch für die Interaktion zwischen den Faktoren ($F(8,72) = 5,048$; $p < 0,001$; ${\eta_p}^2 = 0,359$). In beiden Fällen ist eine große Effektstärke zu beobachten (${\eta_p}^2 > 0,14$). Tab. 12.5 zeigt die Unterschiede je Geste und Gruppe im Detail. Auffällig ist, dass es bei einigen Gesten im Durchschnitt wenig Unterschied zwischen den Gruppen gibt: Bei den Gesten U, Phi, Drag'n'Drop und Spread beträgt der durchschnittliche Unterschied zwischen die Gruppen maximal vier Sekunden; bei der Geste Phi sind die Gruppen sogar im Schnitt gleich schnell gewesen. Bei den anderen Gesten ist der durchschnittliche Unterschied zwischen den Probanden beider Gruppen mit zehn und mehr Sekunden deutlicher ausgeprägt, wobei je nach Geste mal die Gruppe Arrows und mal die Gruppe Hands schneller war. Fasst man alle Daten zusammen, heben sich die Unterschiede aber gegenseitig auf.

Geste	Gruppe Hands	Gruppe Arrows
Boxing	11	36
Conductor	30	18
Gamma	28	14
U	16	20
Phi	15	15
Present	11	27
X	34	24
Drag'n'Drop	30	33
Spread	15	16
Durchschnitt	21	23

Tabelle 12.5: Durchschnittliche Dauer der Bearbeitung einer Geste je Gruppe in Sekunden.

12.2.4 Replays

Im letzten Schritt wurde analysiert, wie oft die Probanden beider Gruppen die Animation der räumlichen Gesten erneut abspielten. Grundlage der Analyse ist, wie oft sie den dafür vorgesehenen Button betätigten. Die Probanden der Gruppe Hands ($M = 1,31$; 95 % CI $[0,86\ s; 1,76\ s]$) wiederholten die Animation häufiger als Probanden der Gruppe Arrows ($M = 0,24$; 95 % CI $[0,11; 0,38]$). Der Unterschied ist statistisch signifikant (t-Test; $t(88) = 4,611$; $p < 0,001$; $d = 0,97$) mit einer großen Effektstärke ($d > 0,8$). Die Nullhypothese $H_{0Replays}$ muss folglich abgelehnt werden.

Das Ergebnis findet sich auch analog in der zweifaktoriellen ANOVA. Erwartungsgemäß zeigt der Faktor Gruppe einen statistisch signifikanten Unterschied ($F(1,72) = 30,926$; $p < 0,001$; ${\eta_p}^2 = 0,3$). Auch der Faktor Geste zeigt Unterschiede ($F(8,72) = 2,953$; $p = 0,007$; ${\eta_p}^2 = 0,247$), ebenso wie die Interaktion der beiden Faktoren ($F(8,72) = 4,047$; $p = 0,001$; ${\eta_p}^2 = 0,31$). Allen gemein ist eine große Effektstärke (${\eta_p}^2 > 0,14$), die bei der Interaktion der Faktoren am größten ausgeprägt ist. Tab. 12.6 macht diese Unterschiede deutlich. Die Gruppe Hands wiederholte die Animationen für fast alle Gesten – mit Ausnahme der Geste Boxing – häufiger als die Gruppe Arrows. Außerdem wurden einige Gesten häufiger wiederholt als andere.

Geste	Gruppe Hands	Gruppe Arrows
Boxing	2	4
Conductor	14	1
Gamma	15	1
U	4	1
Phi	4	0
Present	2	1
X	11	0
Drag'n'Drop	4	2
Spread	3	1
Summe	59	11

Tabelle 12.6: Anzahl der Replays je Gruppe und Geste.

12.3 Diskussion

Die statistische Auswertung zeigt, dass die Zugehörigkeit zu einer bestimmten Gruppe wenig Einfluss auf das Gesamtergebnis hatte. Mit Blick auf die Korrektheit, Zuversicht und durchschnittliche Dauer der Bearbeitung einer Geste konnte kein Unterschied zwischen beiden Gruppen festgestellt werden. Der einzige messbare Unterschied zwischen beiden Gruppen ist, dass die Probanden der Gruppe Hands häufiger die Animation der Geste wiederholten. Warum das der Fall ist, lässt sich an dieser Stelle nicht zweifelsfrei beantworten, sondern lediglich vermuten. Ein denkbarer Grund ist, dass sich die

Animationen von Händen und Pfeilen in einem Detail unterschieden: die Sichtbarkeit des Bewegungspfades nach Ende der Animation. Die Animation der Pfeile verlängerte den Pfeil über Zeit, d. h. am Ende der Animation war der gesamte Verlauf Geste sichtbar. Bei der Animation der Hände hingegen war das nicht der Fall und die Probanden waren dadurch womöglich gezwungen, sich die Animation häufiger anzuschauen, um sich den Verlauf einprägen zu können. Hier wäre eine mögliche Option, bei der Verwendung eines animierten Handmodells auch den Pfad darzustellen, z. B. mithilfe einer dünnen Linie. Interessant wäre an dieser Stelle noch, zu untersuchen, ob sich die Verwendung animierter und statischer Pfeile unterscheidet. Vielleicht ist eine einmalige Animation, die in die statische Darstellung mündet, von Vorteil.

Das Gesamtergebnis der beiden Gruppen wird allerdings durch die Auswahl der Gesten beeinflusst. Betrachtet man die Ergebnisse einzelner Gesten, so wird deutlich, dass sich die beiden Gruppen sehr wohl unterscheiden, jedoch heben sich die Einzelergebnisse gegenseitig auf, sodass das Gesamtergebnis ausgeglichen erscheint. Das wird insbesondere bei der Messung von Korrektheit, Dauer und der Anzahl der Replays deutlich, bei denen die Analyse mit der ANOVA einen Unterschied zwischen den Gesten zeigt. Auffällig ist, dass insbesondere bei Gesten, die einen komplizierteren Bewegungspfad haben (X, Gamma und Conductor), die Gruppe Arrows im Vorteil ist. Scheinbar gibt es Gesten, die mit Pfeilen besser beschrieben werden als mit einem Handmodell. Dem steht entgegen, dass z. B. die Geste Phi, bei der kein Unterschied festgestellt werden konnte, ebenfalls einen komplizierten Bewegungspfad hat, der ähnlich der Gamma aufgebaut ist. Hier ist eine detailliertere Studie notwendig, die die Unterschiede explizit herausarbeitet und bei der sich die Auswahl der Gesten an denkbaren Faktoren orientiert.

Diese Schlussfolgerung wird auch die durch Prüfung der Interaktion beider Faktoren in der ANOVA bestätigt, die für alle Variablen hier einen Unterschied zeigt. Das deutet darauf hin, dass tatsächlich manche Gesten besser mit dem Handmodell und andere besser mit animierten Pfeilen beschrieben werden. Entsprechend muss bei der Beschreibung räumlicher Gesten eine passende graphische Repräsentation ausgewählt werden. Interessant wären an dieser Stelle die Parameter, mit deren Hilfe entschieden werden kann, ob ein animiertes Handmodell oder animierte Pfeile passender sind. Diese können in zukünftigen Studien ermittelt werden. Denkbar ist auch, dass die Parameter ausgewertet und die passende Repräsentation von der Software ausgewählt wird, mit der die GestureCards für die definierten Gesten erstellt werden.

Die eingangs aufgestellte Forschungsfrage – welche graphische Repräsentation besser zur Darstellung räumlicher Gesten geeignet ist – lässt sich also nicht beantworten. Die Ergebnisse legen nahe, dass es auf die jeweilige Geste ankommt. Somit muss eine Notation zur Beschreibung räumlicher Gesten möglicherweise beide Repräsentationen unterstützen, um möglichst viele Gesten bestmöglich darzustellen.

Bei der Geste Boxing wurde außerdem entgegen der Definition von GestureCards eine zeitliche Verzögerung in die Animation integriert. Eigentlich sollten zeitliche Relationen von Teilgesten ausschließlich über die definierten Mittel von GestureCards dargestellt werden (vgl. Kap. 7.2.2.1). Das Ergebnis

dieser Geste soll daher explizit betrachtet werden und es wird schnell ersichtlich, dass die Gruppe Hands mit Blick auf alle erfassten Variablen im Vorteil war. Diese Art von Geste ist mit animierten Pfeilen nicht gut darstellbar. Die Kommentare der Probanden betonen dieses Ergebnis, denn diese Geste war für alle Probanden der Gruppe Arrows sehr schwierig – mit der Konsequenz, dass nur ein Proband die Geste korrekt identifizieren konnte. Proband wblk z. B. macht das auch deutlich mit den Worten „[Ich] hab das Gefühl, das ich genau hingucken muss. Wenn da jemand schludert, merkt er es nicht“. Die Probanden der Gruppe Hands waren zwar im Schnitt erfolgreicher, schneller und zuversichtlicher als die Probanden der Gruppe Arrows, machten aber ebenfalls Fehler. So erkannte Proband nyyc die Geste sehr zügig, dichtete aber einen weiteren Schlag hinzu. Generell erkannten sie aber die zeitliche Relation besser als die Probanden der Gruppe Arrows.

12.4 Risiken für die Validität

Diese Studie teilt einige Risiken für die Validität mit den beiden Studien, die in Kap. 8 und Kap. 9 beschrieben werden: Die Probanden waren vorwiegend männlich und es wurden individuelle, anonymisierte Gesten eingesetzt. Der Aufbau dieser Studie orientiert sich im Wesentlichen am Aufbau der Studie in Kap. 8, daher ist davon auszugehen, dass die Risiken für die Validität auch in dieser Studie weitgehend identisch sind.

Hervorgehoben werden muss an dieser Stelle der eventuelle Einfluss der gewählten Gesten. Wie in Kap. 4.2.2 dargelegt gibt es für räumliche Gesten – im Gegensatz zu Oberflächengesten (vgl. Kap. 4.2.1) – nur eine kleine Zahl typischer Gesten. Aktuell sind auch Anwendungen, die auf räumlichen Gesten basieren, nicht sehr weit in den Alltag integriert, folglich ist davon auszugehen, dass die Probanden – im Gegensatz zu Oberflächengesten – wenig Erfahrung bzw. Vorprägung bzgl. räumlicher Gesten haben. Somit sind die Vorbedingungen bei dieser Studie anders als bei der ähnlichen Studie, die in Kap. 8 beschrieben wird. Es ist unklar, wie sich diese beiden Faktoren auf die Probanden und die Studie auswirkten. Eine Folge daraus ist, dass auch der Einfluss der gewählten Gesten unklar bleibt.

Die Studie legte den Fokus auf die graphische Repräsentation, daher wurde auf die Definition einiger Eigenschaften verzichtet. So wurde bei keiner Geste die Eigenschaft `Tool` gesetzt, die aber notwendig gewesen wäre, um festzulegen, ob die rechte oder linke Hand verwendet werden soll. Das war zwar bei den definierten Gesten nicht relevant, jedoch impliziert die Verwendung eines Handmodells, dass eine und vor allem welche Hand verwendet wird. Es ist unklar, inwiefern dieser Umstand die Probanden beeinflusst hat.

Teil V

Epilog

13 Weitere Anwendungsgebiete für GestureCards

Der in dieser Arbeit vorgestellte Lösungsansatz lässt sich neben der Spezifikation von Gesten zur Kommunikation zwischen Teammitgliedern auch für andere Zwecke verwenden. Im Folgenden sollen Szenarien vorgestellt werden, in denen GestureCards eingesetzt werden können. Dabei wird auf die unterschiedlichen Eigenheiten der einzelnen Szenarien eingegangen sowie darauf, wie diese mit GestureCards gelöst werden.

Zunächst wird der Aufbau eines Gestenerkenners vorgestellt, der mit GestureCards konfiguriert werden kann (Kap. 13.1). Dann wird gezeigt, wie GestureCards im Entwicklungsprozess zur Prüfung der definierten Interaktionsmuster verwendet werden können (Kap. 13.2). In eine ähnliche Richtung geht auch die Automatisierung von UI-Tests, die nachfolgend für gestenbasierte Anwendungen thematisiert wird (Kap. 13.3), und die Konfiguration bestimmter Gestenerkenner mithilfe von Crowdsourcing (Kap. 13.4). Abschließend soll noch betrachtet werden, wie GestureCards in Bug Reports eingesetzt werden können (Kap. 13.5).

13.1 Gestenerkennung[1]

[1] Eine Zusammenfassung dieses Abschnitts sowie die Validierung einer beispielhaften Implementierung wurde unter Hesenius et al. (2017) publiziert.

Zur Implementierung einer gestenbasierten Anwendung müssen eine Reihe technischer Entscheidungen getroffen werden (vgl. Kap. 4). Eine notwendige Komponente ist ein passender Gestenerkenner (vgl. Kap. 4.3.2), der in den einströmenden Eingabedaten die ausgeführte Geste identifiziert. Es liegt nahe, dass eine derartige Komponente auch direkt mithilfe einer Gestennotation konfiguriert wird, um zu definieren, welche Gesten erkannt werden sollen. Im Folgenden soll daher ein Konzept für einen Gestenerkenner dargelegt werden, der GestureCards zur Konfiguration nutzt.

GestureCards basieren auf der Idee, möglichst viele Eigenschaften explizit zu beschreiben. Je mehr Eigenschaften definiert werden, desto genauer ist die Geste beschrieben. Anders ausgedrückt wird mit Eigenschaften auch die Menge der gültigen Gesten eingeschränkt. Eine GestureCard, die eine linienförmige Geste nach rechts definiert, beschreibt alle denkbaren linienförmigen Gesten nach rechts. Wird jedoch die Eigenschaft `Speed: Fast` hinzugefügt, so sind nur noch alle linienförmigen Gesten nach rechts definiert, die schnell ausgeführt werden. Mit weiteren Eigenschaften kann die Geste dann noch genauer eingegrenzt werden: Wird z. B. noch `Location: Object` hinzugefügt, so muss die Geste auf einem virtuellen Objekt ausgeführt werden. Genau hier liegt auch der Schlüssel, um einen Gestenerkenner für GestureCards zu entwickeln.

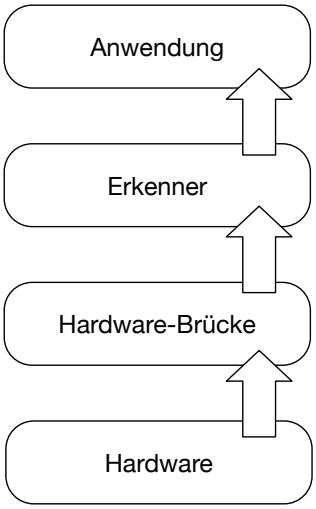


Abbildung 13.1: Datenfluss eines Gestenerkenners für GestureCards

In Kap. 4.3 wurde die typische Architektur gestenbasierter Anwendungen vorgestellt, die als Grundlage für die notwendigen Entscheidungen innerhalb der Entwicklung dient. Diese definiert mehrere Schichten, bei denen Daten von der verwendeten Hardware hin zur Anwendung fließen. Die Erkennung ausgeführter Gesten findet dabei in der Interpretationsschicht statt, die eine Ebene unter der eigentlichen Applikation angesiedelt ist. Ein Gestenerkenner für GestureCards sollte den in Abb. 13.1 dargestellten Datenfluss umsetzen, der letztlich auf der genannten Architektur basiert und die wichtigsten Komponenten definiert. Die Komponente *Hardware-Brücke* übernimmt dabei teilweise die Aufgaben der Hardware-Abstraktionsschicht und insbesondere der Transformationsschicht. Sie kapselt die von den Hardware-Herstellern zur Verfügung gestellten Treiber und Komponenten aus SDKs.

Abb. 13.2 zeigt die grundlegende Architektur eines Gestenerkenners für GestureCards. Das Prinzip ist, dass für jede GestureCard ein Gestenerkenner dynamisch aus mehreren Komponenten zusammengesteckt wird. Jede dieser Komponenten ist spezialisiert auf die Erkennung einer bestimmten Eigenschaft und wird mit den in der GestureCard definierten Werten konfiguriert. Somit wird genau genommen nicht *ein* Gestenerkenner verwendet, sondern *mehrere*, die in Abhängigkeit der jeweiligen GestureCard individuell zu einem dedizierten Erkenner kombiniert werden. Jede GestureCard, die erkannt werden soll, wird durch eine Software-Komponente repräsentiert (die in Abb. 13.2 ganz rechts dargestellt ist), die die speziellen Erkenner für die notwendigen Eigenschaften bündelt. Dieser Ansatz basiert auf einem grundlegenden Prinzip des SE – Trennung der Anliegen – und ermöglicht sowohl größtmögliche Flexibilität als auch einfache Optimierung der Performanz, da gezielt auf die Schwachstellen der spezialisierten Erkenner für Eigenschaften eingegangen werden kann.

Dafür ist zunächst eine Komponente notwendig, die die Unabhängigkeit von Hardware und Sensoren sicherstellt (vgl. Kap. 4.3). Diese Komponente – eingangs bereits als *Hardware-Brücke* bezeichnet – übersetzt herstellerspezifische Daten in ein allgemeines Datenformat. Dabei werden auch Informationen vereinheitlicht, die zwischen Herstellern variieren (vgl. z. B. Kap. 5.1.6). Mehrere Hardware-Brücken sind denkbar, z. B. wenn unterschiedliche Sensoren verwendet werden. Die erfassten Daten müssen zunächst synchronisiert und dann an die verantwortlichen Erkenner für die Eigenschaften übergeben wer-

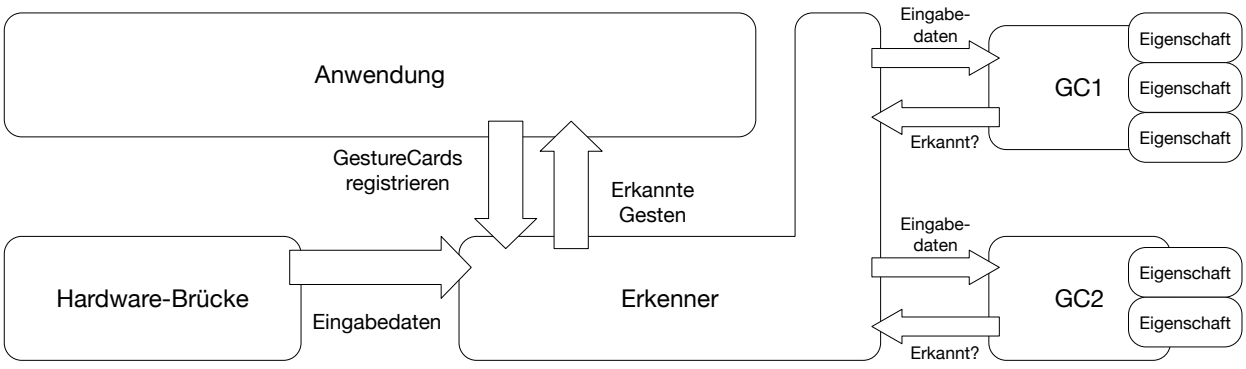


Abbildung 13.2: Überblick über die notwendigen Komponenten eines Gestenerkenners für GestureCards

den. Das zugrundeliegende Problem wurde bereits in Kap. 4.3.2 angesprochen: Die Synchronisierung kann entweder in der Transformations- oder der Interpretationsschicht durchgeführt werden. In der vorgestellten Architektur werden die Daten also entweder in der Hardware-Brücke oder in einer der nachgelagerten Komponenten des Erkenners synchronisiert.

In der Mitte von Abb. 13.2 findet sich die zentrale Komponente, mit der die Anwendung kommuniziert. Diese taucht zwar nach außen als Gestenerkenner auf, verwaltet aber letztlich nur eine Menge registrierter GestureCards und Hardware-Brücken. Die wesentlichen Aufgaben sind: Verwaltung registrierter GestureCards, Verteilung von Eingabedaten an spezialisierte Erkenner, Verwaltung von Hardware-Brücken und Kommunikation mit Komponenten der Anwendung. Der eigentliche Prozess der Erkennung wird an verschiedene, auf bestimmte Eigenschaften spezialisierte Komponenten delegiert.

Jede der in Abb. 13.2 rechts definierten GestureCards ist somit im Wesentlichen selbst der notwendige Gestenerkenner. Dazu werden bei Registrierung einer GestureCard verschiedene spezialisierte Erkennerkomponenten geladen, deren wesentliche Aufgabe die Identifikation der definierten Eigenschaften ist. Wenn also eine GestureCard erkannt werden soll, dann müssen alle damit verbundenen Eigenschaften erfolgreich erkannt werden – oder anders ausgedrückt: Jeder der zugeordneten Erkenner für Eigenschaften muss *seine* Eigenschaft mit den konfigurierten Einschränkungen erkennen. Schlussendlich ist die erfolgreiche Erkennung der Geste somit eine Konjunktion mehrerer kleiner (erfolgreicher) Erkennungen der Eigenschaften. Während der Erkennung sind alle Eigenschaften gleich gewichtet. Wird eine Eigenschaft nicht erfolgreich erkannt, so ist die ganze GestureCard als nicht erfolgreich erkannt zu behandeln, ungeachtet der Ergebnisse anderer Eigenschaften.

Die verschiedenen Eigenschaften sind unterschiedlich kompliziert zu erkennen, entsprechend müssen auch die einzelnen Erkenner unterschiedlich aufgebaut werden. So ist z. B. die Ermittlung der Dauer einer Geste verhältnismäßig einfach: Eine simple Implementierung wäre, die Differenz der Zeitstempel von Start- und Endpunkt der Geste zu berechnen; ein Erkenner für die Dauer würde also letztlich nur eine Subtraktion durchführen.[2] Die Erkennung der Form hingegen ist deutlich aufwendiger, jedoch kann hierfür z. B. ein bereits vorhandener Algorithmus (vgl. Kap. 6.1) wiederverwendet[3] werden.

Die grundlegende Architektur, die im Wesentlichen auf der Trennung in unterschiedliche, alleinstehende Komponenten aufbaut, ermöglicht problemlos den Austausch und die Optimierung einzelner Erkenner für Eigenschaften. So können z. B. unterschiedliche Algorithmen zur Erkennung von Formen eingesetzt und je nach Notwendigkeit geladen werden, denn manche Algorithmen können besser mit verschiedenen zeitlichen Relationen umgehen als andere. Die Architektur ermöglicht außerdem noch weitere Optimierungen, wie die parallele Verarbeitung unterschiedlicher Eigenschaften. Hierfür ist abzuschätzen, ob die Verwaltung der notwendigen Synchronisierung den zeitlichen Gewinn wert ist.[4]

Eine weitere Möglichkeit, den Prozess der Erkennung zu optimieren, ist, eine passende Reihenfolge der Erkenner für Eigenschaften festzulegen. Das Ziel

[2] Dies wäre allerdings nur für einzügige Gesten so einfach. Für mehrzügige Gesten müssten die Züge der Teilgesten noch addiert werden.

[3] Es ist anzunehmen, dass dennoch die ein oder andere Anpassung am Algorithmus vorgenommen werden muss, um ein reibungsloses Zusammenspiel mit anderen Komponenten zu gewährleisten.

[4] Ungeachtet dessen, dass diese Entscheidung auch in Abhängigkeit der verwendeten Hardware getroffen werden muss.

sollte sein, definierte Gesten möglichst schnell auszuschließen. Somit wäre eine Option, die notwendigen Erkenner nach bekannten Performanzwerten zu sortieren und grundsätzlich mit der schnellsten Komponente zu beginnen. Eigenschaften, deren Erkennung zu viel Performanz kostet, würden somit erst geprüft werden, wenn die performanteren Erkenner Erfolg hatten, andernfalls wird die Geste bereits vorher verworfen.

GestureCards ermöglichen es außerdem, Eigenschaften als Rückgaben für die weitere Verarbeitung zu definieren. Entsprechend muss ein passender Erkenner bei der Meldung der erkannten Geste auch die definierten Rückgaben und dazugehörigen Informationen an die Komponenten der Anwendung weitergeben. Jeder Erkenner für eine Eigenschaft muss dafür eine passende Menge an Informationen definieren. Außerdem muss der Erkenner zurückmelden, wie hoch die Konfidenz für die aktuell erkannte Geste ist, damit die Anwendung ggf. das Ergebnis verwerfen kann. Es kann auch sinnvoll sein, dass der Erkenner einfach eine Liste aller bekannten Gesten sowie ggf. notwendiger Rückgaben an die Anwendung zurückgibt und jede mit einer Konfidenz bewertet. Die finale Entscheidung für oder gegen eine Geste wird dann der Anwendung überlassen.

Ein Gestenerkenner für GestureCards erfordert zwangsläufig die Möglichkeit, GestureCards in einer maschinell verarbeitbaren Form zu definieren und zu speichern. Sowohl ein passendes UI als auch die notwendigen Datenstrukturen wurden für die Integration in den AugIR für Oberflächengesten entwickelt (vgl. Kap. 10.1.3) und können als Basis dienen.

13.2 Prüfung definierter Interaktionsmuster

Definierte Gesten, die in einer Anwendung eingesetzt werden sollen, müssen auf unterschiedliche Faktoren geprüft werden. So muss z. B. sichergestellt sein, dass Benutzer durch die Gesten nicht an ihre körperliche Belastungsgrenze gebracht oder gar verletzt werden. So definieren verschiedene Autoren (z. B. Oulasvirta et al. (2014)) Regeln für die Ergonomie von Gesten. Mithilfe von GestureCards könnten definierte Gesten auf die Einhaltung der Regeln geprüft werden.

Es muss auch verhindert werden, dass Gesten definiert werden, die von Benutzern aufgrund körperlicher Einschränkungen gar nicht ausgeführt werden können. So könnten Entwickler versehentlich Gesten definieren, die anatomisch nicht möglich sind. Ein interessanter Spezialfall könnten Benutzer sein, die körperlich eingeschränkt sind, und daher Anwendungen nur innerhalb gewisser Grenzen bedienen können. Gerade für derartige Fälle könnten spezialisierte Regelwerke helfen, bereits in frühen Phasen potentiell unmögliche Gesten zu identifizieren. Definierte GestureCards müssten dann mit dem Regelwerk abgeglichen werden.

Gestensteuerung ist außerdem ungenau: Benutzer sind nicht in der Lage, definierte Gesten immer gleich auszuführen. Dies hat zur Folge, dass Entwickler eine grundlegende Robustheit gegenüber Abweichungen in der Ausführung in Anwendungen integrieren müssen. Gesten, die sich zu sehr ähneln, sind

sowohl für Benutzer als auch für Entwickler besonders schwierig zu handhaben, da sie die Gefahr einer Fehlerkennung erhöhen. Allerdings lässt sich die Ähnlichkeit von Gesten an verschiedenen Merkmalen festmachen (Long et al., 2000). Gleichzeitig steigt durch die Ungenauigkeit der Ausführung die Verwechslungsgefahr, denn bereits kleinste Abweichungen können zu einer anderen Geste führen, wenn sich definierte Gesten zu sehr ähneln. Das Problem wird bei räumlichen Gesten, bei denen die Unterscheidung bzw. Segmentierung zwischen der eigentlichen Geste und unbewussten Bewegungen der Nutzer nicht so einfach ist wie bei Oberflächengesten (vgl. Kap. 4.1), nochmal verstärkt. Eine Notation wie GestureCards kann helfen, definierte Gesten zu vergleichen und diejenigen zu identifizieren, die ein vorher definiertes Maß an Ähnlichkeit überschreiten.

Außerdem müssen definierten Gesten auch während der Entwicklung einer Anwendung regelmäßig überprüft werden. So müssen bei der Integration neuer Funktionalitäten oder paralleler Entwicklungszweige die Gesten, die hinzugefügt werden sollen, mit den bestehenden verglichen werden, um Ähnlichkeiten, Dopplungen oder unpassende Gesten, die nicht in das Gesamtkonzept passen, auszuschließen. Somit muss ein konstanter Abgleich zwischen den unterschiedlichen Interaktionskonzepten stattfinden, die integriert werden. Außerdem kann sich das Team verändern und muss neue Mitglieder aufnehmen, die sich in die Anwendung einarbeiten[5] und neue Gesten definieren müssen. Neue Team-Mitglieder können unterstützt werden, wenn eine Möglichkeit besteht, existierende und neue Gesten abzugleichen. Ein derartiger Abgleich ließe sich auch automatisieren: So könnten neue Gesten bei jeder Änderung, die in das Versionskontrollsystem eingespielt wird, mit den bestehenden verglichen werden.[6] Einige mögliche Metriken und ein passendes Modell, das zum automatischen Vergleich eingesetzt werden kann, sind bei Long et al. (2000) zu finden. Diese müssen jedoch für die Möglichkeiten, die neue Hardware bietet, überprüft und verfeinert werden, insbesondere zum Einbezug weiterer Eigenschaften. Außerdem wären Erkenntnisse über Varianzen bei der Ausführung von Gesten hilfreich, um zu prüfen, wie sich die gleichen Gesten bei unterschiedlichen Nutzergruppen verhalten. Die Funktionalität könnte noch um die Möglichkeit erweitert werden, die eingangs erwähnten schwierigen oder gar unmöglich auszuführenden Gesten automatisch zu identifizieren.

[5] Auch als *Ramp-up*-Problem bezeichnet (Sim & Holt, 1998).

[6] Ähnliches wird bereits bei Änderungen am Quellcode durchgeführt und ist als *Continuous Integration* bekannt: Sobald dem zentralen Versionskontrollsystem Änderungen hinzugefügt werden, wird eine Menge von Tests aufgerufen und der Entwickler informiert, ob seine Änderungen diese erfolgreich durchlaufen haben.

13.3 Testautomatisierung[7]

Testen ist eine wesentliche Aktivität für Software-Entwickler und nimmt bis zu 90 % des gesamten Entwicklungsaufwands ein (Tassey, 2002). Die frühzeitige Erkennung von Fehlern ist für den wirtschaftlichen Erfolg eines Software-Projekts unerlässlich, insbesondere in umkämpften Marktsegmenten wie dem Markt für mobile Anwendungen. Hier sind kurze Release-Zyklen üblich, außerdem können Benutzer eine alternative App schnell über App-Stores installieren – entsprechend einfach gestaltet sich der Wechsel zu einem Konkurrenten. Software-Entwickler haben auf diese Herausforderungen mit der Einführung agiler und iterativer Entwicklungsprozesse reagiert, in die sich die notwendigen Schritte zur Entwicklung einer guten UX integrieren

[7] Das nachfolgend vorgestellte Konzept baut auf der unter Hesenius et al. (2014a) publizierten Arbeit auf.

lassen. Tests sind dabei in allen Phasen der Software-Entwicklung denkbar und notwendig, an dieser Stelle soll aber der Fokus auf den Test der fertigen Implementierung gelegt werden. Jedes der in den einzelnen Iterationen erzeugten Software-Artefakte muss getestet werden, sowohl in Bezug auf die Korrektheit funktionaler Komponenten als auch auf das implementierte UI.

In Hinblick auf die Interaktion mit der Anwendung ist zu testen, ob die Anwendung auf eine Eingabe des Benutzers korrekt reagiert. Dabei spielen für gestenbasierte Anwendungen verschiedene Faktoren eine Rolle:

- Es muss sichergestellt werden, dass die verwendeten Erkenneralgorithmen in der Lage sind, alle definierten Gesten einwandfrei zu erkennen.
- Es muss geprüft werden, ob die Anwendung wie erwartet auf alle Gesten in unterschiedlichen Szenarien reagiert.
- Ungenauigkeiten in der Gestenausführung durch unterschiedliche Benutzer müssen berücksichtigt werden.

Insbesondere aus Letzterem ergibt sich die Notwendigkeit, Tests mit unterschiedlichen Benutzerprofilen zu wiederholen, um trotz Varianz der Gestenausführung eine korrekte Reaktion sicherzustellen.

Typisch sind oftmals manuelle Tests, d. h. ein Tester arbeitet eine Menge definierter Testszenarien mit der Anwendung ab und prüft, ob nach einer Anzahl von Schritten ein vordefiniertes Ergebnis erreicht wird. Manuelles Testen geht jedoch mit erhöhtem Personalaufwand einher und mit Blick auf die zuvor genannten Faktoren, die zwangsläufig mit Gestensteuerung einhergehen, ist abzusehen, dass sich der Aufwand[8] durch den Einsatz von Gesten nochmal erhöht. Verschiedene Strategien sind denkbar, um den Aufwand zu minimieren: z. B. spezialisierte Dienstleister, Crowdsourcing oder Testautomatisierung. Sowohl bei spezialisierten Dienstleistern als auch beim Crowdsourcing wird jedoch letztlich wieder auf menschliche Arbeitskraft zurückgegriffen. Während Dienstleister entsprechendes Personal zur Verfügung stellen, das die einzelnen Tätigkeiten des Testens übernimmt, werden beim Crowdsourcing einzelne Aufgaben quasi öffentlich ausgeschrieben. Gerade für das Testen hat Crowdsourcing unterschiedliche Vorteile, insbesondere jedoch die Möglichkeit, unterschiedliche Qualifikationslevel und Spezialisierungen zu integrieren.[9] Für gestenbasierte Anwendungen hat Crowdsourcing für Tests insbesondere den Vorteil, dass viele unterschiedliche Ausführungen der Gesten gesammelt werden. Gleichzeitig sind Tester ebenso wie Entwickler stark abhängig von dem in frühen Phasen entwickelten Interaktionskonzept, da hier die für die Anwendung relevanten Gesten definiert werden. Die Gesten müssen folglich den Testern kommuniziert werden – eine Notation wie GestureCards ist auch in diesem Fall hilfreich.

Der personelle Aufwand kann mit Techniken zur Testautomatisierung minimiert werden: Die grundlegende Idee ist, eine Menge von Testfällen zu definieren, die beliebig oft maschinell wiederholt werden können. Werden Tests automatisiert, so müssen Eingabeereignisse, die sich aus Eingaben der Benutzer ergeben, simuliert und in die Anwendung injiziert werden. Ein

[8] Haller (2013) berichtet darüber hinaus über weitere Faktoren wie das Testen auf mehreren Geräten (*Device Tests*) und im tatsächlichen Einsatzkontext (*Testing in the Wild*), die z. B. das Testen mobiler Anwendungen erschweren.

[9] Mao et al. (2017) geben einen Überblick über Crowdsourcing für SE im Allgemeinen und gehen dabei speziell auf das Testen ein.

Testfall beschreibt somit eine Menge automatisierter Eingaben in das UI und den nachfolgend erwarteten Zustand. Bei klassischen WIMP-UIs muss z. B. definiert werden, welche Informationen in Textfelder eingegeben werden und auf welchen Button geklickt wird. Für gestenbasierte Anwendungen muss entsprechend definiert werden, welche Geste wie ausgeführt wird.

Somit existieren beim automatisierten Testen gestenbasierter Anwendungen zwei wesentliche Herausforderungen: Gesten müssen exakt beschrieben und zuverlässig reproduziert werden können, idealerweise mit parametrisierbarer Gestenvarianz. Während im Rahmen dieser Arbeit gezeigt wurde, wie Gesten beschrieben werden können, stellt die Integration von Gestenvarianz eine neue Herausforderung dar. Bestehende Tools zur Testautomatisierung gestenbasierter Anwendungen (z. B. Khandkar et al. (2010)) können jedoch allenfalls einmal aufgezeichnete Gesten wieder abspielen – eine Parametrisierung der Ausführung ist nicht möglich. Die aufgezeichnete Geste ist aber von unterschiedlichen Faktoren wie dem verwendeten Gerät, den motorischen Fähigkeiten des Aufzeichnenden oder auch dem verwendeten Finger abhängig, wodurch die Generalisierbarkeit des Tests in Frage gestellt werden muss.

Ein weiteres Problem bei der Automatisierung von Tests ist, dass die vorhanden Tools und Frameworks oftmals ein technisches Verständnis voraussetzen. Insbesondere Akzeptanztests, bei denen lediglich das UI der Anwendung bedient wird, werden aber oftmals von Spezialisten aus dem jeweiligen Fachbereich definiert, die über das eventuell notwendige technische Wissen nicht verfügen. Darüber hinaus können viele Informationen, die in frühen Phasen erfasst wurden, zur Erstellung von Testfällen eingesetzt werden. Dies setzt jedoch ein gemeinsames Verständnis von fachlichen Inhalten und zu testenden Konzepten voraus. Techniken wie Behavior-Driven Development (BDD)[10] versuchen, eine Brücke zwischen gemeinsamen Verständnis und Testautomatisierung zu schlagen. In einfachen Worten ist das Ziel, Testszenarien zu definieren, die sowohl menschen- als auch maschinenlesbar sind: Sie können somit sowohl von fachlichen Experten definiert als auch automatisch ausgeführt werden. Unterschiedliche Frameworks existieren, die BDD für klassische WIMP-UIs, Webseiten oder auch mobile Geräte anbieten. Letztere stellen jedoch – wenn überhaupt – nur einen grundlegenden Satz an Gesten zur Verfügung – so unterstützt z. B. das Framework Calabash sechs vorgefertigte Gesten (Xamarin Developer, 2016), die größtenteils aus dem De-Facto-Standard für Multi-Touch-Gesten (vgl. Kap. 4.2) stammen.

[10] Wynne & Hellesoy (2012) führen in das Konzept und die zugrundeliegenden Ideen ein.

Notwendig ist ein *Gestensimulator*, der Beschreibungen von Gesten in die zugrundeliegenden Eingabedaten übersetzt und in die Anwendung zu Testzwecken injiziert. Gleichzeitig müssen unterschiedliche Varianzen der Ausführung definiert werden können. GestureCards können hier als Startpunkt dienen, um die zu testenden Gesten nicht nur zu definieren, sondern auch unterschiedliche Ausführungen zu beschreiben, z. B. durch Anpassung verschiedener Eigenschaften. Außerdem ist die Integration von Automatismen erstrebenswert, die, aufbauend auf GestureCards, Gesten mit unterschiedlicher Varianz ausführen. Somit ließen sich z. B. Benutzerprofile hinterlegen und die Anwendung unter Berücksichtigung unterschiedlicher Ausführungseigenschaften testen. Ein Testfall könnte so definiert werden, dass das Testautoma-

tisierungssystem die Anwendung wie ein Senior bedient, dessen motorische Fähigkeiten eingeschränkt sind und der daher Gesten weniger genau ausführt als z. B. ein simulierter jugendlicher Nutzer.

Im Folgenden soll konzeptuell dargelegt werden, wie GestureCards in Testszenarien für BDD-Frameworks integriert werden können. Die Grundlage hierfür ist eine eigene Erweiterung des o. g. Frameworks Calabash um die Möglichkeit, Gesten mit GeForMT zu beschreiben und auszuführen (Hesenius et al., 2014a). Notwendig ist dafür die Integration von GestureCards in Gherkin, der DSL, die in Calabash zur Formulierung von Testfällen verwendet wird, sowie ein Gestensimulator, der GestureCards in die notwendigen Daten für tatsächliche Eingaben auf einem Gerät übersetzt.

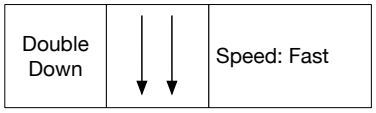


Abbildung 13.3: Die in Abb. 13.4 verwendete Geste.

Die Integration in Gherkin kann über den Namen einer Geste erfolgen, der innerhalb der definierten Tests eindeutig sein sollte. In Abb. 13.4 ist ein Beispiel eines Testfalls zu sehen, der mit Gherkin formuliert wurde. In Zeile 1 wird der Testfall benannt und in den Zeilen 2–5 werden einige Vorbedingungen geprüft. In Zeile 6 wird dann die in Abb. 13.3 gezeigte Geste mithilfe des Gestensimulators ausgeführt. Dazu werden die Schlüsselwörter `perform gesture` verwendet und der Name der in Abb. 13.3 dargestellten Geste angehangen. Zur Definition der Geste ist ein passendes UI notwendig; ggf. kann die für die Konfiguration eines Gestenerkenners notwendige Oberfläche (vgl. Kap. 13.1) mit testspezifischen Informationen wiederverwendet werden. In den Zeilen 7-8 wird dann die erwartete Reaktion auf die Geste definiert.

Abbildung 13.4: Ein potentielles BDD-Feature mit integrierter GestureCard

```
1 Scenario: Activation of push services
2 Given my app is running
3 Then the "GameMain" activity should be open
4 Then the view "push-indicator" should
5   not be visible
6 When I perform gesture "DoubleDown"
7 Then the view "push-indicator" should
8   be visible
```

Ein passender Gestensimulator, der definierte GestureCards in ausgeführte Gesten übersetzt, muss zwei wesentliche Aufgaben übernehmen:

- Eine Menge grundlegender Eingabeereignisse erzeugen, die der definierten Geste mit allen Eigenschaften entsprechen
- Varianzen bei der Ausführung hinzufügen

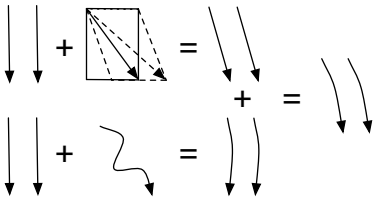


Abbildung 13.5: Eine Geste mit hinzugefügter Varianz

Ein potentieller Simulator könnte in den in Abb. 13.6 dargestellten Phasen arbeiten. Zunächst wird eine prototypische Geste in einem virtuellen Koordinatensystem generiert, die der Form der definierten Geste entspricht. Das Ergebnis dieses Schritts ist eine Menge von Eingabeereignissen, die dem Prototypen der Geste entsprechen. Um die Geste aus Abb. 13.3 abzubilden, würde der Prototyp zwei perfekten Linien nach unten entsprechen. Anschließend werden die zur Geste definierten Eigenschaften eingearbeitet, indem

die Menge der Eingabeereignisse passend manipuliert wird. Wenn wie im Beispiel die Eigenschaft `Speed: Fast` gesetzt ist, so wird sich der Abstand der einzelnen Eingabeereignisse voneinander erhöhen. In der letzten Phase werden dann Varianzen bei der Ausführung hinzugefügt. Denkbar ist z. B., dass Benutzer Gesten nicht ganz senkrecht, sondern etwas schief ausführen, oder eine leichte Kurve dabei ziehen. Die berechnete Geste muss in Abhängigkeit dieser (und vielleicht weiterer Faktoren) angepasst werden, wie in Abb. 13.5 dargestellt. Die unterschiedlichen Optionen könnten zu Benutzerprofilen zusammengefasst und der Test dann mehrfach, aber für unterschiedliche Nutzergruppen ausgeführt werden. Um möglichst viele Informationen über denkbare Gestenausführungen zu sammeln, müssten im Rahmen empirischer Untersuchungen die Abweichungen ermittelt werden, die bestimmte Nutzergruppen von anderen unterscheiden.

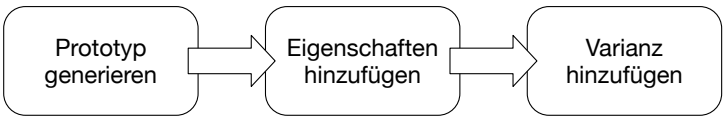


Abbildung 13.6: Arbeitsschritte eines potentiellen Gestensimulators

13.4 Training von Template- und ML-basierten Gestenerkennern

Template-basierte Algorithmen zur Gestenerkennung (vgl. Kap. 6.1) benötigen z. T. mehrere Ausführungen einer Geste als Vorlagen, um bessere Ergebnisse zu erzielen. Dasselbe gilt für Gestenerkenner, die auf Techniken des Machine Learning (ML) basieren, die aber letztlich auch nach bekannten Mustern in unbekannten Daten suchen. Die bekannten Muster müssen dem Erkenner vorab zur Verfügung gestellt werden; somit lässt sich ein ML-basierter Erkenner auch als Template-basiert klassifizieren. Das entsprechende Training kann auf zweierlei Arten umgesetzt werden: durch automatisierte Verfahren, die unterschiedliche Ausführungen der Geste generieren, oder durch Sammlung von Ausführungen unterschiedlicher Benutzer. Beide setzen voraus, dass die Gesten definiert werden.

Fothergill et al. (2012) prüfen, wie die Gesten aufbereitet werden müssen, damit Probanden Gestenerkenner trainieren können. Sie zeigen, dass zwar unterschiedliche Medien (Video, Bilder, Text) eingesetzt werden können, aber die Kombination unterschiedlicher Beschreibungsarten (z. B. Video und Text oder Bilder und Text) zu einem höheren Verständnis der Probanden führt als rein textuelle oder rein graphische Notationen. Somit konnten mit der passenden Beschreibung mehr korrekte Gesten von den Probanden durchgeführt werden und so mehr Varianzen gesammelt werden, was letztlich zu robusteren Gestenerkennern führt. GestureCards als hybride Notation können hier also ihre Vorteile ausspielen.

Amini & Li (2013) lagern das Training von Erkennern in die Crowd aus und ermöglichen so, auf einfachem Wege möglichst viele Probanden zu erreichen. Entwickler können notwendige Gesten – die Autoren fokussieren mobile Geräte – definieren und veröffentlichen. Nutzer der Anwendung führen dann

die Gesten aus, sodass möglichst viele Varianten gesammelt werden. Auch hier stellt sich die Frage, wie Gesten kommuniziert und dokumentiert werden. Notationen wie GestureCards könnten hier eingesetzt werden.

Gleichzeitig könnte das Training von Gestenerkennern mithilfe von GestureCards auch automatisiert werden. Das in Kap. 13.3 vorgestellte Konzept eines Gestensimulators kann auch verwendet werden, um unterschiedliche Ausführungen der gleichen Geste zu generieren. Diese können dann als Trainingsdatensatz verwendet werden. Dabei kann insbesondere auch auf unterschiedliche Varianzen zurückgegriffen werden.

13.5 Bug Reports

Fehler sind in der Software-Entwicklung unvermeidbar und werden häufig erst entdeckt, wenn die Anwendung im Betrieb ist und von echten Benutzern verwendet wird. Benutzer können Fehler häufig in Form von *Bug Reports* an die Entwickler melden, die mithilfe dieser Beschreibung dann den Fehler nachstellen und ggf. beheben können. In einem Bug Report werden unterschiedliche Informationen gesammelt: was der Benutzer getan hat, wie die Anwendung reagierte, welche Reaktion erwartet wurde usw. In gestenbasierten Anwendungen muss somit zwangsläufig die ausgeführte Geste berichtet werden, damit Entwickler die Interaktion des Benutzers nachstellen können. Damit muss letztlich die Frage diskutiert werden, wie Gesten in Bug Reports integriert werden können.

GestureCards können in so einer Situation Hilfestellung bieten. Die einfachste Variante wäre, dass Benutzer, die einen Bug Report verfassen, ihre Gesten direkt in GestureCards formulieren. Sie könnten so analog zum heutigen Vorgehen, bei dem Benutzer beschreiben, welchen Text sie in welches Feld eingegeben haben und auf welchen Button sie dann geklickt haben, auch die ausgeführten Gesten dokumentieren und ggf. in eine Reihenfolge bringen. Benutzer, die zumeist auch noch freiwillig helfen, wären gezwungen, selbst eine detaillierte Analyse der ausgeführten Gesten zu betreiben. Es ist davon auszugehen, dass ein derartiges Vorgehen vom Großteil der Benutzer nicht erwartet werden kann.

Eine Alternative dazu wäre die Möglichkeit, die Interaktion automatisch analysieren zu lassen. Eine passende Software könnte die Eingaben des Benutzers aufzeichnen und in ein entsprechendes Format übersetzen, das dann wiederum vom Entwickler zur Analyse des Fehlers eingesetzt werden kann. Es wäre zur Identifikation der ausgeführten Gesten eine ähnliche Komponente notwendig wie der in Kap. 13.1 beschriebene Gestenerkenner. Anstatt jedoch die Eingaben des Benutzers gegen eine Menge bekannter GestureCards abzugleichen, müsste diese Komponente eine GestureCard generieren.

In Abb. 13.7 ist dargestellt, wie solch ein GestureCard-Generator arbeiten könnte. Dazu müssten Eingabedaten, die sequentiell eintreffen und die notwendigen Informationen über Position usw. enthalten, entgegengenommen und von einer Komponente untersucht werden. Diese wiederum wird in Abb. 13.7 gezeigt und besteht aus spezialisierten Komponenten, die die Ein-

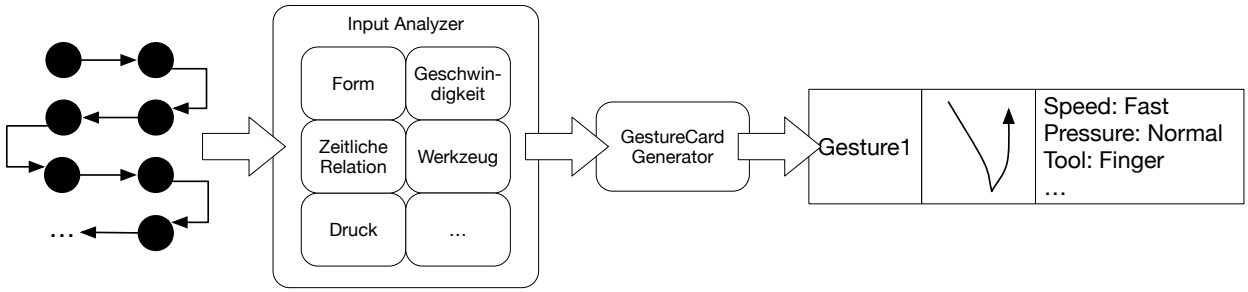


Abbildung 13.7: Grundlegende Funktionsweise eines Generators für GestureCards

gabedaten auf die unterschiedlichen Eigenschaften, die Gesten haben können, untersuchen. Jede Komponente würde im Idealfall exakt eine Eigenschaft untersuchen. Das Ergebnis dieser Analyse wäre eine Datenstruktur, die alle Informationen über die ausgeführte Geste enthält. Im letzten Schritt würde diese Datenstruktur zu einer passenden GestureCard übersetzt werden, die alle Eigenschaften der Ausführung enthält.

Es können jedoch mutmaßlich nicht alle Eigenschaften auf diese Art und Weise erfasst werden. Ein Beispiel ist die Eigenschaft `Reaction`, die angibt, ob die Anwendung auf die Geste bereits zur Laufzeit oder erst nach vollständiger Ausführung reagiert. Somit müsste zunächst untersucht werden, welche Eigenschaften definitiv aus den Eingabedaten abgeleitet werden können, welche eventuell aus anderen Daten interpoliert werden müssen und welche gar nicht automatisch, sondern nur mithilfe des Benutzers erfasst werden können.

Eine weitere in diesem Zusammenhang interessante Frage ist, wie der Bug Report dem Entwickler präsentiert wird. Es ließe sich z. B. aus den Informationen des Benutzers ein Storyboard generieren, das mit GestureCards annotiert wird. Alternativ wären (das Einverständnis des Benutzers und entsprechende Sensoren vorausgesetzt) auch kurze Videosequenzen denkbar, die insbesondere für Anwendungen im Bereich VR/AR interessant sind.

14 Fazit und Ausblick

Die Entwicklung gestenbasierter Anwendungen erfordert die Zusammenarbeit verschiedener Spezialisten sowie den Austausch und die Kommunikation relevanter Merkmale der geplanten Interaktion. Im Laufe der Entwicklung müssen unterschiedliche Entscheidungen getroffen werden, deren Grundlage in frühen Phasen der Entwicklung gelegt wird. Als herausragendes Merkmal der UX kommt der Interaktion mit der Anwendung eine besondere Rolle zu, die im gesamten Entwicklungsprozess beachtet werden muss und deren gemeinsames Verständnis im Projektteam notwendig ist. Diese Arbeit betrachtete zunächst verschiedene Aspekte, die bei der Entwicklung gestenbasierter Anwendungen im Allgemeinen sowie bei der Spezifikation im Besonderen relevant sind:

- Die Unterschiede bzgl. Herangehensweise und Perspektive beim UI-Design und bei der Implementierung von Anwendungen mit Fokus auf HCI (Kap. 2)
- Die typischen Werkzeuge zum UI-Design und wie sich Interaktion und vor allem Gestensteuerung in ihnen manifestiert (Kap. 3)
- Die unterschiedlichen technischen Entscheidungen (Hardware, Sensoren und Algorithmen zur Gestenerkennung), die der Entwicklung gestenbasierter Anwendungen getroffen werden müssen (Kap. 4)

Aufbauend auf diesem Wissen und Verständnis wurde der vorgestellte Lösungsansatz entwickelt, der in verschiedenen Studien validiert wurde. Die Beiträge dieser Arbeit sind im Einzelnen:

- Eine Listung der Eigenschaften von Gesten basierend auf einer Untersuchung wissenschaftlicher Literatur, die beim Einsatz von Gestensteuerung interessant für die weitere Verarbeitung in der Anwendung sind, die Grundlage von Spezifikation bilden und die Basis für die o. g. Entscheidungen während der Implementierung sind (Kap. 5)
- GestureCards, eine hybride Notation zur Beschreibung von Gesten, die textuelle und graphische Elemente zur Beschreibung verschiedener Eigenschaften verwendet (Kap. 7)
- Ein Vergleich von GestureCards mit textuellen und graphischen Notationen zu unterschiedlichen Aspekten (Kap. 8 und Kap. 9)
- Eine Studie zum Einsatz von GestureCards in der Praxis (Kap. 10)
- Ein Vorschlag zur graphischen Repräsentation räumlicher Posen und Gesten in GestureCards mithilfe animierter Handmodelle und animierter Pfeile (Kap. 11)

- Eine Studie, die die beiden genannten Optionen zu Repräsentation räumlicher Gesten vergleicht (Kap. 12)

Die Ergebnisse der Arbeit und die daraus gezogenen Schlüsse werden im Folgenden diskutiert (Kap. 14.1). Abschließend wird noch auf Optionen für zukünftige Arbeiten eingegangen (Kap. 14.2).

14.1 Diskussion der Ergebnisse

Der im Rahmen dieser Arbeit entwickelte Ansatz zur Definition von Gesten unterscheidet sich insbesondere durch die Kombination von graphischen und textuellen Elementen von bestehenden Notationen. Die Evaluierung zeigt, dass die Probanden mit diesem Ansatz gute Ergebnisse erzielten: GestureCards werden schneller und besser verstanden als eine textuelle Vergleichsnotation (vgl. Kap. 8) und als prägnanter und genauer bewertet als eine textuelle und eine graphische Vergleichnsnotationen (vgl. Kap. 9). Insbesondere in der zweiten Studie wird aber auch deutlich, dass die zu definierenden Gesten Einfluss auf die Ergebnisse haben. Der hybride Ansatz spielt seine Stärken vor allem durch die Vermeidung von extrem positiven und extrem negativen Fällen aus. Auch beim praktischen Einsatz – sowohl analog als auch digital durch die Integration in die Sketching-Umgebung AugIR – wurden GestureCards von den Probanden gut aufgenommen und als hilfreiches Werkzeug empfunden (vgl. Kap. 10). GestureCards können daher als verlässliches und vielseitig einsetzbares Werkzeug betrachtet werden, das bei der Entwicklung gestenbasierter Anwendungen unterstützt.

In Kap. 5 wurde ein Überblick über die Eigenschaften von Gesten gegeben, auf dem letztlich die Definition der Notationselemente basiert. In GestureCards werden viele Eigenschaften als Name-Wert-Paare dargestellt. Dieses Vorgehen hat insbesondere den Vorteil, dass Erweiterungen um neue Eigenschaften und eventuell notwendige zusätzliche Werte ohne viel Aufwand bzw. Anpassungen an anderen Notationselementen möglich sind. Auch wenn dieser Punkt nicht explizit im Rahmen der Validierung überprüft wurde, so wird davon ausgegangen, dass Teams GestureCards beim Einsatz erweitern und auf ihre spezifischen Bedürfnisse anpassen werden. Diese Eigenschaft einer Notation ist als positiv zu werten, da sie jedem Team die Freiheiten lässt, die für sich optimalen Arbeitsverfahren zu definieren.

In Kap. 10 wurden GestureCards in eine digitale Sketching-Umgebung integriert und die graphische Repräsentation mit einer Animation der Pfeile erweitert. Obschon die Probanden auch der analogen Variante gute Noten gaben, lobten sie insbesondere die animierten Pfeile. Ungeachtet der Tatsache, dass die Animationen für eine gute UX gesorgt haben, bleibt unklar, ob sie auch tatsächlich einen Vorteil gegenüber den statischen Pfeilen haben; hier ist Potential für weitere Studien zu finden, die diese Frage im Detail betrachten. Dasselbe gilt auch für die unterschiedlichen Optionen zur Darstellung räumlicher Gesten, für die im Rahmen dieser Arbeit lediglich ein Vorschlag gemacht wurde und dessen weitere Ausarbeitung in Zukunft erfolgen wird.

Bei Betrachtung der technischen Weiterentwicklung wird auch deutlich, dass die rein textbasierte Arbeit in Programmierumgebungen in Zukunft zu einem gewissen Teil von einem hybriden Modell verdrängt werden wird. So ist für Anwendungen im Bereich VR/AR die Definition graphischer Szenen notwendig, die mit kurzen Skripten angereichert werden, in denen die Verknüpfung zur notwendigen Logik der Anwendung hergestellt wird. Letztere wird zwar – mutmaßlich – weiterhin textbasierte Programmierung sein, jedoch wird ein wesentlicher Bestandteil der Arbeit sein, die Verknüpfungen an der richtigen Stelle zu implementieren. Es erscheint sinnvoll, auch die Integration der Interaktion mit einem ähnlichen Konzept an dieser Stelle zu verankern. Eine Notation wie GestureCards kann an dieser Stelle verwendet werden. Darüber hinaus passt die Idee von GestureCards sehr gut in eine solche Arbeitsumgebung, denn hier können gezielt Aktionen in der Anwendung miteinander verknüpft und die Verknüpfung mit GestureCards markiert werden. Dadurch entsteht eine gewisse Analogie zum klassischen Storyboard und die Möglichkeit, alle im UI notwendigen Elemente zentral zu definieren.

Es wird erwartet, dass Gestensteuerung in Zukunft häufiger eingesetzt wird und bei der Entwicklung von Anwendungen gezielter betrachtet werden muss. Die zahlreichen Möglichkeiten stellen Entwickler nicht nur vor technische Herausforderungen, sondern machen ein gemeinsames Verständnis der Zielstellung umso wichtiger. GestureCards stellen ein notwendiges Werkzeug zur Verfügung, um den Austausch von Informationen über Gesten zu unterstützen und die Entwicklung gestenbasierter Anwendungen effizienter zu gestalten.

14.2 Erweiterungspotential und zukünftige Arbeiten

In Kap. 13 wurden bereits einige weitere Anwendungsfälle für den Einsatz von GestureCards dargelegt, von denen einige vielversprechendes Potential für zukünftige Arbeiten bieten. Für die Integration in den AugIR (vgl. Kap. 10.1.3) wurde bereits ein Gestenerkenner entwickelt, der mit GestureCards eingesetzt werden kann. Dieser kann jedoch lediglich Oberflächengesten erkennen und eine spannende Forschungsfrage ist, ob sich das Konzept auch reibungslos auf räumliche Gesten übertragen lässt. Für zukünftige Arbeiten ist außerdem – auch im Kontext des AugIR – die Entwicklung eines Algorithmus zur Ermittlung der Ähnlichkeit definierter Gesten interessant (vgl. Kap. 13.2). Die Automatisierung von Oberflächentests für gestenbasierte UIs, wie sie in Kap. 13.3 beschrieben wird, birgt ebenfalls interessantes Potential für zukünftige Arbeiten. Für ein derartiges Testsystem sind auch noch zahlreiche unterschiedliche Fragen zu beantworten, insbesondere bzgl. der automatischen Generierung der passenden Varianz im Kontext unterschiedlicher, simulierter Nutzertypen.

Die genannten Anwendungsfälle transferieren GestureCards aber in erster Linie in andere Anwendungsbereiche, in denen die Spezifikation von Interaktion auch eine zentrale Rolle spielt. Die fortschreitende Entwicklung im Bereich HCI bietet jedoch ebenfalls Potential für die Erweiterung von GestureCards zur Beschreibung anderer Arten von Interaktion. Im Folgenden soll auf die Integration unterschiedlicher Körperteile (Kap. 14.2.1) und von Multimodalität (Kap. 14.2.2) eingegangen werden.

14.2.1 Erweiterung auf Bewegungen und Posen des ganzen Körpers

Im Rahmen dieser Arbeit wurde vorgeschlagen, wie räumliche Gesten und Posen mit den Händen in GestureCards dargestellt werden können (vgl. Kap. 11). Die Validierung (vgl. Kap. 12) zeigt jedoch, dass es zukünftiger Arbeiten bedarf, die die verschiedenen Möglichkeiten detaillierter ausarbeiten und ihre Vor- und Nachteile zeigen. So müssen in einer zukünftigen Arbeit die Parameter definiert werden, nach denen eine Darstellung mit einem Pfeil- oder Handmodell (oder einer anderen Alternative) zu bevorzugen ist. Daraus kann z. B. eine Taxonomie folgen, nach der Gesten für die bestimmte Darstellungen – oder gar Mischformen – klassifiziert werden können.

Ungeachtet der Notwendigkeit weiterer Arbeiten in diesem Gebiet ermöglicht es moderne Sensortechnik auch, eine Vielzahl unterschiedlicher Informationen über Benutzer zu sammeln und sie zur Interaktion einzusetzen. Neben Händen und Fingern, die in dieser Arbeit im Fokus stehen, können auch andere Körperteile für HCI verwendet werden. Dabei ist aber jedes Körperteil mit eigenen Einschränkungen und Besonderheiten verbunden, die eine tiefergehende Auseinandersetzung mit den jeweiligen Möglichkeiten erfordern. Konzepten wie der Prüfung von Interaktionsmustern (vgl. Kap. 13.2) kommt dann eine besondere Bedeutung zu, denn mit der Vielzahl der Möglichkeiten erhöht sich auch die Anzahl der Unmöglichkeiten.

Velloso et al. (2015) geben einen Überblick, wie die Füße in HCI verwendet werden können. Dabei decken die verschiedenen Möglichkeiten ein großes Spektrum von eher feingliedrigen Bewegungen wie der Berührung der großen Zehen beider Füße hin zu groben Bewegungen wie einem Tritt oder einem Aufstampfen ab. Für die Interaktion mit den Füßen können unterschiedliche Sensoren eingesetzt werden: Neben berührungsempfindlichen Oberflächen und Sensoren, die die Füße im Raum erfassen, können auch intelligente Schuhe eingesetzt werden (z. B. bei Horodniczy & Cooperstock, 2017). Für die Beschreibung von Interaktion mit den Füßen können die in GestureCards definierten Konzepte für die Hände weitgehend übernommen werden. Für die graphische Repräsentation bietet sich neben den Pfeilen ein Modell des Fußes an. Es ist allerdings davon auszugehen, dass Posen mit den Füßen eher auf die Positionierung der Füße im Raum reduziert werden müssen. Die für Gesten definierten Eigenschaften können im Wesentlichen übernommen werden.

Wenn Gesten mit Händen und Füßen beschrieben werden können, ist der Schritt zur Beschreibung von Kombinationen nicht weit. Und von hier ist es nur noch ein kurzer Schritt, Bewegungen und Posen des gesamten Körpers zu beschreiben. Neben Bewegung und Position kann dann auch die Körperhaltung eine Rolle spielen.

Eine interessante Möglichkeit, die Grenze zwischen zwischenmenschlicher Kommunikation und HCI noch weiter zu verwischen, ist die Erkennung von Emotionen. Gesten, die mit GestureCards definiert sind, könnten um die Möglichkeit erweitert werden, menschliche Gemütszustände zu definieren, die Anwendungen (z. B. Robotern, die in direktem Kontakt zu Menschen stehen) weiterführendes Verständnis für menschliche Aktionen vermitteln.

14.2.2 Integration von Multimodalität

Die fortschreitende Verfügbarkeit unterschiedlicher, neuartiger Sensoren ermöglicht eine immer tiefere Integration von HCI in den beruflichen und privaten Alltag. Dieser Wandel vollzieht sich mit spürbarer Geschwindigkeit: Nicht nur Gesten-, sondern auch Sprachsteuerung hat es in Form verschiedener digitaler Assistenten auf mobile Geräte und mittlerweile sogar auf den klassischen Desktop-PC geschafft. Die Konsequenz ist letztlich, dass sich die Anzahl der Kanäle, über die Benutzer mit Computern kommunizieren können, vervielfacht: *Multimodalität* bezeichnet die Möglichkeit, verschiedene Spielarten von HCI parallel in einer Anwendung verwenden zu können. Die Grundlage dafür hat Bolt (1980) mit der Anwendung *Put That There* gelegt, mit der er zeigte, wie Sprach- und Gestensteuerung[1] zu einem Kommando verschmolzen werden können. Bei der Entwicklung einer neuen Anwendung muss somit eine noch intensivere Auseinandersetzung mit den für den jeweiligen Anwendungsfall möglichen Modalitäten und ihren konkreten Ausprägungen geschehen. Gleichzeitig ist es wesentlich schwieriger, zu identifizieren, in welchem Kontext sich welcher Benutzer für welche Modalität entscheidet: Auch wenn Multimodalität möglich ist und sogar bevorzugt wird, bedeutet das nicht, dass Benutzer auch wirklich multimodal arbeiten (Oviatt et al., 1997). Ungeachtet dessen bedeuten mehr Modalitäten auch mehr Entscheidungen für die eingesetzten technischen Komponenten. Somit kommt auch der Spezifikation weitere Bedeutung zu.

[1] Oviatt (1999) macht allerdings darauf aufmerksam, dass im Beispiel von Bolt (1980) lediglich das klassische PnC-Paradigma mausbasierter Anwendungen in andere Modalitäten übertragen wird und dass wirkliche Multimodalität viel weitreichendere Möglichkeiten bietet.

Bei der Spezifikation multimodaler Anwendungen müssen zwangsläufig unterschiedliche Modalitäten und ihre Bezüge zueinander beschrieben werden können. Wenn schon Notationen, die das gleiche Interaktionsparadigma beschreiben, sehr unterschiedlich wahrgenommen werden, so muss die Beschreibung multimodaler Anwendung nochmal explizit untersucht werden. Für zukünftige Arbeiten ist somit interessant, wie GestureCards um weitere Modalitäten zu *InteractionCards* erweitert werden können. Das GestureCards zugrundeliegende Schema kann hier u. U. wiederverwendet werden. Für jede Modalität müsste eine Menge denkbarer Eigenschaften definiert werden. In der Modalität Sprachsteuerung wären z. B. die Lautstärke oder der Tonfall denkbare Eigenschaften.

Multimodalität bedeutet aber vor allem, dass unterschiedliche Interaktionsmuster eingesetzt werden, deren tatsächliche Ausprägung stark abhängig vom jeweiligen Benutzer ist und deren konkrete Umsetzung sich in den eingesetzten Modalitäten unterscheiden kann (Oviatt, 1999). Zwangsläufig müssen sich Teams somit viel stärker mit der Interaktion befassen als zuvor. Vielleicht kann eine Integration in eine Design- und Entwicklungsumgebung wie dem AugIR so, wie im Rahmen dieser Arbeit für Gestensteuerung und die Spezifikation mit GestureCards demonstriert, auch die Entwicklung multimodaler Anwendungen unterstützen. So könnte die parallele Definition unterschiedlicher Interaktionskonzepte ermöglicht werden, die die verschiedenen Modalitäten entweder einzeln oder kombiniert umsetzen.

Anhänge

Zusammenfassung aller Eigenschaften und Werte

A

Im Folgenden werden alle Gesteneigenschaften, die mit GestureCards definiert werden können, sowie die entsprechenden Schlüsselwörter und möglichen Werte gelistet.

A.1 Anzahl der Werkzeuge

Schlüsselwort: `Tool Count`

Wert	Beschreibung
Zahl	Anzahl der verwendeten Werkzeuge

Tabelle A.1: Werte der Eigenschaft Anzahl der Werkzeuge

- Kann in der graphischen Representation der Geste mit mehreren Gestenzügen dargestellt werden
- Kann auch durch Definition separater GestureCards für parallel ausgeführter Teilgesten definiert werden. Dieses Vorgehen ist zwingend notwendig, wenn Eigenschaften von Teilgesten sich unterscheiden (z. B. bei Verwendung unterschiedlicher Werkzeuge oder Startpunkte)

A.2 Auflagefläche

Schlüsselwort: `Contact Area`

Wert	Beschreibung
`Increase`	Auflagefläche wächst
`Decrease`	Auflagefläche schrumpft

Tabelle A.2: Werte der Eigenschaft Auflagefläche

- Nur für Oberflächengesten verfügbar

A.3 Ausführender Benutzer

Schlüsselwort: `User`

Tabelle A.3: Werte der Eigenschaft Ausführender Benutzer

Wert	Beschreibung
`All`	Alle Benutzer führen diese Geste aus
Bruch	Teilmenge der Benutzer, die die Geste ausführen
Zahl	Anzahl Benuter, die die Geste ausführen
Buchstabe	Bezeichnung unterschiedlicher Benutzer

A.4 Ausrichtung

Schlüsselwort: `Orientation`

Tabelle A.4: Werte der Eigenschaft Ausrichtung

Wert	Beschreibung
`Fixed`	Die Geste muss bei der Ausführung genauso ausgerichtet sein, wie bei der Spezifikation festgelegt
`None`	Die Geste kann bei der Ausführung beliebig ausgerichtet sein

A.5 Bewegungsrichtung

Schlüsselwort: `Direction`

Tabelle A.5: Werte der Eigenschaft Bewegungsrichtung

Wert	Beschreibung
`Towards`	Geste bewegt sich auf eine andere Geste/ein virtuelles Objekt zu
`Away From`	Geste bewegt sich von einer anderen Geste/einem virtuellen Objekt weg

- Immer durch Pfeile in der graphischen Repräsentation dargestellt
- Definition der Eigenschaft mit Schlüsselwörtern hebt Richtung der Pfeile der graphischen Repräsentation auf, um Dynamik der Richtung zu ermöglichen

A.6 Dauer

Schlüsselwort: `Duration`

Wert	Beschreibung
`Impulse`	Sehr kurze Berührung der Oberfläche, z. B. Berührung und sofortiges Verlassen einer berührungsempfindlichen Oberfläche
`Short`	Kurze Berührung der Oberfläche, z. B. Berührung und kurzes Verweilen auf der Oberfläche
`Long`	Eine lange Berührung der Oberfläche, mindestens doppelt so lang wie `Short`

Tabelle A.6: Werte der Eigenschaft Dauer

- Die genaue Länge der jeweiligen Berührungen muss projektspezifisch definiert werden; als Faustregel gilt, dass sich die Dauer jeweils von Schlüsselwort zu Schlüsselwort verdoppelt

A.7 Distanz

Schlüsselwort: `Distance`

Wert	Beschreibung
`Far From`	Eine Teilgeste befindet sich weit weg von einer anderen Geste
`Close To`	Eine Teilgeste wird in der Nähe einer anderen Teilgeste ausgeführt
`Touches`	Teilgesten werden direkt nebeneinander ausgeführt

Tabelle A.7: Werte der Eigenschaft Distanz

- Werte müssen abhängig von der verwendeten Geräteklasse definiert werden; z. B. kann bei großen berührungsempfindlichen Bildschirmen die Schulterbreite als Maßgabe für `Close To` gelten

A.8 Druck

Schlüsselwort: `Pressure`

Tabelle A.8: Werte der Eigenschaft Druck

Wert	Beschreibung
`Hover`	Kein Druck; Werkzeug schwebt über der Oberfläche
`Very Weak`	Sehr schwacher Druck
`Weak`	Schwacher Druck
`Normal`	Üblicher Druck, z. B. wie zum Antippen eines Buttons üblich
`Strong`	Starker Druck
`Very Strong`	Sehr starker Druck

A.9 Endpunkt

Schlüsselwort: `End`

- Besondere Form des Ortes der Ausführung; verwendet entsprechend die gleichen Werte (vgl. Tab. A.11)
- Kann auch Werte von räumlicher Relation (vgl. Tab. A.15) und Distanz (vgl. Tab. A.7) annehmen

A.10 Form

Schlüsselwort: –

- Grundsätzlich in der graphischen Repräsentation
- Freie Formen werden durch das Schlüsselwort FREE anstelle einer Grafik spezifiziert
- Zeigen/Berühren mit Symbol dargestellt in Abb. A.1

Abbildung A.1: Symbol für Zeigen/-Berüchren in GestureCards

A.11 Geschwindigkeit

Schlüsselwort: `Speed`

Tabelle A.9: Werte der Eigenschaft Geschwindigkeit

Wert	Beschreibung
`Normal`	Eine langsame Bewegung
`Fast`	Eine schnelle Bewegung
`Increasing`	Eine beschleunigende Bewegung
`Decreasing`	Eine abbremsende Bewegung

A.12 Multiplizität

Schlüsselwort: `Multiplicity`

Wert	Beschreibung
`Once`	Genau eine Ausführung
`Repeat`	Mehrmalige Ausführung
Zahl	Anzahl der Wiederholungen

Tabelle A.10: Werte der Eigenschaft Multiplizität

A.13 Ort der Ausführung

Schlüsselwort: `Location`

Wert	Beschreibung
`Top`	Oberer Teil
`Bottom`	Unterer Teil
`Center`	Mittlerer Teil; vertikal gesehen
`Left`	Linker Teil
`Right`	Rechter Teil
`Middle`	Mittlerer Teil; horizontal gesehen
`Front`	Vorderer Teil; nur für räumliche Gesten
`Even`	Gleiche Höhe; nur für räumliche Gesten
`Back`	Hinterer Teil; nur für räumliche Gesten
`Sensor`	Bezugspunkt Sensor, nur für räumliche Gesten
`User`	Bezugspunkt Benutzer, nur für räumliche Gesten
`Off`	Außerhalb des Bildschirms/Sensors
`Not`	Negation von Orten; muss von einem Ort, einer Teilgeste oder einem Objekt gefolgt werden

Tabelle A.11: Werte der Eigenschaft Ort der Ausführung

- Kombinationen sind möglich, z. B. beschreibt `Location: Off Right` den Bereich rechts außerhalb des Bildschirms/Sensors
- Kombinationen sind grundsätzlich Und-Vknüpfungen; zur Definition von Alternativen müssen neue Gesten definiert werden

A.14 Reaktion zur Ausführungszeit

Schlüsselwort: `Reaction`

Wert	Beschreibung
`Online`	Anwendung reagiert zur Ausführungszeit der Geste
`Offline`	Anwendung reagiert nach Ausführung der Geste

Tabelle A.12: Werte der Eigenschaft Reaktion zur Ausführungszeit

A.15 Rotation

Schlüsselwort: `Rotation`

Tabelle A.13: Werte der Eigenschaft Rotation

Wert	Beschreibung
`Clockwise`	Drehung des Werkzeugs im Uhrzeigersinn
`Counterclockwise`	Drehung des Werkzeugs gegen den Uhrzeigersinn

- Grad der Drehung können als numerischer Wert hinzugefügt werden

A.16 Startpunkt

Schlüsselwort: `Start`

- Besondere Form des Ortes der Ausführung; verwendet entsprechend die gleichen Werte (vgl. Tab. A.11)
- Kann auch Werte von räumlicher Relation (vgl. Tab. A.15) und Distanz (vgl. Tab. A.7) annehmen

A.17 Werkzeug

Schlüsselwort: `Tool`

Tabelle A.14: Werte der Eigenschaft Werkzeug

Wert	Beschreibung
`Finger`	Ein beliebiger Finger
`Hand`	Eine beliebige Hand
`Pen`	Ein Stift zur Eingabe auf digitalen Geräten, z. B. ein Stylus
`Device`	Zur Durchführung von Gesten mit dem Gerät/Sensor (z. B. Smartphone, Magic Wand usw.)

- Verfeinerungen sind möglich, z. B. `Left Hand` oder `Right Hand`

A.18 Größe

Schlüsselwort: `Size`

- Erwartet einen Vergleich (mithilfe eines Operators) mit einer anderen Geste oder einem virtuellen Objekt

A.19 Räumlicher Bezug

Schlüsselwort: `Position`

Wert	Beschreibung
`Above`	Über einer anderen Geste
`Below`	Unter einer anderen Geste
`Left of`	Links von einer anderen Geste
`Right of`	Rechts von einer anderen Geste
`In front of`	Vor einer anderen Geste; nur für räumliche Gesten
`Behind`	Hinter einer anderen Geste; nur für räumliche Gesten
`Crosses`	Kreuzt eine andere Geste

Tabelle A.15: Werte der Eigenschaft Räumlicher Bezug

- Immer mit Bezug auf eine andere Teilgeste

A.20 Zeitlicher Bezug

Schlüsselwort: `--`

- Wird über visuelle Darstellung der GestureCards aller Teilgesten dargestellt (vgl. Abb. A.2)
- Zeitlich Verzug oder Pausen zwischen Gestenzügen werden mit geschlängelten Linien dargestellt

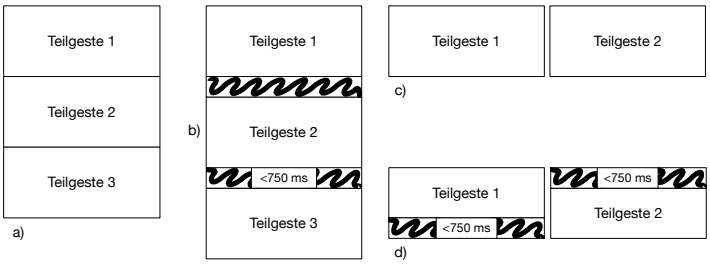


Abbildung A.2: Optionen zur Darstellung der zeitlichen Relation: a) einzügige Gesten, b) mehrzügige Gesten, c) Parallelisierung, d) zeitlicher Versatz von Start- und Endpunkten paralleler Teilgesten

Material zu Kap. 8: Lesbarkeit und Verständlichkeit textueller und hybrider Notationen

B

B.1 Einführung der Notationen

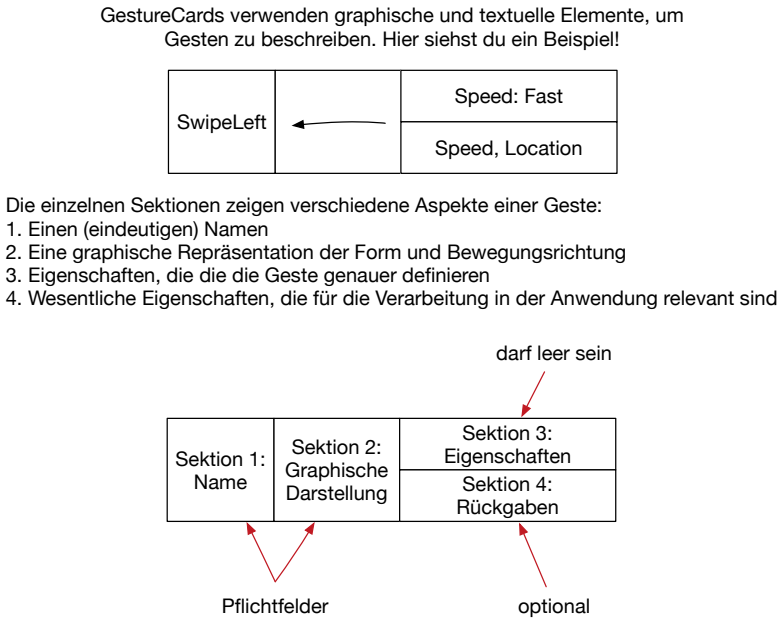


Abbildung B.1: Einführung in GestureCards, Seite 1

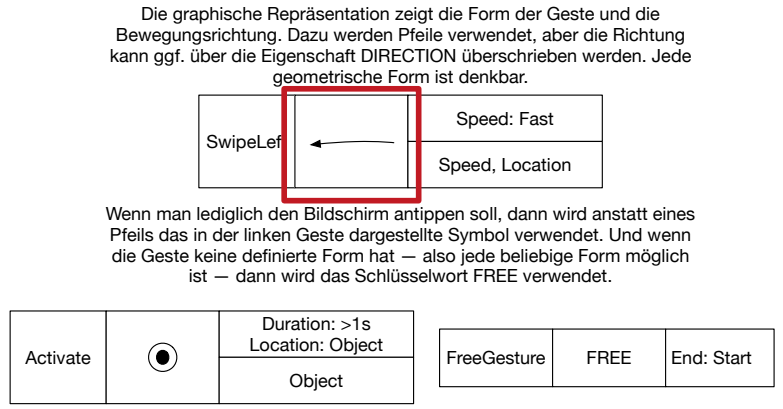


Abbildung B.2: Einführung in GestureCards, Seite 2

Abbildung B.3: Einführung in GestureCards, Seite 3

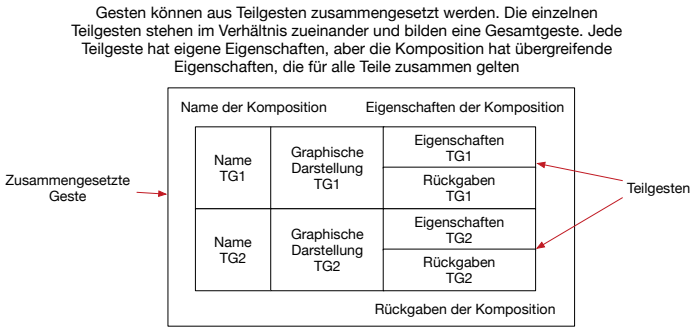


Abbildung B.4: Einführung in GestureCards, Seite 4

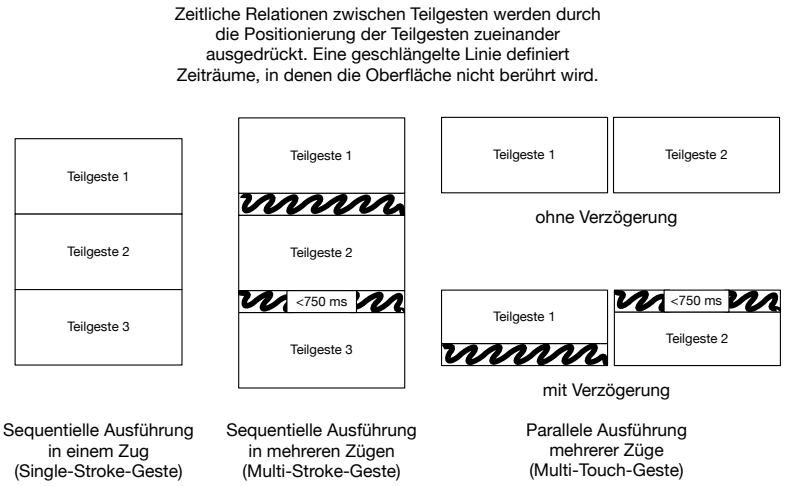
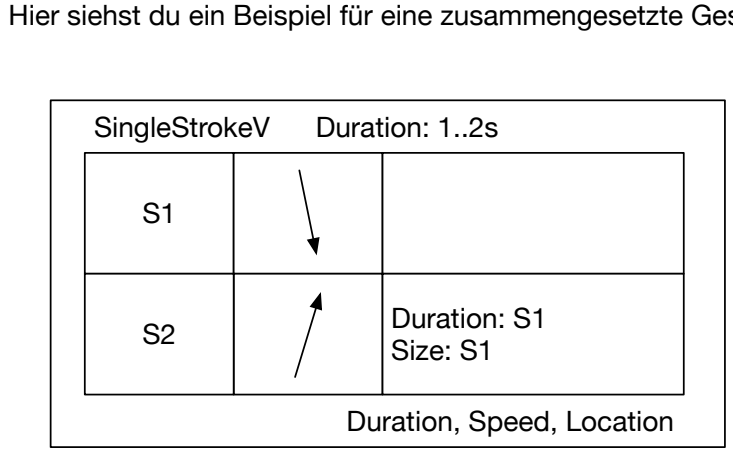


Abbildung B.5: Einführung in GestureCards, Seite 5



GeForMT verwendet eine textuelle Notation, um Gesten zu beschreiben. Hier siehst du ein Beispiel!

1F(LINE_W)

Die einzelnen Sektionen zeigen verschiedene Aspekte einer Geste:
1. Wie viele Finger den Bildschirm berühren
2. Welche Form die Gesten hat
3. In welche Richtung sich die Finger bewegen

GeForMT gibt eine Grammatik mit allerhand unterschiedlichen Möglichkeiten vor, diese siehst du hier abgebildet

```
 1: complex     ::=  gesture (operator gesture)* |
                     relation'['gesture (operator gesture)*']'
 2: gesture     ::=  function'('atomfocus(','atomfocus)*')' |
                     atomfocus(','atomfocus)*
 3: relation    ::=  'CROSS' | 'JOIN' | 'SYNC' | 'SPREAD'
 4: atomfocus   ::=  atom | atom'('focuslist')'
 5: focuslist   ::=  focus(','focus)*
 6: focus       ::=  'a' | 's' | 'o'[integer]
 7: function    ::=  [integer]'F' | [integer]'H' | [integer]'B'
 8: operator    ::=  '*' | '+' | ','
 9: atom        ::=  identifier | vector['_'direction] |
                     shape['_'direction]['_'rotation]
10: identifier  ::=  'MOVE' | 'POINT' | 'DEPOINT' | 'HOLD'
11: vector      ::=  'LINE'
12: shape       ::=  'CIRCLE' | 'SEMICIRCLE'
13: direction   ::=  'NORTH' | 'N' | 'NORTHEAST' | 'NE' |
                     'EAST' | 'E' | 'SOUTHEAST' | 'SE' | 'SOUTH'
                     | 'S' | 'SOUTHWEST' | 'SW' | 'WEST' | 'W' |
                     'NORTHWEST' | 'NW'
14: rotation    ::=  'CLOCKWISE' | 'CW' |
                     'COUNTERCLOCKWISE' | 'CCW'
15: integer     ::=  ('1'-'9') ('0'-'9')*
```

Abbildung B.6: Einführung in GeForMT, Seite 1 (basierend auf Kammer et al. (2015))

Die Form einer Geste wird durch vorgegebene Schlüsselwörter beschrieben.
Die Bewegungsrichtung durch Angabe einer Himmelsrichtung.

1F(LINE_W)

Mögliche Formen sind:
LINE, CIRCLE, SEMICIRCLE

Bei LINE wird die Bewegungsrichtung angegeben

Bei CIRCLE der Start-/Endpunkt und bei SEMICIRCLE der Endpunkt. Außerdem wird die Drehrichtung (mit/gegen Uhrzeigersinn) angegeben.

Wenn man lediglich den Bildschirm antippen soll, dann wird POINT oder HOLD verwendet. Und wenn die Geste keine definierte Form hat — also jede beliebige Form möglich ist — dann wird das Schlüsselwort MOVE verwendet.

Mögliche Richtungen sind:

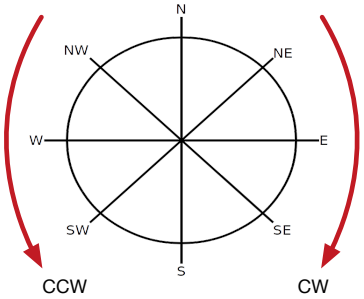


Abbildung B.7: Einführung in GeForMT, Seite 2 (basierend auf Kammer et al. (2015))

Gesten können aus Teilgesten zusammengesetzt werden. Die einzelnen Teilgesten stehen im Verhältnis zueinander und bilden eine Gesamtgeste. Jede Teilgeste explizit definiert und über Operatoren mit anderen Teilgesten zur Gesamtgeste zusammengeführt. Hier der grundlegende Aufbau:

Beziehung[Fingerzahl(Teilgeste1), Fingerzahl(Teilgeste2)]

optional

Operator

Abbildung B.8: Einführung in GeForMT, Seite 3 (basierend auf Kammer et al. (2015))

Abbildung B.9: Einführung in GeForMT, Seite 4 (basierend auf Kammer et al. (2015))

Zeitliche Relationen zwischen Teilgesten werden durch verschiedene Operatoren zwischen Teilgesten ausgedrückt.

+	Asynchrone Ausführung (Multi-Touch-Geste mit Verzögerung)
*	Synchrone Ausführung (Multi-Touch-Geste ohne Verzögerung)
,	Sequentielle Ausführung ohne Unterbrechung (Single-Stroke-Geste)
;	Sequentielle Ausführung mit Unterbrechung (Multi-Stroke-Geste)

Abbildung B.10: Einführung in GeForMT, Seite 5 (basierend auf Kammer et al. (2015))

Hier siehst du ein Beispiel für eine zusammengesetzte Geste

1F(HOLD(o)) * 1F(HOLD(o)) + 1F(LINE(a))

B.2 Spickzettel

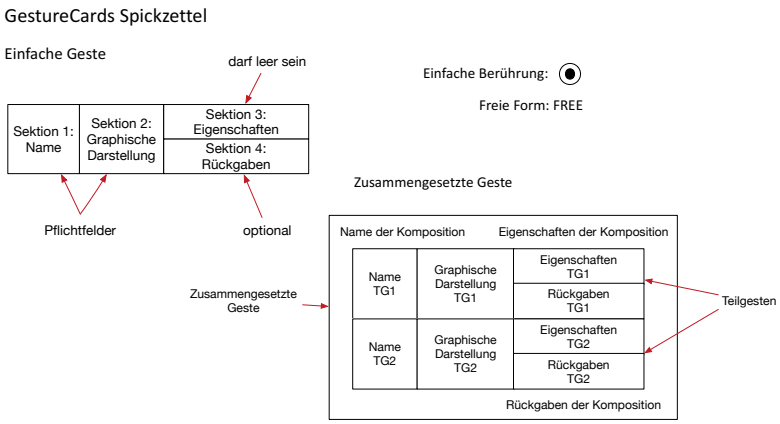


Abbildung B.11: Studie A: Spickzettel für GestureCards, Seite 1

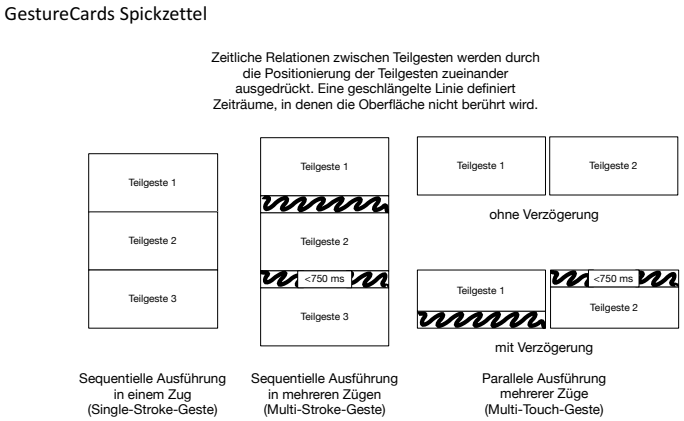


Abbildung B.12: Studie A: Spickzettel für GestureCards, Seite 2

Abbildung B.13: Studie A: Spickzettel für GeForMT, Seite 1 (basierend auf Kammer et al. (2015))

GeForMT Spickzettel

Form	Beschreibung
POINT	Kurzer Kontakt
HOLD	Langer Kontakt
MOVE	Freie Form
LINE	Linie
CIRCLE	Kreis
SEMICIRCLE	Halbkries

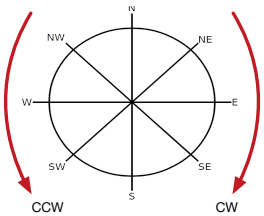
Räumliche Beziehung	Beschreibung
CROSS	Überschneidung
SPREAD	Auseinandergehen
JOIN	Zusammengehen
SYNC	Parallel
CLOSE	Geschlossen
ASIDE	Nebeneinander
AMONG	Untereinander
CONNECT_START	Verbindung am Start
CONNECT_END	Verbindung am Ende

Zeitliche Beziehung	Beschreibung
+	Asynchrone Ausführung (Multi-Touch-Geste mit Verzögerung)
*	Synchrone Ausführung (Multi-Touch-Geste ohne Verzögerung)
,	Sequentielle Ausführung ohne Unterbrechung (Single-Stroke-Geste)
;	Sequentielle Ausführung mit Unterbrechung (Multi-Stroke-Geste)

Abbildung B.14: Studie A: Spickzettel für GeForMT, Seite 2 (basierend auf Kammer et al. (2015))

GeForMT Spickzettel

Bewegungs- und Drehrichtung



Werkzeug	Beschreibung
F	Finger
H	Hand
B	Unspezifiziert

```
1: complex     ::= gesture (operator gesture)* |
                   relation'['gesture (operator gesture)*']'
2: gesture     ::= function'('atomfocus(','atomfocus)*')' |
                   atomfocus(','atomfocus)*
3: relation    ::= 'CROSS' | 'JOIN' | 'SYNC' | 'SPREAD'
4: atomfocus   ::= atom | atom'('focuslist')'
5: focuslist   ::= focus(','focus)*
6: focus       ::= 'a' | 's' | 'o'[integer]
7: function    ::= [integer]'F' | [integer]'H' | [integer]'B'
8: operator    ::= '*' | '+' | ','
9: atom        ::= identifier | vector['_'direction] |
                   shape['_'direction]['_'rotation]
10: identifier ::= 'MOVE' | 'POINT' | 'DEPOINT' | 'HOLD'
11: vector     ::= 'LINE'
12: shape      ::= 'CIRCLE' | 'SEMICIRCLE'
13: direction  ::= 'NORTH' | 'N' | 'NORTHEAST' | 'NE' |
                   'EAST' | 'E' | 'SOUTHEAST' | 'SE' | 'SOUTH'
                   | 'S' | 'SOUTHWEST' | 'SW' | 'WEST' | 'W' |
                   'NORTHWEST' | 'NW'
14: rotation   ::= 'CLOCKWISE' | 'CW' |
                   'COUNTERCLOCKWISE' | 'CCW'
15: integer    ::= ('1'-'9') ('0'-'9')*
```

B.3 Ergebnisse aller Probanden

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
SSL	Sicher	00:02:37,627	00:01:00,875	00:00:06,062	2	ja
X	Sicher	00:01:50,767	00:01:08,454	00:00:03,734	4	nein
HoldMove	Eher Unsicher	00:03:16,174	00:01:58,063	00:00:06,344	5	nein
Y	Sicher	00:02:03,392	00:01:36,329	00:00:08,999	2	nein
LSLS	Sicher	00:00:13,797	00:00:11,531	00:00:02,265	0	ja
Pfeil	Eher Unsicher	00:01:52,735	00:01:16,985	00:00:05,015	3	nein
Spread	Sicher	00:00:24,047	00:00:18,125	00:00:01,390	1	nein
V	Eher Sicher	00:01:05,985	00:00:49,313	00:00:09,218	1	ja

Tabelle B.1: Ergebnisse des Probanden GeForMT1

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
SSL	Eher Sicher	00:00:16,312	00:00:14,828	00:00:01,484	0	ja
Pfeil	Eher Sicher	00:02:01,376	00:01:33,470	00:00:19,765	1	ja
X	Eher Sicher	00:00:33,844	00:00:21,797	00:00:06,968	1	ja
HoldMove	Unsicher	00:02:39,705	00:01:22,485	00:00:00,000	3	nein
Y	Eher Unsicher	00:01:55,876	00:00:35,563	00:00:52,938	1	nein
V	Eher Sicher	00:00:22,281	00:00:15,719	00:00:06,562	0	ja
Spread	Eher Unsicher	00:01:17,454	00:00:47,703	00:00:01,109	2	ja
LSLS	Eher Sicher	00:00:22,656	00:00:19,969	00:00:02,687	0	ja

Tabelle B.2: Ergebnisse des Probanden GeForMT2

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
SSL	Eher Sicher	00:00:58,750	00:00:34,609	00:00:01,734	1	ja
Spread	Eher Sicher	00:00:46,141	00:00:25,625	00:00:06,406	1	nein
V	Eher Unsicher	00:01:28,844	00:01:00,328	00:00:04,484	1	ja
Pfeil	Eher Sicher	00:01:26,844	00:01:09,438	00:00:07,140	2	nein
HoldMove	Eher Unsicher	00:02:07,501	00:00:46,766	00:00:07,968	4	nein
Y	Eher Sicher	00:03:26,721	00:02:11,454	00:00:33,125	3	ja
X	Eher Sicher	00:00:48,172	00:00:38,547	00:00:06,265	1	ja
LSLS	Eher Sicher	00:00:35,312	00:00:23,375	00:00:03,609	1	ja

Tabelle B.3: Ergebnisse des Probanden GeForMT3

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
Spread	Eher Sicher	00:02:33,142	00:01:26,048	00:00:03,609	5	nein
LSLS	Eher Unsicher	00:01:32,141	00:01:08,172	00:00:04,484	1	ja
SSL	Eher Unsicher	00:00:44,813	00:00:33,219	00:00:02,343	1	ja
V	Sicher	00:02:27,767	00:00:50,782	00:00:01,281	5	ja
X	Sicher	00:00:39,719	00:00:30,469	00:00:01,187	2	ja
Pfeil	Sicher	00:01:28,220	00:01:15,688	00:00:03,484	2	ja
Y	Eher Unsicher	00:02:57,596	00:01:41,188	00:00:31,125	5	nein
HoldMove	Unsicher	00:03:12,096	00:01:57,954	00:00:06,671	6	nein

Tabelle B.4: Ergebnisse des Probanden GeForMT4

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
Y	Eher Sicher	00:05:36,707	00:03:23,581	00:00:09,046	11	nein
Pfeil	Sicher	00:01:25,673	00:00:57,594	00:00:05,156	2	nein
Spread	Sicher	00:01:33,032	00:00:49,719	00:00:01,328	2	ja
HoldMove	Eher Sicher	00:02:03,017	00:01:25,470	00:00:07,156	3	ja
X	Eher Sicher	00:01:06,594	00:00:56,672	00:00:02,031	1	ja
SSL	Sicher	00:00:16,656	00:00:12,156	00:00:01,484	1	ja
LSLS	Eher Sicher	00:00:25,812	00:00:21,547	00:00:04,265	0	ja
V	Sicher	00:00:46,766	00:00:45,297	00:00:01,468	0	nein

Tabelle B.5: Ergebnisse des Probanden GeForMT5

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
Spread	Eher Sicher	00:01:29,985	00:00:35,703	00:00:02,640	3	ja
Pfeil	Eher Sicher	00:02:10,892	00:01:02,219	00:00:01,890	5	nein
V	Eher Sicher	00:01:19,282	00:00:21,969	00:00:08,281	3	nein
LSLS	Eher Unsicher	00:00:37,375	00:00:15,547	00:00:01,468	2	nein
Y	Eher Unsicher	00:02:38,377	00:01:31,923	00:00:06,281	4	nein
X	Eher Sicher	00:00:54,532	00:00:26,312	00:00:04,250	1	nein
SSL	Eher Sicher	00:00:40,953	00:00:22,906	00:00:02,968	1	nein
HoldMove	Unsicher	00:01:55,001	00:00:48,157	00:00:00,421	2	nein

Tabelle B.6: Ergebnisse des Probanden GeForMT6

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
X	Eher Sicher	00:00:35,219	00:00:32,359	00:02,9	0	ja
LSLS	Eher Sicher	00:00:35,937	00:00:31,391	00:04,5	0	ja
Pfeil	Eher Unsicher	00:00:34,141	00:00:32,562	00:01,6	0	ja
Spread	Sicher	00:00:25,375	00:00:23,562	00:01,8	0	ja
SSL	Sicher	00:00:15,515	00:00:14,172	00:01,3	0	ja
Y	Eher Sicher	00:00:16,547	00:00:14,250	00:02,3	0	ja
V	Sicher	00:00:07,859	00:00:06,828	00:01,0	0	ja
HoldMove	Eher Unsicher	00:00:37,172	00:00:34,250	00:02,9	0	ja

Tabelle B.7: Ergebnisse des Probanden GC1

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
LSLS	Sicher	00:00:31,309	00:00:28,765	00:00:02,543	0	ja
Spread	Eher Sicher	00:00:41,385	00:00:37,323	00:00:04,062	0	ja
X	Sicher	00:00:25,712	00:00:23,837	00:00:01,875	0	ja
V	Sicher	00:00:10,864	00:00:09,708	00:00:01,156	0	ja
Y	Eher Sicher	00:00:21,988	00:00:19,640	00:00:02,347	0	ja
HoldMove	Eher Unsicher	00:01:29,595	00:01:24,173	00:00:05,421	0	ja
Pfeil	Eher Sicher	00:00:49,130	00:00:47,489	00:00:01,640	0	ja
SSL	Sicher	00:00:18,529	00:00:16,717	00:00:01,812	0	ja

Tabelle B.8: Ergebnisse des Probanden GC2

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
SSL	Sicher	00:01:13,735	00:00:47,688	00:00:02,609	1	ja
Pfeil	Eher Sicher	00:00:46,000	00:00:44,031	00:00:01,968	0	nein
LSLS	Eher Sicher	00:01:08,422	00:00:47,047	00:00:03,078	1	ja
Y	Sicher	00:00:20,578	00:00:17,140	00:00:03,437	0	ja
V	Sicher	00:00:13,640	00:00:12,015	00:00:01,625	0	ja
X	Sicher	00:00:29,375	00:00:26,469	00:00:02,906	0	ja
Spread	Sicher	00:00:17,984	00:00:15,390	00:00:02,593	0	ja
HoldMove	Unsicher	00:01:23,751	00:01:13,704	00:00:10,047	0	nein

Tabelle B.9: Ergebnisse des Probanden GC3

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
Y	Sicher	00:00:11,672	00:00:10,343	00:00:01,328	0	ja
SSL	Sicher	00:00:12,000	00:00:11,062	00:00:00,937	0	ja
Spread	Sicher	00:00:35,891	00:00:33,906	00:00:01,984	0	ja
LSLS	Eher Sicher	00:00:33,094	00:00:31,984	00:00:01,109	0	ja
Pfeil	Sicher	00:00:23,984	00:00:23,140	00:00:00,843	0	ja
HoldMove	Sicher	00:01:17,438	00:00:38,188	00:00:01,671	1	nein
V	Sicher	00:00:07,093	00:00:06,531	00:00:00,562	0	ja
X	Sicher	00:00:41,703	00:00:40,641	00:00:01,062	0	ja

Tabelle B.10: Ergebnisse des Probanden GC4

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
LSLS	Eher Sicher	00:00:45,047	00:00:40,844	00:00:04,203	0	ja
V	Sicher	00:00:23,953	00:00:22,797	00:00:01,156	0	ja
Y	Sicher	00:00:17,531	00:00:15,609	00:00:01,921	0	ja
X	Eher Sicher	00:00:54,203	00:00:26,359	00:00:01,124	1	nein
HoldMove	Eher Unsicher	00:00:30,672	00:00:29,891	00:00:00,781	0	nein
SSL	Eher Sicher	00:00:20,265	00:00:19,500	00:00:00,765	0	ja
Pfeil	Unsicher	00:01:16,141	00:00:39,297	00:00:01,609	1	nein
Spread	Eher Sicher	00:00:22,437	00:00:21,328	00:00:01,109	0	ja

Tabelle B.11: Ergebnisse des Probanden GC5

Geste	Zuversicht	Gesamtzeit	Betrachtungszeit	Bearbeitungszeit	Aufrufe Spickzettel	Korrekt
SSL	Eher Sicher	00:00:46,141	00:00:43,688	00:00:02,453	0	ja
Spread	Eher Sicher	00:00:38,063	00:00:36,625	00:00:01,437	0	ja
V	Sicher	00:00:11,578	00:00:10,531	00:00:01,046	0	ja
X	Eher Sicher	00:00:27,062	00:00:25,437	00:00:01,624	0	ja
Y	Sicher	00:00:15,343	00:00:13,047	00:00:02,296	0	ja
Pfeil	Sicher	00:00:19,000	00:00:17,562	00:00:01,437	0	ja
LSLS	Sicher	00:00:39,406	00:00:36,438	00:00:02,968	0	ja
HoldMove	Eher Unsicher	00:00:41,266	00:00:36,750	00:00:04,515	0	ja

Tabelle B.12: Ergebnisse des Probanden GC6

Material zu Kap. 9: Prägnanz und Genauigkeit verschiedener Paradigmen zur Beschreibung von Gesten

C

C.1 Fragebogen

Vergleich unterschiedlicher Notationen zur Definition von Gesten

Einführung

Allgemeine Informationen

Geschlecht

Weiblich	Männlich	Keine Angabe
☐	☐	☐

Alter _____ Jahre

Studiengang

Angewandte Informatik	Wirtschaftsinformatik	Anderer Studiengang
☐	☐	_____

Semester _____

Angestrebter Abschluss

B.Sc	M.Sc	Anderer Abschluss
☐	☐	_____

Bevorzugtes Interaktionsparadigma

Bitte schätzen Sie ein, ob Sie – wenn die Möglichkeit besteht – eher Multi-Touch-Gesten oder klassische UIs mit Buttons u. ä. (WIMP) verwenden

Intensiv WIMP	Eher WIMP	Eher Multi-Touch	Intensiv Multi-Touch
☐	☐	☐	☐

Erfahrung mit Multi-Touch-Anwendungen

Bitte schätzen Sie ihre Erfahrung in der *Verwendung* multi-touch-basierter Anwendungen ein. Berücksichtigen Sie Erfahrungen mit allen Arten von Anwendungen, also Desktop, Mobile, Virtual/Augmented Reality, etc. Sie benutzen Multi-Touch-Gesten:

Gar nicht	Gelegentlich	Regelmäßig	Vorwiegend
☐	☐	☐	☐

Erfahrung mit der Entwicklung von Multi-Touch-Anwendungen

Bitte schätzen Sie ihre Erfahrung in der *Entwicklung* multi-touch-basierter Anwendungen ein. Berücksichtigen Sie Erfahrungen mit allen Arten von Anwendungen, also Desktop, Mobile, Virtual/Augmented Reality, etc.

keine Erfahrung	etwas Erfahrung (1-2 Anwendungen)	viel Erfahrung (3-5 Anwendungen)	sehr viel Erfahrung (> 5 Anwendungen)
☐	☐	☐	☐

1

Abbildung C.1: Fragebogen, Seite 1

Gestennotationen

Sie sehen im Folgenden eine Reihe von Videos, in denen Gesten ausgeführt werden. Zu jedem Video sehen Sie auf den nächsten Seiten die passenden Gestenbeschreibungen in unterschiedlichen Notationen. Alle Beschreibungen sind – im Rahmen der Möglichkeiten der Notationen – korrekt. Bitte bewerten Sie jede Beschreibung nach zwei Kriterien:

- Prägnanz: Wie umständlich wird die vorgeführte Geste dargestellt? Werden in der Notation Elemente verwendet, die unklar sind oder nicht zu den Kernelementen der Geste gehören?
- Genauigkeit: Wie exakt wird die vorgeführte Geste dargestellt? Sind alle notwendigen Informationen enthalten oder fehlen welche, die für das Verständnis der Geste wesentlich sind?

Nutzen Sie den freien Platz um die jeweilige Beschreibung, um ihre Gedanken und Gründe für ihre Bewertung zu notieren.
Beispiele sind:

- Fehlende Informationen über die Geste
- Unverständliche oder unklare Elemente
- Mehrdeutige Elemente
- etc.

Bitte blättern Sie immer erst nach Aufforderung auf die nächste Seite!

Abbildung C.2: Fragebogen, Seite 2

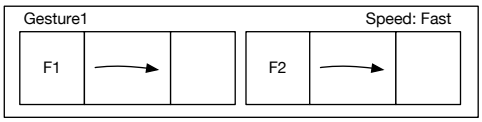
Geste 1

Notation A

Gesture1 = 2F(LINE_E)

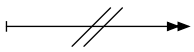
Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation B



Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation C



Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Persönliche Rangfolge der Notationen (beginnend mit der, die Ihnen am besten gefällt):

Bitte erst Nach Aufforderung umblättern.

Abbildung C.3: Fragebogen, Seite 3

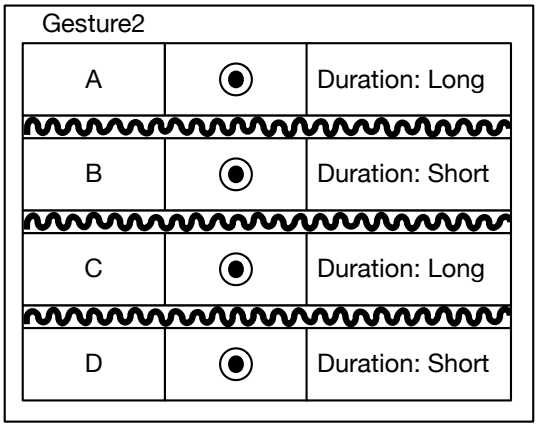
Geste 2

Notation A

Gesture2 = 1F(HOLD;POINT;HOLD;POINT)

Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation B



Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation C

Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Persönliche Rangfolge der Notationen (beginnend mit der, die Ihnen am besten gefällt):

Bitte erst Nach Aufforderung umblättern.

Abbildung C.4: Fragebogen, Seite 4

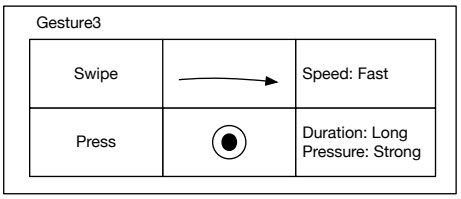
Geste 3

Notation A

Gesture3 = 1F(LINE_E,HOLD)

Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation B



Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation C

Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Persönliche Rangfolge der Notationen (beginnend mit der, die Ihnen am besten gefällt):

Bitte erst Nach Aufforderung umblättern.

Abbildung C.5: Fragebogen, Seite 5

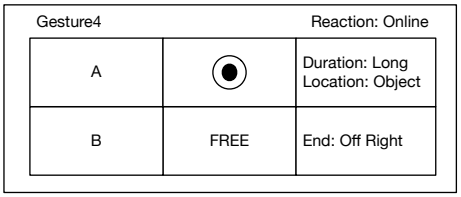
Geste 4

Notation A

Gesture4 = 1F(HOLD(Object),MOVE(Object)):[ON]

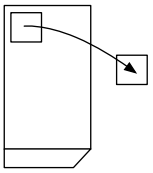
Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation B



Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation C



Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Persönliche Rangfolge der Notationen (beginnend mit der, die Ihnen am besten gefällt):

Bitte erst Nach Aufforderung umblättern.

Abbildung C.6: Fragebogen, Seite 6

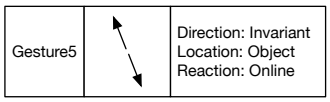
Geste 5

Notation A

Gesture5 = SPREAD[1F(LINE)*1F(LINE)](Object):[ON]

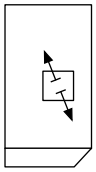
Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation B



Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Notation C



Prägnanz	unklar	---	--	-	+	++	+++	klar
Genauigkeit	ungenau	---	--	-	+	++	+++	genau

Persönliche Rangfolge der Notationen (beginnend mit der, die Ihnen am besten gefällt):

Bitte erst Nach Aufforderung umblättern.

Abbildung C.7: Fragebogen, Seite 7

Abschluss

Sie haben nun in der Veranstaltung die Struktur und die Details der einzelnen Notationen kennengelernt. Bitte bewerten Sie abschließend jede Notation nach folgenden Kriterien:

- Komplexität: Verständlichkeit des Aufbaus und Menge der syntaktischen Regeln
- Lesbarkeit: Verständlichkeit der verwendeten Begriffe und Symbole

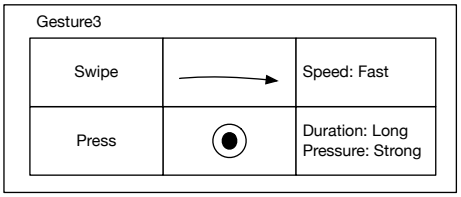
Gestennotationen

Notation A

Gesture3 = 1F(LINE_E,HOLD)

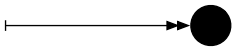
Komplexität	hoch	---	--	-	+	++	+++	niedrig
Genauigkeit	schlecht	---	--	-	+	++	+++	gut

Notation B



Komplexität	hoch	---	--	-	+	++	+++	niedrig
Genauigkeit	schlecht	---	--	-	+	++	+++	gut

Notation C



Komplexität	hoch	---	--	-	+	++	+++	niedrig
Genauigkeit	schlecht	---	--	-	+	++	+++	gut

War Ihnen eine der Notationen bekannt? Wenn ja, welche?

Abbildung C.8: Fragebogen, Seite 8

C.2 Ergebnisse aller Probanden

Proband	Alter	Studiengang	Semester	Abschluss	Bevorzugte Interaktion	Erfahrung Verwendung	Erfahrung Entwicklung
avoy	23	WI	8	B.Sc.	Eher Multi-Touch	Regelmäßig	Keine
bzqs	24	AI	5	B.Sc.	Intensiv Multi-Touch	Regelmäßig	Keine
duhj	k.A.	WI	k.A.	B.Sc.	Eher WIMP	Gelegentlich	Keine
eiku	20	WI	4	B.Sc.	Eher WIMP	Gelegentlich	Keine
epob	24	AI	10	B.Sc.	Eher WIMP	Regelmäßig	Keine
ezsr	26	AI	8	B.Sc.	Eher Multi-Touch	Regelmäßig	Keine
fcbh	21	WI	6	B.Sc.	Eher WIMP	Gelegentlich	Keine
flor	24	AI	5	B.Sc.	Eher Multi-Touch	Regelmäßig	Sehr viel
hkwv	25	AI	10	B.Sc.	Eher WIMP	Gelegentlich	Keine
hxde	27	AI	5	B.Sc.	Eher Multi-Touch	Gelegentlich	Keine
ipuc	k.A.	AI	5	B.Sc.	Intensiv WIMP	Regelmäßig	Keine
iuiu	21	WI	6	B.Sc.	Eher WIMP	Gelegentlich	Keine
kvno	30	AI	6	B.Sc.	Intensiv WIMP	Regelmäßig	Etwas
lkeq	21	AI	6	B.Sc.	Eher WIMP	Regelmäßig	Keine
lwpa	21	AI	6	B.Sc.	Eher WIMP	Gelegentlich	Keine
mcrm	22	WI	8	B.Sc.	Eher Multi-Touch	Regelmäßig	Keine
muih	21	WI	k.A.	B.Sc.	Eher WIMP	Regelmäßig	Keine
ndxo	22	AI	5	B.Sc.	Eher WIMP	Regelmäßig	Etwas
ogjz	23	AI	10	B.Sc.	Eher WIMP	Gar nicht	Keine
pecp	21	AI	6	B.Sc.	Eher WIMP	Gelegentlich	Keine
pqcw	21	WI	6	B.Sc.	Eher WIMP	Regelmäßig	Keine
pwqh	24	AI	8	B.Sc.	Eher WIMP	Regelmäßig	Keine
pxxs	21	AI	6	B.Sc.	Eher Multi-Touch	Regelmäßig	Keine
pyjk	22	Anderer	4	B.Sc.	Eher Multi-Touch	Regelmäßig	Keine
qgus	k.A.	WI	8	B.Sc.	Eher Multi-Touch	Regelmäßig	Keine
qotf	24	WI	6	B.Sc.	Eher WIMP	Regelmäßig	Keine
qyhp	k.A.	AI	k.A.	B.Sc.	Eher WIMP	Regelmäßig	Keine
sumj	k.A.	WI	k.A.	B.Sc.	Eher Multi-Touch	Regelmäßig	Keine
tkts	27	WI	7	B.Sc.	Eher WIMP	Regelmäßig	Keine
tnfn	27	WI	6	B.Sc.	Eher Multi-Touch	Regelmäßig	Sehr viel
tpkq	24	AI	10	B.Sc.	Eher WIMP	Gelegentlich	Keine
ueol	31	AI	20	B.Sc.	Intensiv WIMP	Gar nicht	Keine
uqfh	25	AI	6	B.Sc.	Eher Multi-Touch	Regelmäßig	Keine
vvgx	24	WI	7	B.Sc.	Eher WIMP	Gar nicht	Keine
xmjb	21	AI	6	B.Sc.	Eher Multi-Touch	Regelmäßig	Keine
xobj	21	AI	6	B.Sc.	Eher WIMP	Regelmäßig	Keine
yctp	20	WI	6	B.Sc.	Eher WIMP	Regelmäßig	Keine

Tabelle C.1: Demographische Daten der Probanden

Proband	GeForMT Prägnanz	GeForMT Genauigkeit	GC Prägnanz	GC Genauigkeit	Monox Prägnanz	Monox Genauigkeit	Favorit
avoy	+	--	+	+++	--	---	BAC
bzqs	---	---	--	+++	+++	+++	CBA
duhj	--	--	++	++	-	-	BCA
eiku	+	+	-	--	--	---	ABC
epob	--	--	+	+	-	+	BCA
ezsr	+	+	+++	+++	--	--	BAC
fcbh	+++	+++	+++	+++	--	--	k.A.
flor	-	--	+	+	+	-	BCA
hkwv	-	-	++	++	++	++	BCA
hxde	++	+	+	--	--	--	B
ipuc	---	---	---	---	---	---	k.A.
iuiu	+	---	++	+	+	--	BAC
kvno	--	+	++	-	+	+	CBA
lkeq	---	+	+	-	-	-	BAC
lwpa	---	+	--	-	--	--	BAC
mcrm	--	-	+++	++	---	+	BAC
muih	--	-	-	++	-	--	BAC
ndxo	-	-	+	+	++	++	CBA
ogjz	--	+	--	++	+	++	CBA
pecp	-	++	+	+	--	--	BAC
pqcw	+++	++	+++	+	--	--	BC
pwqh	--	--	+	++	+	+	BCA
pxxs	-	-	+	--	---	---	BAC
pyjk	+++	+++	+++	++	++	+++	B
qgus	++	++	-	-	---	---	ACB
qotf	+	---	++	+	+	--	BCA
qyhp	--	--	-	+	++	++	CBA
sumj	+++	+	---	---	+	+	ACB
tkts	+	++	+	--	-	--	ABC
tnfn	--	-	++	++	+	--	B
tpkq	-	+	--	++	-	+	CBA
ueol	+++	+++	+++	+++	+	+	BAC
uqfh	---	---	-	--	+	+++	CBA
vvgx	--	-	-	++	+	+	CBA
xmjb	---	---	-	-	--	--	k.A.
xobj	--	--	---	---	---	---	ABC
yctp	-	+	++	+++	-	+	BAC

Tabelle C.2: Ergebnisse der Geste Two-Finger-Swipe

Proband	GeForMT Prägnanz	GeForMT Genauigkeit	GC Prägnanz	GC Genauigkeit	Monox Prägnanz	Monox Genauigkeit	Favorit
avoy	+	+++	++	++	---	---	ABC
bzqs	++	-	--	++	+++	--	BCA
duhj	+++	+++	++	++	--	--	ABC
eiku	++	-	--	+	---	---	ABC
epob	++	+	+++	++	--	---	BAC
ezsr	+++	++	+++	+	--	---	ABC
fcbh	+	-	-	-	---	---	k.A.
flor	---	---	-	-	--	--	BCA
hkwv	+++	+++	++	++	--	--	ABC
hxde	+	+	++	+++	--	---	B
ipuc	-	-	-	-	-	-	k.A.
iuiu	++	++	++	++	---	---	BAC
kvno	+	+	+	++	---	---	BAC
lkeq	+	++	++	+++	---	--	BAC
lwpa	-	++	-	-	---	---	ABC
mcrm	-	+	+++	++	---	---	BAC
muih	+	-	+	+	---	---	ABC
ndxo	+	+	+++	+++	--	--	BAC
ogjz	+++	+++	--	+	---	+	ABC
pecp	++	+++	+	+	---	---	ABC
pqcw	+++	-	+	+++	---	---	BC
pwqh	++	++	-	-	---	---	ABC
pxxs	+++	++	+++	+++	---	---	BAC
pyjk	+++	+++	+++	-	+++	+++	A
qgus	++	++	+++	+++	---	---	BAC
qotf	++	++	--	+	---	---	ABC
qyhp	++	+	+++	+++	---	--	ABC
sumj	+++	+++	+++	+++	--	--	ABC
tkts	++	++	+++	++	---	---	BAC
tnfn	++	++	+	+	---	---	A
tpkq	+	+++	+	+	---	---	ABC
ueol	+	+	--	+	--	--	BAC
uqfh	++	+++	+++	+++	---	---	BAC
vvgx	+++	+++	-	+	---	---	ABC
xmjb	+++	++	-	+++	---	---	ABC
xobj	+	+	---	---	---	---	ABC
yctp	+	+++	++	++	---	---	BAC

Tabelle C.3: Ergebnisse der Geste LSLS

Proband	GeForMT Prägnanz	GeForMT Genauigkeit	GC Prägnanz	GC Genauigkeit	Monox Prägnanz	Monox Genauigkeit	Favorit
avoy	+	++	+++	+++	--	--	BAC
bzqs	+	--	-	++	++	++	CBA
duhj	+	+	++	++	+	+	BAC
eiku	++	-	++	+	++	-	BAC
epob	++	+++	++	++	+	++	ABC
ezsr	++	-	++	++	++	++	BCA
fcbh	+	++	-	++	+	+++	k.A.
flor	---	---	+	++	+	+	BCA
hkwv	-	-	++	++	+++	+++	CBA
hxde	++	-	+	++	++	+	B
ipuc	-	-	-	-	-	-	CBA
iuiu	--	--	++	++	+	-	BAC
kvno	-	++	++	-	++	++	CAB
lkeq	-	+	++	++	+	++	BAC
lwpa	+	--	++	++	--	+	BAC
mcrm	+	--	+++	+	--	-	BAC
muih	+	-	++	++	+	--	BAC
ndxo	--	--	+	+	++	++	CBA
ogjz	++	--	-	+++	+++	+++	CBA
pecp	++	++	++	+++	++	+	BAC
pqcw	+++	-	+	+++	-	+	BC
pwqh	-	-	+	+	++	++	CBA
pxxs	++	+++	+	-	++	-	ABC
pyjk	++	-	+	+	--	--	B
qgus	-	-	+	+	+	+	BCA
qotf	++	++	+++	+++	--	--	BAC
qyhp	--	---	++	++	+	--	BCA
sumj	+++	+	-	+++	---	+	ACB
tkts	+	++	++	++	-	--	BAC
tnfn	+	-	++	++	+	--	B
tpkq	++	+++	+++	+++	--	-	BAC
ueol	+++	+++	+++	+++	-	+	BAC
uqfh	-	---	++	+	+	+	BCA
vvgx	-	-	-	++	+++	-	CBA
xmjb	+	+	-	--	++	++	CAB
xobj	++	++	+	+	--	--	ABC
yctp	+	+	+++	+++	++	+	BCA

Tabelle C.4: Ergebnisse der Geste Swipe-And-Press

Proband	GeForMT Prägnanz	GeForMT Genauigkeit	GC Prägnanz	GC Genauigkeit	Monox Prägnanz	Monox Genauigkeit	Favorit
avoy	+	--	-	+	+	+++	CBA
bzqs	--	--	--	+++	+++	+++	CBA
duhj	--	--	+++	+++	+	+	BCA
eiku	++	+	++	++	+	+	BAC
epob	+	+	---	--	++	++	CAB
ezsr	+++	-	-	---	+++	+++	CAB
fcbh	++	--	+	+	++	-	k.A.
flor	---	---	+	++	++	++	CBA
hkwv	+	+	++	++	+++	+++	CBA
hxde	+	-	++	+	++	++	C
ipuc	+	+	+++	+++	++	++	CBA
iuiu	--	--	++	++	-	-	BCA
kvno	--	---	++	+++	++	--	BCA
lkeq	+	-	-	+	++	-	CAB
lwpa	++	-	-	++	++	-	CAB
mcrm	+	--	++	+	+++	+	CBA
muih	+	--	+	+	++	+	BCA
ndxo	-	-	++	++	+	+	BCA
ogjz	++	+	++	+++	+++	++	CBA
pecp	--	-	-	+	++	+	CBA
pqcw	-	--	+	+++	++	++	CBA
pwqh	--	--	-	-	++	+	CBA
pxxs	+	-	+	+	+++	+++	CBA
pyjk	+++	+	+++	+++	+	+	A
qgus	+	+	++	++	+++	+++	CAB
qotf	++	+	+++	+++	+++	++	BCA
qyhp	---	---	+++	+++	+	-	BAC
sumj	+++	+	---	+	-	++	ACB
tkts	+	--	++	+++	+++	+++	CBA
tnfn	+	-	+++	++	+++	+	B
tpkq	-	-	+++	+++	+	+	BCA
ueol	-	+	+++	+++	+++	++	BCA
uqfh	--	---	++	+++	+++	+	BCA
vvgx	--	--	+	-	++	-	CBA
xmjb	--	--	-	++	++	++	CBA
xobj	-	-	---	---	+++	+++	CAB
yctp	++	-	++	+++	++	++	BCA

Tabelle C.5: Ergebnisse der Geste Drag-Out

Proband	GeForMT Prägnanz	GeForMT Genauigkeit	GC Prägnanz	GC Genauigkeit	Monox Prägnanz	Monox Genauigkeit	Favorit
avoy	++	++	+	++	-	--	ABC
bzqs	+	+	k.A.	k.A.	k.A.	k.A.	CAB
duhj	+	+	+	+	++	++	CBA
eiku	-	+	++	++	++	+	BCA
epob	---	---	++	++	+	++	BCA
ezsr	+++	+++	++	++	+++	+++	ACB
fcbh	+	-	++	++	--	-	k.A.
flor	---	---	++	+	+++	+++	CBA
hkwv	---	---	+++	+++	+++	+++	CBA
hxde	---	+	++	++	-	-	B
ipuc	+++	+++	+++	+++	+++	+++	k.A.
iuiu	-	-	++	++	+	+	BCA
kvno	---	---	+++	+++	+++	+++	BCA
lkeq	--	+	++	++	++	+	BCA
lwpa	--	-	-	+++	+++	+	C
mcrm	-	-	++	-	++	+	CBA
muih	--	-	++	++	+	+	BCA
ndxo	+	+	++	++	+++	+++	CBA
ogjz	--	-	+	++	++	+	CBA
pecp	-	+	++	++	++	++	BCA
pqcw	-	-	-	++	++	++	CBA
pwqh	---	-	--	-	-	+	CAB
pxxs	-	-	+++	+++	+++	+++	CBA
pyjk	--	--	-	-	--	--	k.A.
qgus	+	+	-	-	++	++	CAB
qotf	+	-	++	++	+++	+++	CBA
qyhp	+	+	++	++	+	--	BAC
sumj	+++	+	+	+++	++	--	ABC
tkts	-	-	+++	+++	+++	++	BCA
tnfn	+	+	+++	+++	++	+	B
tpkq	--	-	+	++	+++	++	CBA
ueol	--	+	++	+++	+++	+++	CBA
uqfh	+	++	+++	+++	+++	+	BCA
vvgx	-	++	++	++	++	+	BCA
xmjb	--	+	++	+	++	+	BCA
xobj	+	+	--	--	-	-	ACB
yctp	+	+	+++	+++	+++	++	CBA

Tabelle C.6: Ergebnisse der Geste Spread

Proband	GeForMT	GCs	Monox	Gesamtpunkte
avoy	11	12	7	30
bzqs	6	10	14	30
duhj	8	13	9	30
eiku	11	13	6	30
epob	9	12	9	30
ezsr	11	10	9	30
fcbh	0	0	0	0
flor	5	13	12	30
hkwv	7	11	12	30
hxde	0	12	3	15
ipuc	2	4	6	12
iuiu	8	15	7	30
kvno	7	12	11	30
lkeq	9	13	8	30
lwpa	9	9	9	27
mcrm	8	13	9	30
muih	9	14	7	30
ndxo	6	12	12	30
ogjz	7	10	13	30
pecp	9	13	8	30
pqcw	2	13	12	27
pwqh	8	10	12	30
pxxs	9	12	9	30
pyjk	6	6	0	12
qgus	10	9	11	30
qotf	8	13	9	30
qyhp	9	13	8	30
sumj	15	7	8	30
tkts	9	13	8	30
tnfn	3	12	0	15
tpkq	8	12	10	30
ueol	8	14	8	30
uqfh	6	14	10	30
vvgx	7	11	12	30
xmjb	7	8	9	24
xobj	14	8	8	30
yctp	7	14	9	30

Tabelle C.7: Gesamtpunkte

Material zu Kap. 10: Einsatz in der Praxis

D

D.1 Einführung in GestureCards

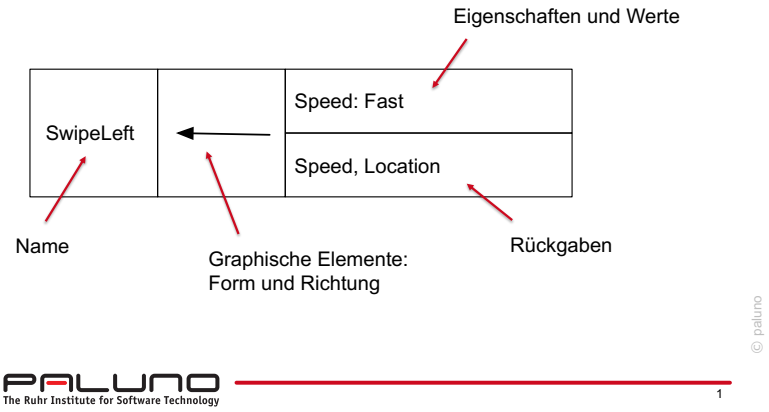


Abbildung D.1: Einführung in GestureCards, Seite 1

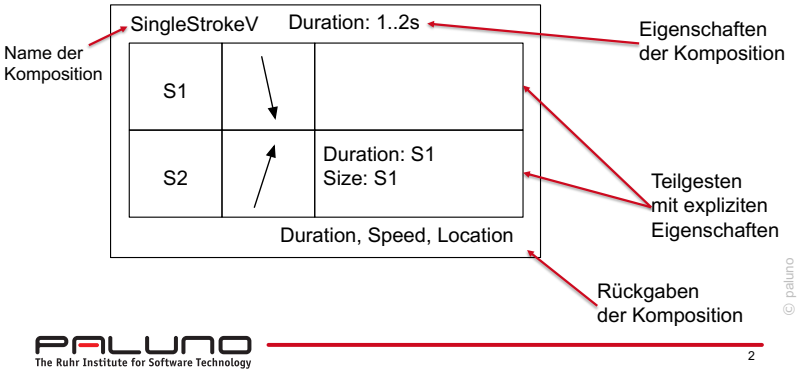
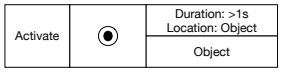


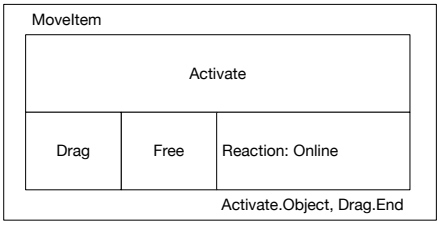
Abbildung D.2: Einführung in GestureCards, Seite 2

Abbildung D.3: Einführung in GestureCards, Seite 3

GestureCards

Wiederverwendung





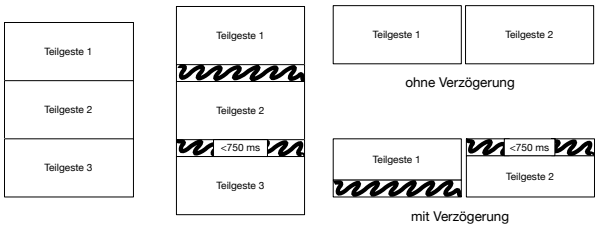
© paluno

Abbildung D.4: Einführung in GestureCards, Seite 4

GestureCards

Zeitlicher Bezug

Zeitliche Relationen zwischen Teilgesten werden durch die Positionierung der Teilgesten zueinander ausgedrückt. Eine geschlängelte Linie definiert Zeiträume, in denen die Oberfläche nicht berührt wird.



Sequentielle Ausführung in einem Zug (Single-Stroke-Geste)

Sequentielle Ausführung in mehreren Zügen (Multi-Stroke-Geste)

Parallele Ausführung mehrerer Züge (Multi-Touch-Geste)

© paluno

D.2 Fragebogen

Fragebogen zur Studie
GestureCards & AugIR

Vielen herzlichen Dank für Ihre Teilnahme an dieser kurzen Befragung!

Bitte bewerten Sie Ihre Eindrücke,
indem Sie in nachfolgender Tabelle Zutreffendes ankreuzen:

		1	2	3	4	5
1	Das Verstehen vordefinierter Gesten auf analogen Whiteboards war…	Intuitiv				Nicht intuitiv
2		Schnell				Zeitlich aufwändig
3	Das Verstehen vordefinierter Gesten im AugIR war…	Intuitiv				Nicht intuitiv
4		Schnell				Zeitlich aufwändig
5	Im Vergleich zu analogen Whiteboards war das Verstehen vordefinierter Gesten im AugIR…	Intuitiver				Weniger intuitiv
6		Schneller				Zeitlich aufwändiger
7	Im Vergleich zu analogen Whiteboards hat der AugIR mich beim Verstehen vordefinierter Gesten…	Unterstützt				Behindert
8	Die Definition neuer Gesten auf analogen Whiteboards war…	Leicht				Schwierig
9		Schnell				Zeitlich aufwändig

(Fortsetzung auf nächster Seite…)

Abbildung D.5: Fragebogen, Seite 1

10	Die Definition neuer Gesten im AugIR war…	Leicht				Schwierig
11		Schnell				Zeitlich aufwändig
12	Im Vergleich zu analogen Whiteboards war die Definition neuer Gesten im AugIR…	Leichter				Schwieriger
13		Schneller				Zeitlich aufwändiger
14	Im Vergleich zu analogen Whiteboards hat der AugIR mich bei der Definition neuer Gesten…	Unterstützt				Behindert
15	Die Erstellung von UI-Skizzen auf analogen Whiteboards war…	Leicht				Schwierig
16		Schnell				Zeitlich aufwändig
17	Die Erstellung von UI-Skizzen im AugIR war…	Leicht				Schwierig
18		Schnell				Zeitlich aufwändig
19	Im Vergleich zu analogen Whiteboards war die Erstellung von UI-Skizzen im AugIR…	Leichter				Schwieriger
20		Schneller				Zeitlich aufwändiger
21	Im Vergleich zu analogen Whiteboards hat der AugIR mich bei der Erstellung von UI-Skizzen…	Unterstützt				Behindert

(Fortsetzung auf nächster Seite…)

Abbildung D.6: Fragebogen, Seite 2

22	Für das Verstehen der Dialogübergänge war der Simulationsmodus im AugIR…	Hilfreich				Nicht hilfreich
23	Für das Verstehen der Gesten war der Simulationsmodus im AugIR…	Hilfreich				Nicht hilfreich
24	Für das Verstehen der Gesten war die Animation der Gestenbewegung im AugIR…	Hilfreich				Nicht hilfreich
25	Insgesamt empfinde ich die Bedienung des AugIRs als…	Intuitiv				Nicht intuitiv
26	Den Umgang mit dem AugIR zu erlernen war…	Leicht				Schwierig
27	Insgesamt halte ich GestureCards zur Beschreibung von Gesten für…	Nützlich				Nicht nützlich
28	Insgesamt halte ich den AugIR für…	Nützlich				Nicht nützlich
29	Insgesamt würde ich Folgendes präferieren…	Analoge GestureCards				Digitale GestureCards
30		Analoge Whiteboards				AugIR

(Fortsetzung auf nächster Seite…)

Abbildung D.7: Fragebogen, Seite 3

31	Folgende Dinge haben mir an der AugIR-Software und den digitalen GestureCards gut gefallen:
32	Folgende Dinge haben mir an der AugIR-Software und den digitalen GestureCards nicht gefallen:
33	Folgende Dinge würde ich mir für eine zukünftige Version wünschen:

Abbildung D.8: Fragebogen, Seite 4

D.3 Ergebnisse aller Probanden

Proband	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	2	1	1	2	2	2	2	2	2	2	4	5	3	2
2	3	3	3	3	3	3	3	3	3	1	1	1	1	2	4
3	1	1	1	1	3	3	3	4	4	2	2	2	2	2	2
4	2	1	2	2	1	2	2	3	3	3	3	2	2	2	4
5	2	2	1	1	1	1	1	1	2	1	1	1	1	1	2
6	1	1	3	2	3	3	3	1	1	3	2	4	4	4	1
7	4	5	2	1	2	1	1	2	4	1	2	1	1	1	1
8	2	3	2	3	2	3	2	3	2	3	3	3	4	2	2
9	1	2	1	1	3	2	2	2	4	2	1	2	2	2	1
10	2	2	1	2	1	2	1	1	2	3	2	2	3	2	1
11	2	1	2	1	2	3	3	1	1	3	4	4	4	4	1
12	1	2	1	1	2	3	2	1	1	2	3	4	5	3	1
13	2	2	1	1	1	1	1	1	1	1	1	3	3	2	1
14	3	4	1	1	1	1	1	2	2	1	2	1	1	1	1

Tabelle D.1: Ergebnisse aller Probanden, Fragen 1-15

Proband	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	2	2	2	4	4	2	2	3	1	3	2	2	2	4	2
2	4	1	1	1	1	1	1	1	1	1	1	2	1	5	5
3	2	1	1	2	2	1	1	1	1	1	1	1	1	4	4
4	3	2	2	3	2	2	1	1	1	2	2	2	1	4	4
5	2	1	1	1	1	1	1	1	1	1	1	1	1	5	5
6	1	3	3	4	4	4	2	3	2	3	2	2	2	2	2
7	1	1	1	3	3	3	1	1	1	2	1	1	1	5	5
8	2	2	2	2	2	2	2	2	2	2	2	2	2	4	4
9	1	1	2	4	3	3	1	1	1	2	2	1	1	4	4
10	1	1	1	3	2	2	1	1	1	2	2	1	1	4	4
11	1	1	1	2	3	3	3	4	2	3	2	1	2	1	1
12	1	1	1	3	4	3	1	1	3	2	1	1	1	2	4
13	1	3	3	3	3	2	1	1	1	1	1	1	1	4	2
14	2	1	1	3	3	2	1	1	1	1	1	1	1	5	5

Tabelle D.2: Ergebnisse aller Probanden, Fragen 16-30

Material zu Kap. 12: Vergleich graphischer Repräsentationen für räumliche Gesten

E

E.1 Einführung in GestureCards

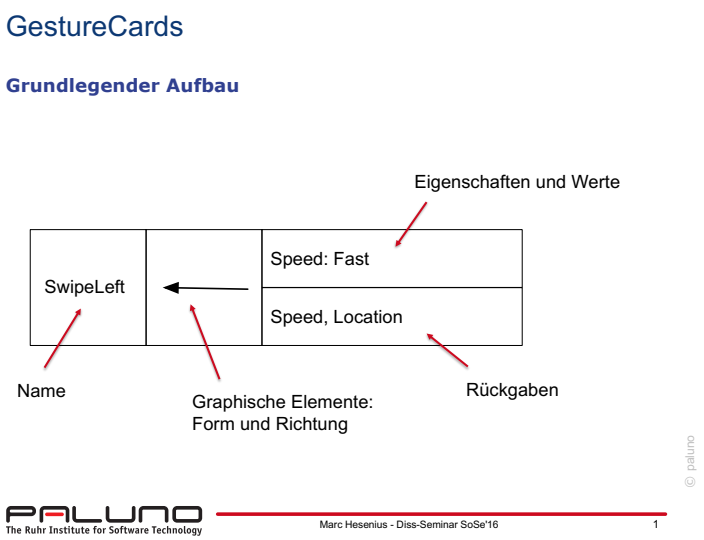


Abbildung E.1: Einführung in GestureCards, Seite 1

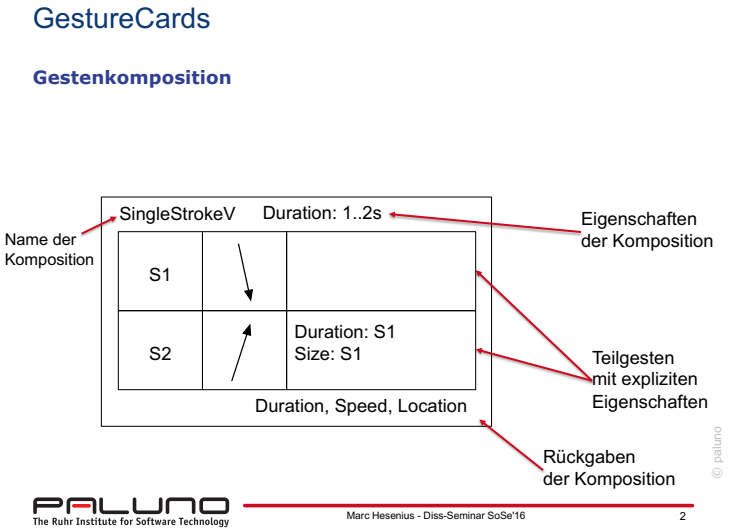


Abbildung E.2: Einführung in GestureCards, Seite 2

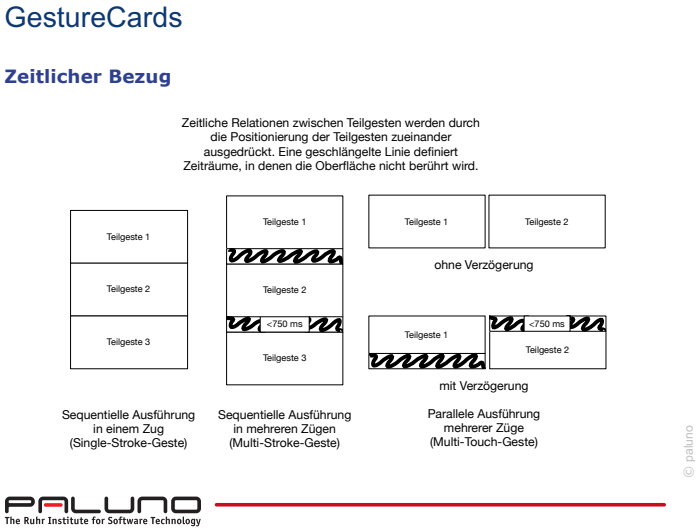


Abbildung E.3: Einführung in GestureCards, Seite 3

E.2 Ergebnisse aller Probanden

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Sehr Sicher	0	7806	ja
Conductor	Sicher	0	9637	ja
DragNDrop	Sicher	0	10209	ja
Gamma	Sicher	0	8151	ja
Phi	Sicher	0	7482	ja
Present	Sicher	0	9164	ja
Spread	Sicher	0	9755	nein
U	Sicher	0	7843	ja
X	Sicher	2	35737	ja

Tabelle E.1: Ergebnisse des Probanden asch, Gruppe Hand

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Sehr Sicher	1	13539	ja
Conductor	Sicher	2	33073	nein
DragNDrop	Sicher	2	36402	ja
Gamma	Unsicher	2	22790	nein
Phi	Sicher	1	13270	ja
Present	Sicher	1	15589	ja
Spread	Unsicher	0	11988	ja
U	Sicher	1	15886	ja
X	Sicher	3	33184	nein

Tabelle E.2: Ergebnisse des Probanden dbhc, Gruppe Hand

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Unsicher	1	25448	ja
Conductor	Sehr Sicher	0	13284	ja
DragNDrop	Sicher	0	22457	ja
Gamma	Sehr Sicher	0	10158	ja
Phi	Sehr Sicher	0	10951	ja
Present	Sicher	0	14522	nein
Spread	Sicher	0	15730	ja
U	Sicher	0	18251	ja
X	Sehr Sicher	0	15430	ja

Tabelle E.3: Ergebnisse des Probanden dmfy, Gruppe Arrows

Tabelle E.4: Ergebnisse des Probanden ghpp, Gruppe Hand

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Sehr Sicher	0	10964	nein
Conductor	Sicher	2	23503	ja
DragNDrop	Sehr Sicher	1	25267	ja
Gamma	Sehr Unsicher	3	32944	ja
Phi	Sehr Sicher	1	13829	ja
Present	Sehr Sicher	0	8076	ja
Spread	Sehr Sicher	0	11361	ja
U	Unsicher	1	25136	nein
X	Sehr Sicher	3	43043	ja

Tabelle E.5: Ergebnisse des Probanden idyx, Gruppe Hand

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Sehr Sicher	1	12676	ja
Conductor	Unsicher	6	42302	nein
DragNDrop	Sehr Sicher	0	25833	ja
Gamma	Sicher	5	32564	ja
Phi	Sicher	1	17721	ja
Present	Sehr Sicher	0	10073	nein
Spread	Sicher	2	18925	ja
U	Sicher	2	18388	ja
X	Sicher	1	23351	nein

Tabelle E.6: Ergebnisse des Probanden kohn, Gruppe Arrows

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Unsicher	1	38441	nein
Conductor	Sicher	0	19729	ja
DragNDrop	Sicher	1	37819	ja
Gamma	Sicher	0	12884	ja
Phi	Sicher	0	15448	ja
Present	Unsicher	0	37571	ja
Spread	Sicher	0	13894	ja
U	Sicher	1	23264	nein
X	Sicher	0	31708	ja

Tabelle E.7: Ergebnisse des Probanden nyyc, Gruppe Hand

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Sehr Sicher	0	10277	nein
Conductor	Unsicher	4	39863	nein
DragNDrop	Sicher	1	52646	ja
Gamma	Sicher	5	43242	ja
Phi	Sicher	1	21147	ja
Present	Sicher	1	13604	ja
Spread	Sicher	1	21598	ja
U	Sehr Sicher	0	11560	ja
X	Sehr Sicher	2	35379	ja

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Unsicher	1	41468	nein
Conductor	Unsicher	0	22174	ja
DragNDrop	Unsicher	1	57252	nein
Gamma	Sehr Sicher	0	14347	ja
Phi	Sehr Sicher	0	23510	ja
Present	Sehr Unsicher	1	41586	nein
Spread	Sehr Sicher	0	13890	ja
U	Sehr Sicher	0	18337	nein
X	Sehr Sicher	0	26980	ja

Tabelle E.8: Ergebnisse des Probanden wblk, Gruppe Arrows

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Sehr Unsicher	1	48146	nein
Conductor	Sehr Sicher	1	22944	ja
DragNDrop	Sehr Sicher	0	28025	ja
Gamma	Unsicher	1	23342	ja
Phi	Sehr Sicher	0	14692	ja
Present	Unsicher	0	24627	nein
Spread	Sicher	1	22802	ja
U	Sicher	0	30947	ja
X	Sehr Sicher	0	23603	ja

Tabelle E.9: Ergebnisse des Probanden wqyx, Gruppe Arrows

Geste	Zuversicht	Replays	Dauer (ms)	Korrekt
Boxen	Sehr Unsicher	0	24342	nein
Conductor	Sicher	0	13291	ja
DragNDrop	Sehr Unsicher	0	21819	nein
Gamma	Sicher	0	10796	ja
Phi	Sicher	0	8533	ja
Present	Unsicher	0	17670	nein
Spread	Sicher	0	13873	ja
U	Sicher	0	10090	ja
X	Unsicher	0	23480	ja

Tabelle E.10: Ergebnisse des Probanden yfdh, Gruppe Arrows

Verzeichnisse

Literatur

Agogué, M., Le Masson, P., Dalmasso, C., Houdé, O. & Cassotti, M. (2015). „Resisting Classical Solutions: The Creative Mind of Industrial Designers and Engineers.“ In: *Psychology of Aesthetics, Creativity, and the Arts* 9.3, S. 313–318. ISSN: 1931-390X. DOI: 10.1037/a0039414.

Aigner, R., Wigdor, D., Benko, H., Haller, M., Lindbauer, D., Ion, A., Zhao, S. & Koh, J. (2012). *Understanding Mid-Air Hand Gestures: A Study of Human Preferences in Usage of Gesture Types for HCI*. Techn. Ber. Microsoft.

Altakrouri, B. & Schrader, A. (2014). „Describing Movements for Motion Gestures“. In: *Proceedings of the Workshop on Engineering Gestures for Multimodal Interfaces*. EGMI '14, CEUR Workshop Proceedings, vol. 1190. Rome, Italy.

– (2015). „Challenging Documentation Practices for Interactions in Natural User Interfaces“. In: *1st Workshop on Large-scale and Model-based Interactive Systems: Approaches and Challenges*. Hrsg. von R. Seiger, B. Altakrouri, A. Schrader & T. Schlegel. LMIS '15, CEUR Workshop Proceedings, S. 29–34.

Amini, S. & Li, Y. (2013). „CrowdLearner: Rapidly Creating Mobile Recognizers Using Crowdsourcing“. In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST '13. New York, NY, USA: ACM Press, S. 163–172. ISBN: 9781450322683. DOI: 10.1145/2501988.2502029.

Android Developers (oJ[a]). *Android GestureLibraries*. http://developer.android.com/reference/android/gesture/GestureLibraries.html [Zugriff 2015-11-11].

– (oJ[b]). *Android Motion Event*. https://developer.android.com/reference/android/view/MotionEvent.html [Zugriff 2015-12-01].

– (oJ[c]). *Android Touch Devices*. https://source.android.com/devices/input/touch-devices.html [Zugriff 2015-12-05].

– (oJ[d]). *Detecting Common Gestures*. http://developer.android.com/training/gestures/detector.html [Zugriff 2015-11-11].

– (oJ[e]). *GestureDetector*. https://developer.android.com/reference/android/view/GestureDetector.html [Zugriff 2017-10-27].

– (oJ[f]). *GestureDetector.OnGestureListener*. https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener.html [Zugriff 2016-05-20].

Anthony, L. & Wobbrock, J. O. (2010). „A Lightweight Multistroke Recognizer for User Interface Prototypes“. In: *Proceedings of Graphics Interface 2010*. GI '10. Ottawa, Ontario, Canada: Canadian Information Processing Society, S. 245–252. ISBN: 978-1-56881-712-5.

– (2012). „$N-protractor: A Fast and Accurate Multistroke Recognizer“. In: *Proceedings of Graphics Interface 2012*. GI '12. Toronto, Ont., Canada: Canadian Information Processing Society, S. 117–120. ISBN: 978-1-4503-1420-6.

Appert, C. & Zhai, S. (2009). „Using Strokes as Command Shortcuts: Cognitive Benefits and Toolkit Support“. In: *Proceedings of the 27th International Conference on Human Factors in Computing Systems*. CHI '09. New York, NY, USA: ACM Press. ISBN: 9781605582467. DOI: 10.1145/1518701.1519052.

Apple Developer (2015). *Instance Property applicationSupportsShakeToEdit*. https://developer.apple.com/documentation/uikit/uiapplication/1623127-applicationsupportsshaketoedit [Zugriff 2017-12-13].

– (2017a). *Human Interface Guidelines*. https://developer.apple.com/ios/human-interface-guidelines/user-interaction/gestures/ [Zugriff 2017-12-13].

Apple Developer (2017b). *UIGestureRecognizer*. https://developer.apple.com/documentation/uikit/uigesturerecognizer [Zugriff 2017-12-13].

– (oJ[a]). *UIScreenEdgePanGestureRecognizer*. https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIScreenEdgePanGestureRecognizer_class/index.html [Zugriff 2016-06-18].

– (oJ[b]). *UITouch Class Reference*. https://developer.apple.com/library/ios/documentation/UIKit/Reference/UITouch_Class/ [Zugriff 2015-12-05].

Apple Support (2016). *Use Multi-Touch gestures on your Mac*. https://support.apple.com/en-us/HT204895 [Zugriff 2016-06-05].

Balsamiq Blog (2011). *Why We Aren't Doing Deep Interaction in Mockups*. https://blog.balsamiq.com/why-we-arent-doing-interaction/ [Zugriff 2017-08-13].

Baltes, S. & Diehl, S. (2014). „Sketches and Diagrams in Practice". In: *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE '14. Hong Kong, China: ACM, S. 530–541. ISBN: 978-1-4503-3056-5. DOI: 10.1145/2635868.2635891.

Battocchi, A., Ben-Sasson, A., Esposito, G., Gal, E., Pianesi, F., Tomasini, D., Venuti, P., Weiss, P. & Zancanaro, M. (2010). „Collaborative Puzzle Game: A Tabletop Interface for Fostering Collaborative Skills in Children with Autism Spectrum Disorders". In: *Journal of Assistive Technologies* 4.1, S. 4–13. DOI: 10.5042/jat.2010.0040.

Benko, H., Wilson, A. D. & Baudisch, P. (2006). „Precise Selection Techniques for Multi-Touch Screens". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. New York, NY, USA: ACM Press. ISBN: 1595933727. DOI: 10.1145/1124772.1124963.

Bevan, C. & Fraser, D. S. (2016). „Different Strokes for Different Folks? Revealing the Physical Characteristics of Smartphone Users from their Swipe Gestures". In: *International Journal of Human-Computer Studies* 88, S. 51–61. ISSN: 10715819. DOI: 10.1016/j.ijhcs.2016.01.001.

Bitkom (2017). *Anteil der Smartphone-Nutzer in Deutschland nach Altersgruppe im Jahr 2017*. https://de.statista.com/statistik/daten/studie/459963/umfrage/anteil-der-smartphone-nutzer-in-deutschland-nach-altersgruppe/ [Zugriff 2018-01-27].

Blandford, A. (2013). „Engineering Works: What is (and is not) "Engineering" for Interactive Computer Systems?" In: *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '13. New York, NY, USA: ACM Press. ISBN: 9781450321389. DOI: 10.1145/2494603.2480342.

Böhmer, M., Hecht, B., Schöning, J., Krüger, A. & Bauer, G. (2011). „Falling Asleep with Angry Birds, Facebook and Kindle: A Large Scale Study on Mobile Application Usage". In: *Proceedings of the 13th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI '11. Stockholm, Sweden: ACM, S. 47–56. ISBN: 978-1-4503-0541-9. DOI: 10.1145/2037373.2037383.

Bolt, R. A. (1980). „"Put-that-there": Voice and Gesture at the Graphics Interface". In: *ACM SIGGRAPH Computer Graphics* 14.3, S. 262–270. ISSN: 00978930. DOI: 10.1145/965105.807503.

Bomsdorf, B. & Blum, R. (2014). „Early Prototyping of 3D-Gesture Interaction within the Presentation-Gesture-Dialog Design Space". In: *Human-Computer Interaction. Advanced Interaction Modalities and Techniques*. Hrsg. von M. Kurosu. Cham: Springer International Publishing, S. 12–23. ISBN: 978-3-319-07230-2.

Book, M., Gruhn, V. & Striemer, R. (2017). *Erfolgreiche agile Projekte: Pragmatische Kooperation und faires Contracting*. Springer Vieweg.

Boring, S., Ledo, D., Chen, X., Marquardt, N., Tang, A. & Greenberg, S. (2012). „The Fat Thumb: Using the Thumb's Contact Size for Single-handed Mobile Interaction". In: *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*. MobileHCI '12. New York, NY, USA: ACM, S. 39–48. ISBN: 978-1-4503-1105-2. DOI: 10.1145/2371574.2371582.

Bortz, J. (2004). *Statistik für Human- und Sozialwissenschaftler*. 6. Aufl. Springer. ISBN: 354021271X.

Bosch, J (2016). „Speed, Data, and Ecosystems: The Future of Software Engineering“. In: *IEEE Software* 33.1, S. 82–88. ISSN: 0740-7459. DOI: 10.1109/MS.2016.14.

Bourier, G. (2012). *Beschreibende Statistik: Praxisorientierte Einführung mit Aufgaben und Lösungen*. 10. Aufl. Springer Gabler. ISBN: 3834932701.

Bragdon, A., Nelson, E., Li, Y. & Hinckley, K. (2011). „Experimental Analysis of Touch-Screen Gesture Designs in Mobile Environments“. In: *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems*. CHI '11. New York, NY, USA: ACM Press, S. 403. ISBN: 9781450302289. DOI: 10.1145/1978942.1979000.

Brandl, P., Forlines, C., Wigdor, D., Haller, M. & Shen, C. (2008). „Combining and Measuring the Benefits of Bimanual Pen and Direct-Touch Interaction on Horizontal Interfaces“. In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '08. New York, NY, USA: ACM Press, S. 154. ISBN: 9781605581415. DOI: 10.1145/1385569.1385595.

Bull, P. E. (1987). *Posture & Gesture*. Amsterdam: Pergamon. ISBN: 978-0-08-033971-9. DOI: https://doi.org/10.1016/B978-0-08-033971-9.50006-4.

Buxton, B. (2007). *Sketching User Experiences: Getting the Design Right and the Right Design (Interactive Technologies)*. Morgan Kaufmann. ISBN: 0123740371.

– (2014). *Multi-Touch Systems that I Have Known and Loved*. http://www.billbuxton.com/multitouchOverview.html [Zugriff 2016-01-07].

Cao, X., Wilson, A. D., Balakrishnan, R., Hinckley, K. & Hudson, S. E. (2008). „ShapeTouch: Leveraging Contact Shape on Interactive Surfaces“. In: *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*. TABLETOP '08, S. 129–136. DOI: 10.1109/TABLETOP.2008.4660195.

Chi, T. (2012). *Rapid prototyping Google Glass*. http://ed.ted.com/lessons/rapid-prototyping-google-glass-tom-chi [Zugriff 2017-06-15].

Choi, E., Kim, H. & Chung, M. K. (2014). „A Taxonomy and Notation Method for Three-Dimensional Hand Gestures“. In: *International Journal of Industrial Ergonomics* 44.1, S. 171–188. ISSN: 0169-8141. DOI: 10.1016/j.ergon.2013.10.011.

Ciger, J., Gutierrez, M., Vexo, F. & Thalmann, D. (2003). „The Magic Wand“. In: *Proceedings of the 18th Spring Conference on Computer Graphics*. SCCG '03. New York, NY, USA: ACM Press. ISBN: 158113861X. DOI: 10.1145/984952.984972.

Cirelli, M. & Nakamura, R. (2014). „A Survey on Multi-touch Gesture Recognition and Multi-touch Frameworks“. In: *Proceedings of the 9th ACM International Conference on Interactive Tabletops and Surfaces*. ITS '14. New York, NY, USA: ACM Press, S. 35–44. ISBN: 9781450325875. DOI: 10.1145/2669485.2669509.

Cirelli, M., Nakamura, R. & Filgueiras, L. (2015). „The MiTable Multi-Touch Gestures Engine“. In: *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. ITS '15. New York, NY, USA: ACM Press, S. 425–428. ISBN: 9781450338998. DOI: 10.1145/2817721.2823483.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. 2. Aufl. Routledge. ISBN: 0805802835.

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Colgan, A. (2014). *How Does the Leap Motion Controller Work?* http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/ [Zugriff 2018-08-21].

Colley, A. & Häkkilä, J. (2014). „Exploring Finger Specific Touch Screen Interaction for Mobile Phone User Interfaces“. In: *Proceedings of the 26th Australian Conference on Human-Computer Interaction*. OzCHI '14. New York, New York, USA: ACM Press, S. 539–548. ISBN: 9781450306539. DOI: 10.1145/2686612.2686699.

Coutaz, J. & Caelen, J. (1991). „A Taxonomy for Multimedia and Multimodal User Interfaces“. In: *Proceedings of the ERCIM Workshop on User Interfaces and Multimedia*, S. 143–147.

Coyette, A., Kieffer, S. & Vanderdonckt, J. (2007). „Multi-Fidelity Prototyping of User Interfaces“. In: *Human-Computer Interaction – INTERACT 2007*. Hrsg. von C. Baranauskas, P. Palanque, J. Abascal & S. D. J. Barbosa. Lecture Notes in Computer Science, vol 4662. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 150–164. ISBN: 978-3-540-74796-3. DOI: 10.1007/978-3-540-74796-3_16.

Dang, C. & André, E. (2011). „Usage and Recognition of Finger Orientation for Multi-Touch Tabletop Interaction“. In: *Human-Computer Interaction – INTERACT 2011*. Hrsg. von P. Campos, N. Graham, J. Jorge, N. Nunes, P. Palanque & M. Winckler. Lecture Notes in Computer Science, vol 6948. Springer Berlin Heidelberg, S. 409–426. ISBN: 978-3-642-23764-5. DOI: 10.1007/978-3-642-23765-2{_}28.

de Sousa, A. M. F. & Morimoto, C. H. (2015). „5* Magic Wand: An RGBD Camera-Based 5 DoF User Interface for 3D Interaction“. In: *XVII Symposium on Virtual and Augmented Reality*. SVR '15, S. 15–22. DOI: 10.1109/SVR.2015.10.

de Souza Alcantara, T., Denzinger, J., Ferreira, J. & Maurer, F. (2012). „Learning Gestures for Interacting with Low-fidelity Prototypes“. In: *Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering*. RAISE '12. Zurich, Switzerland: IEEE Press, S. 32–36. ISBN: 978-1-4673-1753-5.

de Souza Alcantara, T., Ferreira, J. & Maurer, F. (2013). „Interactive Prototyping of Tabletop and Surface Applications“. In: *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '13. New York, NY, USA: ACM Press. ISBN: 9781450321389. DOI: 10.1145/2494603.2480313.

Dietz, P. & Leigh, D. (2001). „DiamondTouch“. In: *Proceedings of the 14th annual ACM symposium on User interface software and technology*. UIST '01. New York, NY, USA: ACM Press. ISBN: 158113438X. DOI: 10.1145/502348.502389.

Dorau, R. (2011). *Emotionales Interaktionsdesign: Gesten und Mimik interaktiver Systeme*. Springer. ISBN: 3642031005.

D'Orazio, T., Marani, R., Renò, V. & Cicirelli, G. (2016). „Recent Trends in Gesture Recognition: How Depth Data Has Improved Classical Approaches“. In: *Image and Vision Computing* 52.Supplement C, S. 56 –72. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2016.05.007.

Döring, T., Kern, D., Marshall, P., Pfeiffer, M., Schöning, J., Gruhn, V. & Schmidt, A. (2011). „Gestural Interaction on the Steering Wheel“. In: *Proceedings of the 2011 Annual Conference on Human factors in Computing Systems*. CHI '11. New York, NY, USA: ACM Press, S. 483. ISBN: 9781450302289. DOI: 10.1145/1978942.1979010.

Dumas, B., Lalanne, D. & Ingold, R. (2010). „Description Languages for Multimodal Interaction: A Set of Guidelines and its Illustration with SMUIML“. In: *Journal on Multimodal User Interfaces* 3.3, S. 237–247. ISSN: 1783-7677, 1783-8738. DOI: 10.1007/s12193-010-0043-3.

Echtler, F. & Butz, A. (2012). „GISpL: Gestures Made Easy“. In: *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. TEI '12. New York, NY, USA: ACM, S. 233–240.

Echtler, F. & Klinker, G. (2008). „A Multitouch Software Architecture“. In: *Proceedings of the 5th Nordic Conference on Human-Computer Interaction: Building Bridges*. NordiCHI '08. Lund, Sweden: ACM, S. 463–466. ISBN: 978-1-59593-704-9. DOI: 10.1145/1463160.1463220.

Echtler, F., Klinker, G. & Butz, A. (2010). „Towards a Unified Gesture Description Language“. In: *Proceedings of the 13th International Conference on Humans and Computers*. HC '10. University of Aizu Press, S. 177–182.

English, W. K., Engelbart, D. C. & Berman, M. L. (1967). „Display-Selection Techniques for Text Manipulation“. In: *IEEE Transactions on Human Factors in Electronics* HFE-8.1, S. 5–15.

Enström, W., Dennis, J., Lynch, B. & Schlei, K. (2015). „Musical Notation for Multi-Touch Interfaces“. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. Hrsg. von E. Berdahl & J. Allison. NIME ’15. Baton Rouge, Louisiana, USA: Louisiana State University, S. 83–86.

Favilla, S. & Pedell, S. (2014). „Touch Screen Collaborative Music: Designing NIME for Older People with Dementia“. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. Hrsg. von B. Caramiaux, K. Tahiroglu, R. Fiebrink & A. Tanaka. NIME ’14. London, United Kingdom: Goldsmiths, University of London, S. 35–39.

Figueiredo, L. S., Pinheiro, M., Neto, E. V., Chaves, T. & Teichrieb, V. (2015). „Sci-Fi Gestures Catalog“. In: *Human-Computer Interaction – INTERACT 2015*. Hrsg. von J. Abascal, S. Barbosa, M. Fetter, T. Gross, P. Palanque & M. Winckler. Lecture Notes in Computer Science, vol 9297. Cham: Springer International Publishing, S. 395–411. ISBN: 978-3-319-22668-2. DOI: 10.1007/978-3-319-22668-2_30.

Figueiredo, L. S., Livshits, B., Molnar, D. & Veanes, M. (2016). „Prepose: Privacy, Security, and Reliability for Gesture-Based Programming“. In: *2016 IEEE Symposium on Security and Privacy*. SP ’16, S. 122–137. DOI: 10.1109/SP.2016.16.

Fish, J. & Scrivener, S. (1990). „Amplifying the Mind’s Eye: Sketching and Visual Cognition“. In: *Leonardo* 23.1, S. 117–126. ISSN: 0024094X, 15309282.

Forlines, C., Wigdor, D., Shen, C. & Balakrishnan, R. (2007). „Direct-Touch vs. Mouse Input for Tabletop Displays“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’07. New York, NY, USA: ACM Press. ISBN: 9781595935939. DOI: 10.1145/1240624.1240726.

Fothergill, S., Mentis, H., Kohli, P. & Nowozin, S. (2012). „Instructing People for Training Gestural Interactive Systems“. In: *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems*. CHI ’12. New York, NY, USA: ACM Press. ISBN: 9781450310154. DOI: 10.1145/2207676.2208303.

Frisch, M. & Dachselt, R. (2013). „Kombinierte Multi-Touch und Stift-Interaktion: Ein Gesten-Set zum Editieren von Diagrammen“. In: *Multi-Touch*. Hrsg. von T. Schlegel. Xpert.press. Springer Berlin Heidelberg, S. 89–116. ISBN: 978-3-642-36112-8. DOI: 10.1007/978-3-642-36113-5{_}5.

Frisch, M., Heydekorn, J. & Dachselt, R. (2009). „Investigating Multi-Touch and Pen Gestures for Diagram Editing on Interactive Surfaces“. In: *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ITS ’09. New York, NY, USA: ACM Press. ISBN: 9781605587332. DOI: 10.1145/1731903.1731933.

– (2010). „Diagram Editing on Interactive Displays Using Multi-Touch and Pen Gestures“. In: *Diagrammatic Representation and Inference*. Hrsg. von A. Goel, M. Jamnik & N. Narayanan. Bd. 6170. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 182–196. ISBN: 978-3-642-14599-5. DOI: 10.1007/978-3-642-14600-8{_}18.

Ganhör, R. & Spreicer, W. (2014). „Monox: Extensible Gesture Notation for Mobile Devices“. In: *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services*. MobileHCI ’14. New York, NY, USA: ACM Press, S. 203–212. ISBN: 9781450330046. DOI: 10.1145/2628363.2628394.

Ghomi, E., Faure, G., Huot, S., Chapuis, O. & Beaudouin-Lafon, M. (2012). „Using Rhythmic Patterns as an Input Method“. In: *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems*. CHI ’12. New York, NY, USA: ACM Press, S. 1253–1262. ISBN: 9781450310154. DOI: 10.1145/2207676.2208579.

Giusti, L., Zancanaro, M., Gal, E. & Weiss, P. L. T. (2011). „Dimensions of Collaboration on a Tabletop Interface for Children with Autism Spectrum Disorder“. In: *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems*. CHI ’11. New York, NY, USA: ACM Press, S. 3295. ISBN: 9781450302289. DOI: 10.1145/1978942.1979431.

Goldberg, D. & Richardson, C. (1993). „Touch-typing with a Stylus“. In: *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*. CHI '93. Amsterdam, The Netherlands: ACM, S. 80–87. ISBN: 0-89791-575-5. DOI: 10.1145/169059.169093.

Greenberg, S., Carpendale, S., Marquardt, N. & Buxton, B. (2011). *Sketching User Experiences: The Workbook*. Morgan Kaufmann. ISBN: 978-0-12-381959-8.

Grieve-Smith, A. (2002). „SignSynth: A Sign Language Synthesis Application Using Web3D and Perl“. In: *Gesture and Sign Language in Human-Computer Interaction*. Hrsg. von I. Wachsmuth & T. Sowa. Bd. 2298. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 134–145. ISBN: 978-3-540-43678-2. DOI: 10.1007/3-540-47873-6{_}14.

Gutierrez, M., Cirelli, M., Nakamura, R. & Filgueiras, L. (2013). „Evaluating Paper Prototype for Tabletop Collaborative Game Applications“. In: *Entertainment Computing – ICEC 2013*. Hrsg. von J. Anacleto, E. Clua, F. da Silva, S. Fels & H. Yang. Lecture Notes in Computer Science, vol 8215. Springer Berlin Heidelberg, S. 168–171. ISBN: 978-3-642-41105-2. DOI: 10.1007/978-3-642-41106-9{_}23.

Hachaj, T. & Ogiela, M. R. (2014). „Rule-Based Approach to Recognizing Human Body Poses and Gestures in Real Time“. In: *Multimedia Systems* 20.1, S. 81–99. ISSN: 1432-1882. DOI: 10.1007/s00530-013-0332-2.

Haller, K. (2013). „Mobile Testing“. In: *SIGSOFT Software Engineering Notes* 38.6, S. 1–8. ISSN: 0163-5948. DOI: 10.1145/2532780.2532813.

Harrison, C. & Hudson, S. E. (2008). „Scratch Input: Creating Large, Inexpensive, Unpowered and Mobile Finger Input Surfaces“. In: *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*. UIST '08. Monterey, CA, USA: ACM, S. 205–208. ISBN: 978-1-59593-975-3. DOI: 10.1145/1449715.1449747.

Harrison, C., Schwarz, J. & Hudson, S. E. (2011). „TapSense: Enhancing Finger Interaction on Touch Surfaces“. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. New York, NY, USA: ACM, S. 627–636. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047279.

Hartson, R. & Pyla, P. S. (2012). *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*. Morgan Kaufmann.

Hassenzahl, M. & Tractinsky, N. (2006). „User Experience – A Research Agenda“. In: *Behaviour & Information Technology* 25.2, S. 91–97. ISSN: 0144-929X. DOI: 10.1080/01449290500330331.

Heo, S. & Lee, G. (2011). „Force Gestures: Augmenting Touch Screen Gestures with Normal and Tangential Forces“. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. New York, NY, USA: ACM, S. 621–626. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047278.

Heo, S., Gu, J. & Lee, G. (2014). „Expanding Touch Input Vocabulary by Using Consecutive Distant Taps“. In: *Proceedings of the 32nd Annual ACM Conference on Human factors in Computing Systems*. CHI '14. New York, NY, USA: ACM Press, S. 2597–2606. ISBN: 9781450324731. DOI: 10.1145/2556288.2557234.

Herold, J. & Stahovich, T. F. (2012). „The 1¢ Recognizer: A Fast, Accurate, and Easy-to-Implement Handwritten Gesture Recognition Technique“. In: *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*. SBIM '12. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, S. 39–46. ISBN: 978-3-905674-42-2.

Herot, C. F. & Weinzapfel, G. (1978). „One-Point Touch Input of Vector Information for Computer Displays“. In: *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*. Bd. 12. SIGGRAPH '78 3. ACM, S. 210–216. DOI: 10.1145/965139.807392.

Hesenius, M., Griebe, T., Gries, S. & Gruhn, V. (2014a). „Automating UI Tests for Mobile Applications with Formal Gesture Descriptions“. In: *Proceedings of the 16th ACM International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI ’14. New York, NY, USA: ACM, S. 213–222. ISBN: 9781450327718. DOI: 10.1145/2628363.2628391.

Hesenius, M., Griebe, T. & Gruhn, V. (2014b). „Towards a Behavior-Oriented Specification and Testing Language for Multimodal Applications“. In: *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS ’14. New York, NY, USA: ACM, S. 117–122. ISBN: 9781450327251. DOI: 10.1145/2607023.2610278.

Hesenius, M. & Gruhn, V. (2016). „Introducing GestureCards: A Prototyping Gesture Notation“. In: *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*. NordiCHI ’16. Gothenburg, Sweden: ACM, 111:1–111:6. ISBN: 978-1-4503-4763-1. DOI: 10.1145/2971485.2996746.

Hesenius, M., Kleffmann, M. & Gruhn, V. (2016). „Sketching Gesture-Based Applications in a Collaborative Working Environment with Wall-Sized Displays“. In: *2016 International Conference on Collaboration Technologies and Systems*. CTS ’16. Orlando, FL, USA: IEEE Conference Publications, S. 327–336. DOI: 10.1109/cts.2016.0066.

Hesenius, M., Sternal, S. & Gruhn, V. (2017). „A Multi-Touch-Recognizer for GestureCards“. In: *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS ’17. Lisbon, Portugal: ACM, S. 75–80. ISBN: 978-1-4503-5083-9. DOI: 10.1145/3102113.3102132.

Hesenius, M., Meyer, O., Börsting, I. & Gruhn, V. (2018). „Don’t Panic! Guiding Pedestrians in Autonomous Traffic with Augmented Reality“. In: *Adjunct Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI ’18. Barcelona, Spain: ACM. DOI: 10.1145/3236112.3236148.

Hinckley, K. & Wigdor, D. (2012). „Input Technologies and Techniques“. In: *Human-Computer Interaction Handbook*. Hrsg. von J. A. Jacko. 3. Aufl. Taylor & Francis. Kap. 6, S. 95–132.

Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H. & Buxton, B. (2010). „Pen + Touch = New Tools“. In: *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. UIST ’10. New York, NY, USA: ACM Press, S. 27. ISBN: 9781450302715. DOI: 10.1145/1866029.1866036.

Hinckley, K., Heo, S., Pahud, M., Holz, C., Benko, H., Sellen, A., Banks, R., O’Hara, K., Smyth, G. & Buxton, W. (2016). „Pre-Touch Sensing for Mobile Interaction“. In: *Proceedings of the 2016 Conference on Human Factors in Computing Systems*. CHI ’16. San Jose, California, USA: ACM, S. 2869–2881. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858095.

Hinrichs, U. & Carpendale, S. (2011). „Gestures in the Wild: Studying Multi-Touch Gesture Sequences on Interactive Tabletop Exhibits“. In: *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems*. CHI ’11. New York, NY, USA: ACM Press, S. 3023. ISBN: 9781450302289. DOI: 10.1145/1978942.1979391.

Hoggan, E., Williamson, J., Oulasvirta, A., Nacenta, M., Kristensson, P. O. & Lehtiö, A. (2013). „Multi-Touch Rotation Gestures: Performance and Ergonomics“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. New York, NY, USA: ACM, S. 3047–3050. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481423.

Horodniczy, D. & Cooperstock, J. R. (2017). „Free the Hands! Enhanced Target Selection via a Variable-Friction Shoe“. In: *Proceedings of the 2017 Conference on Human Factors in Computing Systems*. CHI ’17. Denver, Colorado, USA: ACM, S. 255–259. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025625.

Höst, M., Regnell, B. & Wohlin, C. (2000). „Using Students as Subjects – A Comparative Study of Students and Professionals in Lead-Time Impact Assessment“. In: *Empirical Software Engineering* 5.3, S. 201–214. ISSN: 1573-7616. DOI: 10.1023/A:1026586415054.

Hoste, L. & Signer, B. (2014). „Criteria, Challenges and Opportunities for Gesture Programming Languages“. In: *Proceedings of the Workshop on Engineering Gestures for Multimodal Interfaces*. EGMI '14, CEUR Workshop Proceedings, vol. 1190, S. 22–29.

Hoste, L., Dumas, B. & Signer, B. (2011). „Mudra: A Unified Multimodal Interaction Framework“. In: *Proceedings of the 13th International Conference on Multimodal Interfaces*. ICMI '11. Alicante, Spain: ACM, S. 97–104. ISBN: 978-1-4503-0641-6. DOI: 10.1145/2070481.2070500.

Hutchinson Guest, A. (1998). *Choreographics: A Comparison of Dance Notation Systems from the Fifteenth Century to the Present*. Routledge. ISBN: 9057000032.

– (2005). *Labanotation: The System of Analyzing and Recording Movement*. Routledge. ISBN: 0415965624.

Hutchinson Guest, A. & Kolff, J. (2002). *Hands, Fingers (Advanced Labanotation)*. Dance Books Ltd. ISBN: 1852730862.

Hwang, S., Bianchi, A. & Wohn, K.-y. (2013). „VibPress: Estimating Pressure Input Using Vibration Absorption on Mobile Devices“. In: *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*. MobileHCI '13. New York, NY, USA: ACM, S. 31–34. ISBN: 978-1-4503-2273-7. DOI: 10.1145/2493190.2493193.

IDC (2017). *Smartphone Growth Expected to Remain Positive as Shipments Forecast to Grow to 1.7 Billion in 2021*. https://www.idc.com/getdoc.jsp?containerId=prUS43010517 [Zugriff 2017-12-13].

Janssen, J. & Laatz, W. (2013). *Statistische Datenanalyse mit SPSS: Eine anwendungsorientierte Einführung in das Basissystem und das Modul Exakte Tests*. 8. Aufl. Springer. ISBN: 3642325068.

Jedlitschka, A., Ciolkowski, M. & Pfahl, D. (2008). „Reporting Experiments in Software Engineering“. In: *Guide to Advanced Empirical Software Engineering*. Hrsg. von F. Shull, J. Singer & D. I. K. Sjøberg. London: Springer London, S. 201–228. ISBN: 978-1-84800-044-5. DOI: 10.1007/978-1-84800-044-5_8.

Jerald, J. (2016). *The VR Book: Human-Centered Design for Virtual Reality*. New York, NY, USA: Association for Computing Machinery & Morgan & Claypool. ISBN: 978-1-97000-112-9.

Johnson, E. A. (1965). „Touch Display: A Novel Input/Output Device for Computers“. In: *Electronics Letters* 1.8, S. 219–220. ISSN: 0013-5194. DOI: 10.1049/el:19650200.

– (1967). „Touch Displays: A Programmed Man-Machine Interface“. In: *Ergonomics* 10.2, S. 271–277. DOI: 10.1080/00140136708930868. eprint: https://doi.org/10.1080/00140136708930868.

Johnson, G., Gross, M. D., Hong, J. & Yi-Luen Do, E. (2009). „Computational Support for Sketching in Design: A Review“. In: *Foundations and Trends in Human–Computer Interaction* 2.1, S. 1–93. ISSN: 1551-3955.

Justinmind Support (oJ[a]). *How to create prototypes with rich events*. https://www.justinmind.com/support/how-to-create-prototypes-with-rich-events/ [Zugriff 2017-08-13].

– (oJ[b]). *How to simulate mobile gestures in your app wireframes*. https://www.justinmind.com/support/how-to-simulate-mobile-gestures-in-your-app-wireframes/ [Zugriff 2017-08-13].

Kaasinen, E., Roto, V., Hakulinen, J., Heimonen, T., Jokinen, J. P. P., Karvonen, H., Keskinen, T., Koskinen, H., Lu, Y., Saariluoma, P., Tokkonen, H. & Turunen, M. (2015). „Defining User Experience Goals to Guide the Design of Industrial Systems“. In: *Behaviour & Information Technology* 34.10, S. 976–991. DOI: 10.1080/0144929X.2015.1035335.

Kaltenbrunner, M. (2014). *TUIO 2.0 Protocol Specification*. http://www.tuio.org/?tuio20 [Zugriff 2016-01-10].

Kaltenbrunner, M., Bovermann, T., Bencina, R. & Costanza, E. (2005). „TUIO: A Protocol for Table-Top Tangible User Interfaces“. In: *Proc. of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation*. Vannes, France.

Kammer, D. (2014). „Formalisierung gestischer Eingabe für Multitouch-Systeme“. Diss. Technische Universität Dresden.

Kammer, D., Wojdziak, J., Keck, M., Groh, R. & Taranko, S. (2010). „Towards a Formalization of Multi-Touch Gestures“. In: *ACM International Conference on Interactive Tabletops and Surfaces*. ITS ’10. New York, NY, USA: ACM, S. 49–58. ISBN: 978-1-4503-0399-6. DOI: 10.1145/1936652.1936662.

Kammer, D., Henkens, D., Wojdziak, J. & Freitag, G. (2011). „Formalization and Combination of Touch and Point Interaction“. In: *Workshop on Engineering Patterns for Multitouch Interfaces*.

Kammer, D., Henkens, D. & Groh, R. (2012). „GeForMTjs: A JavaScript Library Based on a Domain Specific Language for Multi-touch Gestures“. In: *Web Engineering*. Hrsg. von M. Brambilla, T. Tokuda & R. Tolksdorf. ICWE 2012, Lecture Notes in Computer Science, vol. 7387. Springer Berlin Heidelberg, S. 444–447. ISBN: 978-3-642-31752-1. DOI: 10.1007/978-3-642-31753-8{_}40.

Kammer, D., Henkens, D., Henzen, C. & Groh, R. (2015). „Gesture Formalization for Multitouch“. In: *Software: Practice and Experience* 45.4, S. 527–548. ISSN: 1097-024X. DOI: 10.1002/spe.2247.

Khan, V.-J., Pekelharing, M. & Desle, N. (2012). „Efficient Navigation in Virtual Environments: A Comparative Study of Two Interaction Techniques: The Magic Wand vs. the Human Joystick“. In: *Proceedings of the 4th International Conference on Intelligent Human Computer Interaction*. IHCI ’12, S. 1–5. DOI: 10.1109/IHCI.2012.6481795.

Khandkar, S. H. & Maurer, F. (2010a). „A Domain Specific Language to Define Gestures for Multi-touch Applications“. In: *Proceedings of the 10th Workshop on Domain-Specific Modeling*. DSM ’10. Reno, Nevada: ACM, 2:1–2:6. ISBN: 978-1-4503-0549-5. DOI: 10.1145/2060329.2060339.

– (2010b). „A Language to Define Multi-touch Interactions“. In: *ACM International Conference on Interactive Tabletops and Surfaces*. ITS ’10. Saarbrücken, Germany: ACM, S. 269–270. ISBN: 978-1-4503-0399-6. DOI: 10.1145/1936652.1936710.

Khandkar, S. H., Sohan, S. M., Sillito, J. & Maurer, F. (2010). „Tool Support for Testing Complex Multi-touch Gestures“. In: *ACM International Conference on Interactive Tabletops and Surfaces*. ITS ’10. Saarbrücken, Germany: ACM, S. 59–68. ISBN: 978-1-4503-0399-6. DOI: 10.1145/1936652.1936663.

Kim, J.-W. & Nam, T.-J. (2013). „EventHurdle: Supporting Designers’ Exploratory Interaction Prototyping with Gesture-based Sensors“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. New York, NY, USA: ACM, S. 267–276. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2470691.

Kim, J.-W., Kim, H.-J. & Nam, T.-J. (2016). „M.Gesture: An Acceleration-Based Gesture Authoring System on Multiple Handheld and Wearable Devices“. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. New York, NY, USA: ACM, S. 2307–2318. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858358.

Kin, K., Hartmann, B., DeRose, T. & Agrawala, M. (2012a). „Proton++: A Customizable Declarative Multitouch Framework“. In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST ’12. New York, NY, USA: ACM, S. 477–486. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380176.

– (2012b). „Proton: Multitouch Gestures as Regular Expressions“. In: *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing System*. CHI ’12. New York, NY, USA: ACM, S. 2885–2894. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2208636.2208694.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K. & Rosenberg, J. (2002). „Preliminary Guidelines for Empirical Research in Software Engineering“. In: *IEEE Transactions on Software Engineering* 28.8, S. 721–734. ISSN: 0098-5589. DOI: 10.1109/TSE.2002.1027796.

Kleffmann, M. (2014). „AugIR: The Conceptual Design and Evaluation of an Augmented Interaction Room“. In: *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*. ASE '14. New York, NY, USA: ACM, S. 883–886. ISBN: 978-1-4503-3013-8. DOI: 10.1145/2642937.2653467.

Kleffmann, M., Book, M. & Gruhn, V. (2014). „Supporting Collaboration of Heterogeneous Teams in an Augmented Team Room“. In: *Proceedings of the 6th International Workshop on Social Software Engineering*. SSE 2014. New York, NY, USA: ACM, S. 9–16. ISBN: 978-1-4503-3227-9. DOI: 10.1145/2661685.2661688.

Kleffmann, M., Hesenius, M. & Gruhn, V. (2015a). „Connecting UI and Business Processes in a Collaborative Sketching Environment“. In: *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '15. Duisburg, Germany: ACM, S. 200–205. ISBN: 978-1-4503-3646-8. DOI: 10.1145/2774225.2775076.

Kleffmann, M., Röhl, S., Gruhn, V. & Book, M. (2015b). „Establishing and Navigating Trace Links between Elements of Informal Diagram Sketches“. In: *2015 IEEE/ACM 8th International Symposium on Software and Systems Traceability*, S. 1–7. DOI: 10.1109/SST.2015.8.

Kleffmann, M., Röhl, S., Book, M. & Gruhn, V. (2018). „Evaluation of a Traceability Approach for Informal Freehand Sketches“. In: *Automated Software Engineering* 25.1, S. 1–43. ISSN: 1573-7535. DOI: 10.1007/s10515-017-0221-6.

Kordts, B., Altakrouri, B. & Schrader, A. (2015). „Capturing and Analysing Movement Using Depth Sensors and Labanotation“. In: *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '15. New York, NY, USA: ACM Press, S. 132–141. ISBN: 9781450336468. DOI: 10.1145/2774225.2774842.

Kratz, S. & Rohs, M. (2010). „A $3 Gesture Recognizer – Simple Gesture Recognition for Devices Equipped with 3D Acceleration Sensors“. In: *Proceedings of the 15th International Conference on Intelligent User Interfaces*. IUI '10. New York, New York, USA: ACM Press, S. 341. ISBN: 9781605585154. DOI: 10.1145/1719970.1720026.

Kray, C., Nesbitt, D., Dawson, J. & Rohs, M. (2010). „User-Defined Gestures for Connecting Mobile Phones, Public Displays, and Tabletops“. In: *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*. MobileHCI '10. New York, NY, USA: ACM Press. ISBN: 9781605588353. DOI: 10.1145/1851600.1851640.

Kristensson, P.-O. & Zhai, S. (2004). „SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers“. In: *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*. UIST '04. New York, NY, USA: ACM, S. 43–52. ISBN: 1-58113-957-8. DOI: 10.1145/1029632.1029640.

Kruger, R., Carpendale, S., Scott, S. D. & Greenberg, S. (2003). „How People Use Orientation on Tables: Comprehension, Coordination and Communication“. In: *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*. GROUP '03. New York, NY, USA: ACM Press, S. 369. ISBN: 1581136935. DOI: 10.1145/958160.958219.

Kuusinen, K. & Mikkonen, T. (2014). „On Designing UX for Mobile Enterprise Apps“. In: *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. EUROMICRO '14. IEEE, S. 221–228. ISBN: 978-1-4799-5795-8. DOI: 10.1109/SEAA.2014.17.

Landay, J. A. & Myers, B. A. (1996). „Sketching Storyboards to Illustrate Interface Behaviors“. In: *Conference Companion on Human Factors in Computing Systems*. CHI '96. Vancouver, British Columbia, Canada: ACM, S. 193–194. ISBN: 0-89791-832-0. DOI: 10.1145/257089.257257.

Laufs, U. & Ruff, C. (2013). „Aufbau von Multi-Touch-Systemen“. In: *Multi-Touch: Interaktion durch Berührung*. Hrsg. von T. Schlegel. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 145–152. ISBN: 978-3-642-36113-5. DOI: 10.1007/978-3-642-36113-5_7.

Law, E. L.-C., Roto, V., Hassenzahl, M., Vermeeren, A. P.O. S. & Kort, J. (2009). „Understanding, Scoping and Defining User Experience: A Survey Approach“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. New York, NY, USA: ACM, S. 719–728. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518813.

Leap Motion Developer (oJ[a]). *Creating Your Own Hand Prefabs*. https://developer.leapmotion.com/documentation/v2/unity/unity/Unity_MakeDiscreteHand.html [Zugriff 2016-01-08].

– (oJ[b]). *Fingers*. https://developer.leapmotion.com/documentation/cpp/devguide/Leap_Pointables.html [Zugriff 2016-01-08].

– (oJ[c]). *Frame*. https://developer.leapmotion.com/documentation/v2/csharp/api/Leap.Frame.html [Zugriff 2017-12-13].

– (oJ[d]). *Gestures*. https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Gestures.html [Zugriff 2016-01-08].

– (oJ[e]). *Hand Assets*. https://developer.leapmotion.com/documentation/cpp/unity/Unity_Hand_Assets.html [Zugriff 2016-01-08].

Lebourque, T & Gibet, S (1999). „A Complete System for the Specification and the Generation of Sign Language Gestures“. In: *Gesture-Based Communication in Human-Computer Interaction*. Hrsg. von A. Braffort, R. Gherbi, S. Gibet, D. Teil & J. Richardson. Bd. 1739. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 227–238. ISBN: 978-3-540-66935-7. DOI: 10.1007/3-540-46616-9{_}20.

Lee, S., Buxton, W. & Smith, K. (1985). „A Multi-Touch Three Dimensional Touch-Sensitive Tablet“. In: *Proceedings of the 1985 ACM Annual Conference on Human Factors in Computing Systems*. CHI '85. ACM, NY, New York.

Liu, C., Chapuis, O., Beaudouin-Lafon, M. & Lecolinet, E. (2017). „CoReach: Cooperative Gestures for Data Manipulation on Wall-Sized Displays“. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. New York, NY, USA: ACM, S. 6730–6741. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025594.

Long Jr., A. C., Landay, J. A., Rowe, L. A. & Michiels, J. (2000). „Visual Similarity of Pen Gestures“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '00. The Hague, The Netherlands: ACM, S. 360–367. ISBN: 1-58113-216-6. DOI: 10.1145/332040.332458.

Losson, O. & Vannobel, J.-M. (1999). „Sign Specification and Synthesis“. In: *Gesture-Based Communication in Human-Computer Interaction*. Hrsg. von A. Braffort, R. Gherbi, S. Gibet, D. Teil & J. Richardson. Bd. 1739. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 239–251. ISBN: 978-3-540-66935-7. DOI: 10.1007/3-540-46616-9{_}21.

Low, S., Sugiura, Y., Lo, D. & Inami, M. (2014). „Pressure Detection on Mobile Phone by Camera and Flash“. In: *Proceedings of the 5th Augmented Human International Conference*. AH '14. New York, NY, USA: ACM, 11:1–11:4. ISBN: 978-1-4503-2761-9. DOI: 10.1145/2582051.2582062.

Lü, H., Fogarty, J. A. & Li, Y. (2014). „Gesture Script: Recognizing Gestures and Their Structure Using Rendering Scripts and Interactively Trained Parts“. In: *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*. CHI '14. New York, NY, USA: ACM Press, S. 1685–1694. ISBN: 9781450324731. DOI: 10.1145/2556288.2557263.

Ludbrook, J. (2008). „Analysis of 2x2 Tables of Frequencies: Matching Test to Experimental Design“. In: *International Journal of Epidemiology* 37.6, S. 1430–1435. DOI: 10.1093/ije/dyn162.

Lunney, G. H. (1970). „Using Analysis of Variance with a Dichotomous Dependent Variable: An Empirical Study“. In: *Journal of Educational Measurement* 7.4, S. 263–269. ISSN: 00220655, 17453984.

Mangano, N., Baker, A. & Hoek, A. van der (2008). „Calico: A Prototype Sketching Tool for Modeling in Early Design“. In: *Proceedings of the 2008 International Workshop on Models in Software Engineering*. MiSE '08. Leipzig, Germany: ACM, S. 63–68. ISBN: 978-1-60558-025-8. DOI: 10.1145/1370731.1370747.

Mao, K., Capra, L., Harman, M. & Jia, Y. (2017). „A Survey of the Use of Crowdsourcing in Software Engineering“. In: *Journal of Systems and Software* 126.Supplement C, S. 57 –84. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2016.09.015.

Marcus, A. (2013). „The History of the Future: Sci-fi Movies and HCI“. In: *interactions* 20.4, S. 64–67. ISSN: 1072-5520. DOI: 10.1145/2486227.2486240.

material.io (oJ). *Gestures*. https://material.io/guidelines/patterns/gestures.html [Zugriff 2017-10-27].

Matsumura, K. (2015). „Studying User-Defined Gestures Toward Off the Screen Interactions“. In: *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. ITS '15. New York, NY, USA: ACM Press, S. 295–300. ISBN: 9781450338998. DOI: 10.1145/2817721.2823496.

McNeill, D. (1992). *Hand and Mind: What Gestures Reveal about Thought*. University Of Chicago Press. ISBN: 0226561321.

Microsoft Developer (oJ). *Gestures*. https://developer.microsoft.com/en-us/windows/mixed-reality/gestures [Zugriff 2017-12-13].

Microsoft Developer Network (2016). *Skeletal Tracking*. https://msdn.microsoft.com/en-us/library/hh973074.aspx [Zugriff 2016-07-30].

Mijailovic, Z. & Milicev, D. (2013). „A Retrospective on User Interface Development Technology“. In: *IEEE Software* 30.6, S. 76–83. ISSN: 0740-7459. DOI: 10.1109/MS.2013.45.

Mitra, S. & Acharya, T. (2007). „Gesture Recognition: A Survey“. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.3, S. 311–324. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2007.893280.

Morris, M. R., Ryall, K., Shen, C., Forlines, C. & Vernier, F. (2004). „Beyond Social Protocols: Multi-User Coordination Policies for Co-Located Groupware“. In: *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*. CSCW '04. New York, NY, USA: ACM Press, S. 262. ISBN: 1581138105. DOI: 10.1145/1031607.1031648.

Morris, M. R., Huang, A., Paepcke, A. & Winograd, T. (2006). „Cooperative Gestures: Multi-User Gestural Interactions for Co-Located Groupware“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. New York, NY, USA: ACM Press, S. 1201. ISBN: 1595933727. DOI: 10.1145/1124772.1124952.

Morris, M. R., Wobbrock, J. O. & Wilson, A. D. (2010). „Understanding Users' Preferences for Surface Gestures“. In: *Proceedings of Graphics Interface 2010*. GI '10. Toronto, Ont., Canada: Canadian Information Processing Society, S. 261–268. ISBN: 978-1-56881-712-5.

Moscovich, T. (2009). „Contact Area Interaction with Sliding Widgets“. In: *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*. UIST '09. New York, NY, USA: ACM, S. 13–22. ISBN: 978-1-60558-745-5. DOI: 10.1145/1622176.1622181.

Myers, B. A. (1998). „A Brief History of Human-Computer Interaction Technology“. In: *interactions* 5.2, S. 44–54. ISSN: 10725520. DOI: 10.1145/274430.274436.

Nagata, S. (1987). „How to Reinforce Perception of Depth in Single Two-dimensional Pictures“. In: *Spatial Displays and Spatial Instruments*. Hrsg. von S. R. Ellis, M. K. Kaiser & A. Grunwald. NASA Conference Publication, S. 287 –304.

Nancel, M., Pietriga, E., Chapuis, O. & Beaudouin-Lafon, M. (2015). „Mid-Air Pointing on Ultra-Walls“. In: *ACM Transactions on Computer-Human Interaction* 22.5, S. 1–62. ISSN: 10730516. DOI: 10.1145/2766448.

Navarre, D., Palanque, P., Ladry, J.-F. & Barboni, E. (2009). „ICOs: A Model-Based User Interface Description Technique Dedicated to Interactive Systems Addressing Usability, Reliability and Scalability“. In: *ACM Transactions on Computer-Human Interaction* 16.4, S. 1–56. ISSN: 10730516. DOI: 10.1145/1614390.1614393.

Nielsen, J. (1994). *Usability Engineering (Interactive Technologies)*. Morgan Kaufmann.

Nielsen, M., Störring, M., Moeslund, T. B. & Granum, E. (2003). *A Procedure for Developing Intuitive and Ergonomic Gesture Intefaces for Man-Machine Interaction*. Techn. Ber. Aalborg University.

Nielsen, M., Störring, M., Moeslund, T. B. & Granum, E. (2004). „A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for HCI“. English. In: *Gesture-Based Communication in Human-Computer Interaction*. Hrsg. von A. Camurri & G. Volpe. Bd. 2915. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 409–420. ISBN: 978-3-540-21072-6. DOI: 10.1007/978-3-540-24598-8_38.

Oakley, I., Lee, D., Islam, M. D. R. & Esteves, A. (2015). „Beats: Tapping Gestures for Smart Watches“. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. New York, NY, USA: ACM, S. 1237–1246. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702226.

Oh, U. & Findlater, L. (2013). „The Challenges and Potential of End-User Gesture Customization“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. New York, NY, USA: ACM Press, S. 1129. ISBN: 9781450318990. DOI: 10.1145/2470654.2466145.

Opera Browser (oJ). *Mouse gestures in Opera*. https://www.opera.com/help/tutorials/gestures/ [Zugriff 2018-01-31].

Oulasvirta, A., Weinkauf, T., Bachynskyi, M. & Palmas, G. (2014). „Gestikulieren mit Stil“. In: *Informatik-Spektrum* 37.5, S. 449–453. ISSN: 1432-122X. DOI: 10.1007/s00287-014-0816-2.

Oviatt, S. (1999). „Ten Myths of Multimodal Interaction“. In: *Commun. ACM* 42.11, S. 74–81. ISSN: 0001-0782. DOI: 10.1145/319382.319398.

Oviatt, S., DeAngeli, A. & Kuhn, K. (1997). „Integration and Synchronization of Input Modes During Multimodal Human-computer Interaction“. In: *Referring Phenomena in a Multimedia Context and Their Computational Treatment*. ReferringPhenomena '97. Madrid, Spain: Association for Computational Linguistics, S. 1–13.

Parra González, O., España, S. & Pastor, O. (2015). „Including Multi-Stroke Gesture-based Interaction in User Interfaces Using a Model-Driven Method“. In: *Proceedings of the XVI International Conference on Human Computer Interaction*. Interacción '15. New York, NY, USA: ACM, 6:1–6:8. ISBN: 978-1-4503-3463-1. DOI: 10.1145/2829875.2829931.

Peres, A. L. & Meira, S. L. (2015). „Towards a Framework that Promotes Integration between the UX Design and SCRUM, Aligned to CMMI“. In: *10th Iberian Conference on Information Systems and Technologies*. CISTI '15, S. 1–4. DOI: 10.1109/CISTI.2015.7170443.

Piper, A. M., O'Brien, E., Morris, M. R. & Winograd, T. (2006). „SIDES: A Cooperative Tabletop Computer Game for Social Skills Development“. In: *Proceedings of the 2006 20th Anniversary Cconference on Computer Supported Cooperative Work*. CSCW '06. New York, NY, USA: ACM Press, S. 1. ISBN: 1595932496. DOI: 10.1145/1180875.1180877.

Pöhland, R., Korzetz, M. & Schlegel, T. (2013). „Semantische Modellierung und Klassifikation von Touch-Interaktionen“. In: *Multi-Touch – Interaktion durch Berührung*. Hrsg. von T. Schlegel. Xpert.press. Springer Berlin Heidelberg, S. 69–87. ISBN: 978-3-642-36112-8. DOI: 10.1007/978-3-642-36113-5{_}4.

Pugh, S. (1991). *Total Design: Integrated Methods for Successful Product Engineering*. Addison-Wesley Pub (Sd). ISBN: 0201416395.

Ramakers, R., Vanacken, D., Luyten, K., Coninx, K. & Schöning, J. (2012). „Carpus: A Non-Intrusive User Identification Technique for Interactive Surfaces“. In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. New York, NY, USA: ACM Press, S. 35. ISBN: 9781450315807. DOI: 10.1145/2380116.2380123.

Ramos, G., Boulos, M. & Balakrishnan, R. (2004). „Pressure Widgets“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '04. Vienna, Austria: ACM, S. 487–494. ISBN: 1-58113-702-8. DOI: 10.1145/985692.985754.

Rekik, Y., Vatavu, R.-D. & Grisoni, L. (2014). „Understanding Users' Perceived Difficulty of Multi-Touch Gesture Articulation“. In: *Proceedings of the 16th International Conference on Multimodal Interaction*. ICMI '14. New York, NY, USA: ACM Press, S. 232–239. ISBN: 9781450328852. DOI: 10.1145/2663204.2663273.

Renaux, T., Hoste, L., Scholliers, C. & De Meuter, W. (2014). „Software Engineering Principles in the Midas Gesture Specification Language“. In: *Proceedings of the 2nd Workshop on Programming for Mobile & Touch*. PROMOTO '14. Portland, Oregon, USA: ACM, S. 9–16. ISBN: 978-1-4503-2295-9. DOI: 10.1145/2688471.2688478.

Rendl, C., Greindl, P., Probst, K., Behrens, M. & Haller, M. (2014). „Presstures: Exploring Pressure-Sensitive Multi-Touch Gestures on Trackpads“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, S. 431–434. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557146.

Romuald, D. (2013). „A Domain-Specific Modeling Approach for Gestural Interaction“. In: *2013 IEEE Symposium on Visual Languages and Human-Centric Computing*. VL/HCC '13, S. 181–182. DOI: 10.1109/VLHCC.2013.6645275.

Romuald, D. & Mens, T. (2015). „GISMO: A Domain-Specific Modelling Language for Executable Prototyping of Gestural Interaction“. In: *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '15. New York, NY, USA: ACM Press, S. 34–43. ISBN: 9781450336468. DOI: 10.1145/2774225.2774838.

Romuald, D., Mens, T. & Palanque, P. (2013). „A Generic Framework for Executable Gestural Interaction Models“. In: *2013 IEEE Symposium on Visual Languages and Human-Centric Computing*. VL/HCC '13, S. 35–38. DOI: 10.1109/VLHCC.2013.6645240.

Rubine, D. (1991). „Specifying Gestures by Example“. In: *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '91. New York, NY, USA: ACM, S. 329–337. ISBN: 0-89791-436-8. DOI: 10.1145/127719.122753.

Saffer, D. (2008). *Designing Gestural Interfaces: Touchscreens and Interactive Devices*. 1. Aufl. Sebastopol: O'Reilly Media, Inc. ISBN: 9780596518394.

Samsung Tech Life (oJ). *Tips & Tricks: Galaxy S 4*. http://techlife.samsung.com/tips-tricks-galaxy-s-4-1355.html [Zugriff 2016-06-26].

Schmidt, A., Kranz, M. & Holleis, P. (2005). „Interacting with the Ubiquitous Computer: Towards Embedding Interaction“. In: *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies*. sOc-EUSAI '05. New York, NY, USA: ACM, S. 147–152. ISBN: 1-59593-304-2. DOI: 10.1145/1107548.1107588.

Scholliers, C., Hoste, L., Signer, B. & De Meuter, W. (2011). „Midas: A Declarative Multi-Touch Interaction Framework“. In: *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*. TEI '11. New York, NY, USA: ACM, S. 49–56. ISBN: 978-1-4503-0478-8. DOI: 10.1145/1935701.1935712.

Scott, S. D., Carpendale, M. S. T. & Inkpen, K. M. (2004). „Territoriality in Collaborative Tabletop Workspaces“. In: *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*. CSCW '04. New York, NY, USA: ACM, S. 294–303. ISBN: 1-58113-810-5. DOI: 10.1145/1031607.1031655.

Serrano, M., Lecolinet, E. & Guiard, Y. (2013). „Bezel-Tap Gestures: Quick Activation of Commands from Sleep Mode on Tablets“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. New York, NY, USA: ACM Press, S. 3027–3036. ISBN: 9781450318990. DOI: 10.1145/2470654.2481421.

Shimon, S. S. A., Morrison-Smith, S., John, N., Fahimi, G. & Ruiz, J. (2015). „Exploring User-Defined Back-Of-Device Gestures for Mobile Devices“. In: *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI '15. New York, NY, USA: ACM Press, S. 227–232. ISBN: 9781450336529. DOI: 10.1145/2785830.2785890.

Shipman, F. M. & Marshall, C. C. (1999). „Formality Considered Harmful: Experiences, Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems“. English. In: *Computer Supported Cooperative Work* 8.4, S. 333–352. ISSN: 0925-9724.

Sim, K. C. (2012). „Speak-As-You-Swipe (SAYS): A Multimodal Interface Combining Speech and Gesture Keyboard Synchronously for Continuous Mobile Text Entry“. In: *Proceedings of the 14th ACM International Conference on Multimodal Interaction*. ICMI '12. New York, NY, USA: ACM, S. 555–560. ISBN: 978-1-4503-1467-1. DOI: 10.1145/2388676.2388793.

Sim, S. E. & Holt, R. C. (1998). „The Ramp-Up Problem in Software Projects: A Case Study of How Software Immigrants Naturalize“. In: *Proceedings of the 20th International Conference on Software Engineering*. ICSE '98. IEEE, S. 361–370. DOI: 10.1109/ICSE.1998.671389.

Smith, J. D., Graham, T. N., Holman, D. & Borchers, J. (2007). „Low-Cost Malleable Surfaces with Multi-Touch Pressure Sensitivity“. English. In: *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems*. TABLETOP'07. IEEE, S. 205–208. ISBN: 0-7695-3013-3. DOI: 10.1109/TABLETOP.2007.28.

Spano, L. D., Cisternino, A. & Paternò, F. (2012). „A Compositional Model for Gesture Definition“. In: *Proceedings of the 4th International Conference on Human-Centered Software Engineering*. HCSE '12. Toulouse, France: Springer-Verlag, S. 34–52. ISBN: 978-3-642-34346-9. DOI: 10.1007/978-3-642-34347-6_3.

Spano, L. D., Cisternino, A., Paternò, F. & Fenu, G. (2013). „GestIT: A Declarative and Compositional Framework for Multiplatform Gesture Definition“. In: *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '13. New York, NY, USA: ACM, S. 187–196. ISBN: 9781450321389. DOI: 10.1145/2494603.2480307.

Steinheider, B., Bayerl, P. S., Menold, N. & Bromme, R. (2009). „Entwicklung und Validierung einer Skala zur Erfassung von Wissensintegrationsproblemen in interdisziplinären Projektteams (WIP)“. In: *Zeitschrift für Arbeits- und Organisationspsychologie A&O* 53.3, S. 121–130. DOI: 10.1026/0932-4089.53.3.121.

Taranta II, E. M. & LaViola Jr., J. J. (2015). „Penny Pincher: A Blazing Fast, Highly Accurate $-family Recognizer“. In: *Proceedings of the 41st Graphics Interface Conference*. GI '15. Toronto, Ont., Canada: Canadian Information Processing Society, S. 195–202. ISBN: 978-0-9947868-0-7.

Taranta II, E. M., Samiei, A., Maghoumi, M., Khaloo, P., Pittman, C. R. & LaViola Jr., J. J. (2017). „Jackknife: A Reliable Recognizer with Few Samples and Many Modalities“. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: ACM, S. 5850–5861. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3026002.

Tassey, G. (2002). *The Economic Impacts of Inadequate Infrastructure for Software Testing*. Techn. Ber. National Institute of Standards and Technology.

Trump, S. & Bullock, J. (2014). „Orphion: A Gestural Multi-Touch Instrument for the iPad“. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. Hrsg. von B. Caramiaux, K. Tahiroglu, R. Fiebrink & A. Tanaka. NIME '14. London, United Kingdom: Goldsmiths, University of London, S. 159–162.

Tu, H., Ren, X. & Zhai, S. (2012). „A Comparative Evaluation of Finger and Pen Stroke Gestures“. In: *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems*. CHI ’12. New York, NY, USA: ACM, S. 1287. ISBN: 9781450310154. DOI: 10.1145/2207676.2208584.

– (2015). „Differences and Similarities between Finger and Pen Stroke Gestures on Stationary and Mobile devices“. In: *ACM Transactions on Computer-Human Interaction* 22.5, S. 1–39. ISSN: 10730516. DOI: 10.1145/2797138.

Valdes, C., Eastman, D., Grote, C., Thatte, S., Shaer, O., Mazalek, A., Ullmer, B. & Konkel, M. K. (2014). „Exploring the Design Space of Gestural Interaction with Active Tokens Through User-defined Gestures“. In: *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, S. 4107–4116. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557373.

van Dam, A. (1997). „Post-WIMP User Interfaces“. In: *Communications of the ACM* 40.2, S. 63–67. ISSN: 00010782. DOI: 10.1145/253671.253708.

Vatavu, R.-D. (2013). „There’s a World Outside Your TV: Exploring Interactions Beyond the Physical TV Screen“. In: *Proceedings of the 11th European Conference on Interactive TV and Video*. EuroITV ’13. New York, NY, USA: ACM, S. 143. ISBN: 9781450319515. DOI: 10.1145/2465958.2465972.

Vatavu, R.-D., Anthony, L. & Wobbrock, J. O. (2012). „Gestures as Point Clouds“. In: *Proceedings 14th ACM International Conference on Multimodal Interaction*. ICMI ’12. New York, NY, USA: ACM, S. 273. ISBN: 9781450314671. DOI: 10.1145/2388676.2388732.

Vatavu, R.-D., Anthony, L. & Wobbrock, J. O. (2014). „Gesture Heatmaps: Understanding Gesture Performance with Colorful Visualizations“. In: *Proceedings of the 16th International Conference on Multimodal Interaction*. ICMI ’14. New York, NY, USA: ACM, S. 172–179. ISBN: 978-1-4503-2885-2. DOI: 10.1145/2663204.2663256.

Velloso, E., Schmidt, D., Alexander, J., Gellersen, H. & Bulling, A. (2015). „The Feet in Human-Computer Interaction: A Survey of Foot-Based Interaction“. In: *ACM Computing Surveys* 48.2, 21:1–21:35. ISSN: 0360-0300. DOI: 10.1145/2816455.

Viana, D. L. & Medeiros Santos, A. L. de (2017). „A Domain-Specific Language for the Specification of Gesture-based Applications“. In: *Proceedings of the 21st Brazilian Symposium on Programming Languages*. SBLP 2017. Fortaleza, CE, Brazil: ACM, 11:1–11:7. ISBN: 978-1-4503-5389-2. DOI: 10.1145/3125374.3125376.

Walker, G. (2012). „A Review of Technologies for Sensing Contact Location on the Surface of a Display“. In: *Journal of the Society for Information Display* 20.8, S. 413–440. ISSN: 1938-3657. DOI: 10.1002/jsid.100.

Wang, F. & Ren, X. (2009). „Empirical Evaluation for Finger Input Properties in Multi-Touch Interaction“. In: *Proceedings of the 27th International Conference on Human Factors in Computing Systems*. CHI ’09. New York, NY, USA: ACM, S. 1063. ISBN: 9781605582467. DOI: 10.1145/1518701.1518864.

Watzman, S. & Re, M. (2012). „Visual Design Principles for Usable Interfaces: Everything Is Designed: Why We should Think Before Doing“. In: *Human-Computer Interaction Handbook*. Hrsg. von J. A. Jacko. 3. Aufl. Taylor & Francis. Kap. 14, S. 315–340.

Wigdor, D. & Wixon, D. (2011). *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Morgan Kaufmann/Elsevier, S. 253. ISBN: 0123822319. DOI: 10.1016/B978-0-12-382231-4.X0001-9. arXiv: arXiv:1011.1669v3.

Wigdor, D., Benko, H., Pella, J., Lombardo, J. & Williams, S. (2011). „Rock & Rails: Extending Multi-Touch Interactions with Shape Gestures to Enable Precise Spatial Manipulations“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: ACM, S. 1581–1590. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979173.

Wobbrock, J. O., Wilson, A. D. & Li, Y. (2007). „Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes“. In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST '07. Newport, Rhode Island, USA: ACM, S. 159–168. ISBN: 978-1-59593-679-0. DOI: 10.1145/1294211.1294238.

Wobbrock, J. O., Morris, M. R. & Wilson, A. D. (2009). „User-Defined Gestures for Surface Computing“. In: *Proceedings of the 27th International Conference on Human Factors in Computing Systems*. CHI '09. New York, NY, USA: ACM, S. 1083. ISBN: 9781605582467. DOI: 10.1145/1518701.1518866.

Wong, Y. Y. (1992). „Rough and Ready Prototypes: Lessons from Graphic Design“. In: *Posters and Short Talks of the 1992 SIGCHI Conference on Human Factors in Computing Systems*. CHI '92. New York, NY, USA: ACM, S. 83–84. DOI: 10.1145/1125021.1125094.

Wüest, D., Seyff, N. & Glinz, M. (2013). „FlexiSketch: A Mobile Sketching Tool for Software Modeling“. In: *Mobile Computing, Applications, and Services*. Bd. 110. Springer Berlin Heidelberg, S. 225–244. ISBN: 978-3-642-36631-4.

– (2015). „FlexiSketch Team: Collaborative Sketching and Notation Creation on the Fly“. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Bd. 2. ICSE '15, S. 685–688. DOI: 10.1109/ICSE.2015.223.

Wynne, M. & Hellesoy, A. (2012). *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*. Pragmatic Bookshelf. ISBN: 1934356808, 9781934356807.

Xamarin Developer (2016). *Gestures – Simulating Gestures in Calabash Tests*. https://developer.xamarin.com/guides/testcloud/calabash/working-with/gestures/ [Zugriff 2018-01-02].

Yota Devices (2016). *Yotaphone 2 Technical Specifications*. https://yotaphone.com/de-en/product/yotaphone2/#techspec [Zugriff 2016-08-06].

Zeleznik, R., Bragdon, A., Adeputra, F. & Ko, H.-S. (2010). „Hands-On Math: A Page-Based Multi-Touch and Pen Desktop for Technical Work and Problem Solving“. In: *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*. UIST '10. New York, NY, USA: ACM, S. 17. ISBN: 9781450302715. DOI: 10.1145/1866029.1866035.

Zhai, S. & Kristensson, P. O. (2012). „The Word-Gesture Keyboard: Reimagining Keyboard Interaction“. In: *Communications of the ACM* 55.9, S. 91–101. ISSN: 0001-0782. DOI: 10.1145/2330667.2330689.

Zhai, S., Kristensson, P. O., Appert, C., Andersen, T. H. & Cao, X. (2012). „Foundational Issues in Touch-Surface Stroke Gesture Design – An Integrative Review“. en. In: *Foundations and Trends in Human-Computer Interaction* 5.2, S. 97–205. ISSN: 1551-3955. DOI: 10.1561/1100000012.

Zhang, H., Yang, X.-D., Ens, B., Liang, H.-N., Boulanger, P. & Irani, P. (2012). „See Me, See You: A Lightweight Method for Discriminating User Touches on Tabletop Displays“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. New York, NY, USA: ACM, S. 2327–2336. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208392.

Zhang, Y. & Harrison, C. (2015). „Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition“. In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST '15. New York, NY, USA: ACM, S. 167–173. ISBN: 9781450337793. DOI: 10.1145/2807442.2807480.

Zhang, Y., Zhou, J., Laput, G. & Harrison, C. (2016). „SkinTrack: Using the Body As an Electrical Waveguide for Continuous Finger Tracking on the Skin“. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. New York, NY, USA: ACM, S. 1491–1503. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858082.

Abbildungen

2.1 Exemplarisches Eintreffen (gestrichelte Linie) und logische Zuordnung (durchgezogene Linie) der Berührungspunkte zweier Finger auf einem berührungsempfindlichen Bildschirm 15

2.2 Gesten im SE-Zyklus 16

2.3 Verschiedene Konzepte, die beim UI-Design erarbeitet, verworfen und zum finalen Prototyp weiterentwickelt werden; in Rot der finale Prototyp sowie der Weg dorthin (Quelle: Eigene Darstellung nach Greenberg et al. (2011)) 19

2.4 Der Design Funnel nach Buxton (2007) (Quelle: Eigene Darstellung nach Greenberg et al. (2011)) 19

2.5 Typische Entwicklersicht auf eine Anwendung 20

2.6 Typische Benutzersicht auf eine Anwendung 20

2.7 Zweck von Spezifikationssprachen für interaktive Anwendungen (Bildquelle: Eigene Darstellung nach Dumas et al. (2010)) 23

3.1 Abgrenzung von Sketches und Prototypen (Bildquelle: Eigene Darstellung nach Buxton (2007)). Im Vergleich zur Quelle wurde der letzte Punkt bei Prototypen geändert: Buxton (2007) verwendet hier den englischen Begriff *depiction*, der soviel wie *Bildliche Darstellung* bedeutet, aber im Deutschen nicht so kontrastreich zum Begriff *unverbindlich* steht. 33

3.2 Entwicklung von Sketches zu Prototypen im Laufe des Design Funnels (Bildquelle: Eigene Darstellung nach Greenberg et al. (2011)) 34

3.3 Ein Scribble Sketch für eine mobile Anwendung 35

3.4 Eine Verfeinerungsstufe des *Scribble Sketch* in Abb. 3.3 36

3.5 Der Sketch aus Abb. 3.4 mit einigen Annotationen 36

3.6 Ein Sketch mit Büromaterial 38

3.7 Interaktion in einem Sketch mit Büromaterial 38

3.8 Prototyp für eine Anwendung auf einer Smartwatch 39

3.9 Ein Wireframe für eine mobile Anwendung (erstellt mit *Balsamiq*) 40

3.10 Ein alternativer Wireframe für eine mobile Anwendung (erstellt mit *Balsamiq*) 40

3.11 Ein Storyboard für eine Anwendung auf einer Smartwatch 42

3.12 Ein Storyboard mit Annotationen für eine Anwendung auf einer Smartwatch 43

3.13 Ein verzweigendes Storyboard 43

4.1 Grundlegende Architektur gestenbasierter Anwendungen (Bildquelle: Eigene Darstellung nach Echtler & Klinker (2008)) 58

4.2 Phasen der Gestenerkennung 62

5.1 Einsatz einer Geste in Form eines V für verschiedene Buchstaben nach Goldberg & Richardson (1993) (Bildquelle: Eigene Darstellung nach ibd.) . . . 74

5.2 Zwei Beispiele zur Begrenzung des Ortes. Zum einen auf die obere Hälfte des Bildschirms: Die Geste wird nur erkannt, wenn der Benutzer die angegebene Grenze während der Ausführung nicht überschreitet. Zum anderen auf ein bestimmtes Objekt: Die Geste muss innerhalb der durch das Objekt bestimmten Grenzen ausgeführt werden. . . . 81

5.3 Verschiedene Möglichkeiten zur Aufteilung des Bildschirms in Planquadrate am Beispiel eines Smartphones . . . 81

5.4 Beispiel einer Geste verteilt auf mehrere Planquadrate . . . 82

5.5 Planquadrate für räumliche Gesten nach McNeill (1992) (Bildquelle: Choi et al. (2014)) . . . 82

5.6 Gesten, die ein Gebiet außerhalb des Bildschirms einbeziehen: a) ein virtuelles Objekt nach links in eine Ablage schieben; b) die Ablage wieder sichtbar machen . . . 83

5.7 Rotation für Oberflächengesten . . . 87

5.8 Rotationsachsen für räumliche Gesten . . . 87

5.9 Einzügige Geste in Form eines V . . . 90

5.10 Mehrzügige Geste in Form eines V . . . 90

5.11 Multi-Touch-Geste in Form eines V . . . 90

5.12 Zeitliche Variationen bei Multi-Touch-Gesten . . . 90

6.1 Ein Swipe mit zwei Fingern in GISpL (Quelle: Echtler & Butz (2012)) . . . 96

6.2 Regeln zur Beschreibung einer klatschenden Bewegung in GDL nach Hachaj & Ogiela (2014) (Quelle: ibd.) . . . 97

6.3 Regeln zur Beschreibung einer Geste in Form eines Lassos in GDL nach Khandkar & Maurer (2010b) (Quelle: ibd.) . . . 98

6.4 Eine Pinch-Geste in GestIT (Quelle: Spano et al. (2013)) . . . 100

6.5 Eine Geste, bei der die geöffnete Hand geschlossen wird, in Hand Gesture Notation (Quelle: Choi et al. (2014)) . . . 101

6.6 Beschreibung einer Geste in Form eines Winkens in LEG (Quelle: Viana & Medeiros Santos (2017)) . . . 102

6.7 Ein Swipe nach links formuliert in Midas (Quelle: Scholliers et al. (2011)) . . . 103

6.8 Beispiel einer Ballettfigur mit Prepose. Die Darstellung ist vereinfacht: In der Execution wird auf bereits vorher definierten Posen zurückgegriffen, die nicht dargestellt sind (Quelle: Figueiredo et al. (2016)). . . . 104

6.9 Ein Swipe nach rechts mit zwei Fingern in Proton . . . 105

6.10 Eine einfache Geste in EventHurdle (Bildquelle: Kim & Nam (2013)) . . . 107

6.11 Ein Gestenobjekt in GISMO, das das Drehen beider Hände im Uhrzeigersinn (analog zu einem Lenkrad) darstellt (Bildquelle: Romuald & Mens (2015)). . . . 108

6.12 Darstellung von Berührungen in Monox (Bildquelle: Ganhör & Spreicer (2014)) . . . 109

6.13 Darstellung von Bewegungen in Monox (Bildquelle: Ganhör & Spreicer (2014)) 110

6.14 Verschiedene Möglichkeiten, um Gesten mit Gesture Script zu definieren (Bildquelle: Lü et al. (2014)) 111

6.15 Das Wort *Geben* in Gebärdensprache (Bildquelle: Losson & Vannobel (1999)) 112

6.16 Ein Pinch mit der rechten Hand in Labanotation 113

7.1 Grundlegender Aufbau einer GestureCard mit den vier verwendeten Sektionen 118

7.2 Beispiel einer GestureCard für eine einfache Geste: eine Linie nach links, die schnell ausgeführt werden muss 119

7.3 Beispiel einer GestureCard für eine einfache Geste: eine linienförmige Bewegung nach links, bei der die Ausführungsgeschwindigkeit für die Erkennung keine Rolle spielt, aber für die Anwendung relevant ist und ausgewertet werden muss 120

7.4 Grundlegender Aufbau einer GestureCard für eine zusammengesetzte Geste 121

7.5 Beispiel einer GestureCard für eine zusammengesetzte Geste: eine Geste mit zwei Schenkeln in Form eines Häkchens 122

7.6 Beispiel einer GestureCard, die von mehreren Objekten abhängig ist 124

7.7 GestureCard, die definiert, dass eine Berührung länger als eine Sekunde dauern und auf einem Objekt ausgeführt werden muss 125

7.8 Beispiel einer GestureCard für die Wiederverwendung einer zuvor definierten Teilgeste . . . 125

7.9 Alternatives Beispiel einer GestureCard für die Wiederverwendung einer zuvor definierten Teilgeste 126

7.10 Beispiel einer GestureCard zur Spezifikation parallelen Werkzeugeinsatzes in der graphischen Sektion 127

7.11 Beispiel einer GestureCard zur Spezifikation parallelen Werkzeugeinsatzes in der Sektion für Eigenschaften 128

7.12 GestureCard für die gleiche Geste wie in Abb. 7.10, jedoch mit unterschiedlichen Eigenschaften je Teilgeste 128

7.13 GestureCard für die Darstellung wachsender Auflagefläche 129

7.14 GestureCard für eine Geste mit zwei Benutzern 131

7.15 GestureCard für eine Geste in Form eines V, die beliebig ausgerichtet sein kann 132

7.16 GestureCard für einen Swipe, der sich auf eine andere Teilgeste zubewegt 133

7.17 GestureCard für eine rhythmische Geste, bei der Teilgesten unterschiedlich lange dauern . . 134

7.18 GestureCard, die einen Swipe nach Rechts gefolgt von einem Druck auf die Oberfläche definiert 135

7.19 GestureCard für eine Geste in Form des griechischen Buchstabens Phi 136

7.20 GestureCard für eine Geste ohne festgelegte Form, deren Start- und Endpunkt aber übereinstimmen müssen 136

7.21 Symbol für unbewegliche Teilgesten in GestureCards 136

7.22 GestureCard, die eine kreisförmige mehrfach ausführbare Bewegung gegen den Uhrzeigersinn beschreibt . . . 138
7.23 GestureCard für eine Geste in Form eines X, die auf einem Objekt ausgeführt werden muss . 140
7.24 GestureCard für eine Geste, auf die die Anwendung während der Ausführung reagiert 140
7.25 Beispiele für GestureCards, die Rotation verwenden: Drehung des Werkzeugs a) ohne Bewegung und b) mit Bewegung . . . 141
7.26 GestureCard für die Geste zum Öffnen des Nachrichtencenters auf gängigen mobilen Geräten 142
7.27 GestureCard für eine Multi-Touch-Geste, bei der ein Finger und ein Stift parallel verwendet werden . . . 143
7.28 Alternative GestureCard für eine V-förmige Geste, die in einem Zug durchgeführt wird . . . 143
7.29 GestureCard für eine V-förmige Geste, die in einem Zug durchgeführt wird . . . 144
7.30 GestureCard für eine V-förmige Geste, die in zwei Zügen durchgeführt wird; a) ohne zeitliche Begrenzung, b) mit zeitlicher Begrenzung . . . 144
7.31 GestureCard für eine V-förmige Geste, die in zwei parallen Zügen durchgeführt wird 145
7.32 Gegenüberstellung der Optionen zur Darstellung der zeitlichen Relation in GestureCards: a) einzügige Gesten, b) mehrzügige Gesten, c) Parallelisierung, d) Versatz von Start- und Endzeitpunkten paralleler Teilgesten . . . 146
7.33 GestureCard zur Definition einer Geste, bei der eine Teilgeste rechts von einer anderen platziert wird . . . 147
7.34 GestureCard zur Definition einer Geste, bei der die Teilgeste Point von der Teilgeste Semi eingeschlossen wird . . . 148
7.35 Distanzen zwischen Teilgesten auf einem digitalen Whiteboard . . . 149

8.1 UI der für die Studie verwendeten Anwendung mit einer visualisierten Geste (rote Linie). Die dargestellte Abbildung war das für GestureCards verwendete Beispiel; die Geste selbst wurde nicht in der Studie eingesetzt. . . . 159
8.2 Vergleich der Zuversicht der Gruppen GeForMT und GestureCards über alle Gesten 170
8.3 Vergleich der Betrachtungszeit aller Gesten beider Gruppen . . . 171

9.1 Szene aus dem vorgeführten Video für die Geste Two-Finger-Swipe. . . . 187
9.2 Interaktionsdiagramm der durchschnittlichen Bewertung der Prägnanz für alle Notationen je Geste. . . . 188
9.3 Interaktionsdiagramm der durchschnittlichen Bewertung der Genauigkeit für alle Notationen je Geste. . . . 190
9.4 Aggregierte Punktzahl der persönlichen Präferenz aller Probanden. . . . 191

10.1 Dialog im AugIR zur Definition einfacher GestureCards . . . 198
10.2 Dialog im AugIR zur Definition zusammengesetzter GestureCards . . . 199

10.3 Analoges Storyboard für das Szenario Foto-App 200
10.4 Digitales Storyboard im AugIR für das Szenario MP3-Player 201

11.1 GestureCard einer Pose mit einem Foto als graphische Repräsentation 213
11.2 GestureCard einer Pose mit einer Silhouette als graphische Repräsentation 213
11.3 Das im Folgenden verwendete Modell der Hand (dem SDK des Leap Motion Controllers (Leap Motion Developer, oJ[e]) entnommen; Bildquelle: Darstellung aus eigener Software) 213
11.4 Eine Pose in GestureCards dargestellt mit einem Handmodell 214
11.5 Eine Pose in GestureCards mit einer zusätzlichen Eigenschaft 214
11.6 Eine Pose mit beiden Händen 215
11.7 Eine Pose mit beiden Händen als getrennte GestureCards 216
11.8 Darstellung von Schlüsselszenen in einem Video, das die Ausführung einer Pose zeigt 216
11.9 GestureCard für die Geste `Scissor` 217
11.10 GestureCard für die Geste `Scissor` mit integrierter Animation der graphischen Repräsentation 218
11.11 GestureCard für die Geste `DragNDrop` 219
11.12 Schlüsselszenen des Videos für die Teilgesten a) `Grab` und b) `Drop` 219
11.13 GestureCard für die Geste `Semicircle` 220
11.14 Schlüsselszenen des Videos für die Geste in Abb. 11.13 220

12.1 Schlüsselszenen der Geste Boxing für die Gruppen Hands und Arrows 224
12.2 Schlüsselszenen der Geste Conductor für die Gruppen Hands und Arrows 224
12.3 Schlüsselszenen der Geste Gamma für die Gruppen Hands und Arrows 224
12.4 Schlüsselszenen der Geste U für die Gruppen Hands und Arrows 225
12.5 Schlüsselszenen der Geste Phi für die Gruppen Hands und Arrows 225
12.6 GestureCards der Geste Present für die Gruppen Hands und Arrows sowie zugehörige Schlüsselszenen 225
12.7 GestureCards der Geste X für die Gruppen Hands und Arrows sowie zugehörige Schlüsselszenen 226
12.8 GestureCards der Geste Drag'n'Drop für die Gruppen Hands und Arrows sowie zugehörige Schlüsselszenen 226
12.9 Schlüsselszenen der Geste Spread für die Gruppen Hands und Arrows 227
12.10 UI der für diese Studie verwendeten Anwendung. Die Darstellung wurde den Probanden während der Einführung als Beispiel gezeigt. 227
12.11 Vergleich der Zuversicht der Gruppen Arrows und Hands über alle Gesten 233
12.12 Vergleich der Dauer aller Gesten beider Gruppen 233

13.1 Datenfluss eines Gestenerkenners für GestureCards 240

13.2 Überblick über die notwendigen Komponenten eines Gestenerkenners für GestureCards . . . 240

13.3 Die in Abb. 13.4 verwendete Geste. . . . 246

13.4 Ein potentielles BDD-Feature mit integrierter GestureCard . . . 246

13.5 Eine Geste mit hinzugefügter Varianz . . . 246

13.6 Arbeitsschritte eines potentiellen Gestensimulators . . . 247

13.7 Grundlegende Funktionsweise eines Generators für GestureCards . . . 249

A.1 Symbol für Zeigen/Berühren in GestureCards . . . 262

A.2 Optionen zur Darstellung der zeitlichen Relation: a) einzügige Gesten, b) mehrzügige Gesten, c) Parallelisierung, d) zeitlicher Versatz von Start- und Endpunkten paralleler Teilgesten . . . 265

B.1 Einführung in GestureCards, Seite 1 . . . 267

B.2 Einführung in GestureCards, Seite 2 . . . 267

B.3 Einführung in GestureCards, Seite 3 . . . 268

B.4 Einführung in GestureCards, Seite 4 . . . 268

B.5 Einführung in GestureCards, Seite 5 . . . 268

B.6 Einführung in GeForMT, Seite 1 (basierend auf Kammer et al. (2015)) . . . 269

B.7 Einführung in GeForMT, Seite 2 (basierend auf Kammer et al. (2015)) . . . 269

B.8 Einführung in GeForMT, Seite 3 (basierend auf Kammer et al. (2015)) . . . 269

B.9 Einführung in GeForMT, Seite 4 (basierend auf Kammer et al. (2015)) . . . 270

B.10 Einführung in GeForMT, Seite 5 (basierend auf Kammer et al. (2015)) . . . 270

B.11 Studie A: Spickzettel für GestureCards, Seite 1 . . . 271

B.12 Studie A: Spickzettel für GestureCards, Seite 2 . . . 271

B.13 Studie A: Spickzettel für GeForMT, Seite 1 (basierend auf Kammer et al. (2015)) . . . 272

B.14 Studie A: Spickzettel für GeForMT, Seite 2 (basierend auf Kammer et al. (2015)) . . . 272

C.1 Fragebogen, Seite 1 . . . 277

C.2 Fragebogen, Seite 2 . . . 278

C.3 Fragebogen, Seite 3 . . . 279

C.4 Fragebogen, Seite 4 . . . 280

C.5 Fragebogen, Seite 5 . . . 281

C.6 Fragebogen, Seite 6 . . . 282

C.7 Fragebogen, Seite 7 . . . 283

C.8 Fragebogen, Seite 8 . . . 284

D.1 Einführung in GestureCards, Seite 1 293
D.2 Einführung in GestureCards, Seite 2 293
D.3 Einführung in GestureCards, Seite 3 294
D.4 Einführung in GestureCards, Seite 4 294
D.5 Fragebogen, Seite 1 295
D.6 Fragebogen, Seite 2 296
D.7 Fragebogen, Seite 3 297
D.8 Fragebogen, Seite 4 298

E.1 Einführung in GestureCards, Seite 1 301
E.2 Einführung in GestureCards, Seite 2 301
E.3 Einführung in GestureCards, Seite 3 302

Tabellen

2.1 HCI-Probleme in den Phasen der Software-Entwicklung 18

4.1 De-Facto-Standard für Oberflächengesten 56

7.1 Mögliche Operatoren zur Definition von Eigenschaften 121

8.1 Die Geste V ausgedrückt in GeForMT und GestureCards 155

8.2 Die Geste X ausgedrückt in GeForMT und GestureCards 155

8.3 Die Geste Y ausgedrückt in GeForMT und GestureCards 156

8.4 Die Geste Pfeil ausgedrückt in GeForMT und GestureCards 156

8.5 Die Geste HoldMove ausgedrückt in GeForMT und GestureCards 157

8.6 Die Geste Spread ausgedrückt in GeForMT und GestureCards 157

8.7 Die Geste SSL ausgedrückt in GeForMT und GestureCards 158

8.8 Die Geste LSLS ausgedrückt in GeForMT und GestureCards 158

8.9 Überblick über alle abhängigen Variablen 161

8.10 Überblick über alle Hypothesen der ersten Studie 162

8.11 Ergebnisse der Studie kumuliert nach Probanden. Die Werte für Zuversicht liegen zwischen 1 (sicher) und 4 (unsicher). 166

8.12 Korrekt und nicht korrekt identifizierte Gesten je Gruppe. 167

8.13 Korrekt identifizierte Gesten je Geste und Gruppe 168

8.14 Kreuztabellen der korrekt und nicht korrekt identifizierten Gesten je Geste und Gruppe 168

8.15 Aufrufe des Spickzettels je Gruppe und Geste 169

8.16 Median der Zuversicht je Gruppe und Geste; Werte reichen von *sicher* (1) bis *unsicher* (4) . . 170

8.17 Zuversicht der Probanden je Notation für alle Gesten 170

8.18 Durchschnittliche Gesamtzeit je Geste (in Sekunden) 171

8.19 Durchschnittliche Betrachtungszeit je Geste (in Sekunden) 171

9.1 Erfahrung in der Bedienung und Entwicklung von Multi-Touch-Anwendungen aller Probanden 182

9.2 Die Geste Two-Finger-Swipe ausgedrückt in GeForMT, GestureCards und Monox 183

9.3 Die Geste LSLS ausgedrückt in GeForMT, GestureCards und Monox 183

9.4 Die Geste Drag-Out ausgedrückt in GeForMT, GestureCards und Monox 184

9.5 Die Geste Spread ausgedrückt in GeForMT, GestureCards und Monox 184

9.6 Überblick über alle Hypothesen der zweiten Studie für die Variable Genauigkeit 186

10.1 User Stories für das Szenario MP3-Player 199
10.2 User Stories für das Szenario Foto-App 200
10.3 Aggregierte Ergebnisse der Fragen 1–7, 23 und 24 204
10.4 Ergebnisse der Frage 14. 204
10.5 Aggregierte Ergebnisse der Fragen 8–13 204
10.6 Aggregierte Ergebnisse der Fragen 27, 29 und 30 205

12.1 Überblick über alle abhängigen Variablen 228
12.2 Überblick über alle Hypothesen der Studie zum Vergleich graphischer Repräsentationen für räumliche Gesten 230
12.3 Korrekt und nicht korrekt identifzierte Gesten je Gruppe. 232
12.4 Summe der korrekt identifizierten Gesten je Gruppe und Geste. 232
12.5 Durchschnittliche Dauer der Bearbeitung einer Geste je Gruppe in Sekunden. 234
12.6 Anzahl der Replays je Gruppe und Geste. 234

A.1 Werte der Eigenschaft Anzahl der Werkzeuge 259
A.2 Werte der Eigenschaft Auflagefläche 259
A.3 Werte der Eigenschaft Ausführender Benutzer 260
A.4 Werte der Eigenschaft Ausrichtung 260
A.5 Werte der Eigenschaft Bewegungsrichtung 260
A.6 Werte der Eigenschaft Dauer 261
A.7 Werte der Eigenschaft Distanz 261
A.8 Werte der Eigenschaft Druck 262
A.9 Werte der Eigenschaft Geschwindigkeit 262
A.10 Werte der Eigenschaft Multiplizität 263
A.11 Werte der Eigenschaft Ort der Ausführung 263
A.12 Werte der Eigenschaft Reaktion zur Ausführungszeit 263
A.13 Werte der Eigenschaft Rotation 264
A.14 Werte der Eigenschaft Werkzeug 264
A.15 Werte der Eigenschaft Räumlicher Bezug 265

B.1 Ergebnisse des Probanden GeForMT1 273
B.2 Ergebnisse des Probanden GeForMT2 273

B.3 Ergebnisse des Probanden GeForMT3 . . . 273
B.4 Ergebnisse des Probanden GeForMT4 . . . 274
B.5 Ergebnisse des Probanden GeForMT5 . . . 274
B.6 Ergebnisse des Probanden GeForMT6 . . . 274
B.7 Ergebnisse des Probanden GC1 . . . 275
B.8 Ergebnisse des Probanden GC2 . . . 275
B.9 Ergebnisse des Probanden GC3 . . . 275
B.10 Ergebnisse des Probanden GC4 . . . 276
B.11 Ergebnisse des Probanden GC5 . . . 276
B.12 Ergebnisse des Probanden GC6 . . . 276

C.1 Demographische Daten der Probanden . . . 285
C.2 Ergebnisse der Geste Two-Finger-Swipe . . . 286
C.3 Ergebnisse der Geste LSLS . . . 287
C.4 Ergebnisse der Geste Swipe-And-Press . . . 288
C.5 Ergebnisse der Geste Drag-Out . . . 289
C.6 Ergebnisse der Geste Spread . . . 290
C.7 Gesamtpunkte . . . 291

D.1 Ergebnisse aller Probanden, Fragen 1-15 . . . 299
D.2 Ergebnisse aller Probanden, Fragen 16-30 . . . 299

E.1 Ergebnisse des Probanden asch, Gruppe Hand . . . 303
E.2 Ergebnisse des Probanden dbhc, Gruppe Hand . . . 303
E.3 Ergebnisse des Probanden dmfy, Gruppe Arrows . . . 303
E.4 Ergebnisse des Probanden ghpp, Gruppe Hand . . . 304
E.5 Ergebnisse des Probanden idyx, Gruppe Hand . . . 304
E.6 Ergebnisse des Probanden kohn, Gruppe Arrows . . . 304
E.7 Ergebnisse des Probanden nyyc, Gruppe Hand . . . 304
E.8 Ergebnisse des Probanden wblk, Gruppe Arrows . . . 305
E.9 Ergebnisse des Probanden wqyx, Gruppe Arrows . . . 305
E.10 Ergebnisse des Probanden yfdh, Gruppe Arrows . . . 305