Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart
Germany

Diplomarbeit Nr. 3288

Gestensteuerung für Powerwall-basierte Visualisierungen

Nico Ploner

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Thomas Ertl
Betreuer:	Prof. Dr. Albrecht Schmidt Dipl.-Phys. Michael Raschke Dipl.-Inf. Bastian Pfleging
begonnen am:	10.01.2012
beendet am:	09.07.2012
CR-Klassifikation:	H.5.2 H.5.3 I.3.6

Kurzfassung

Eine Stärke von großflächigen, hochauflösenden Displays ist es, große Datenmengen anzuzeigen und beispielsweise einer Gruppe zur Analyse zu präsentieren. Insbesondere für mehrere Benutzer, die gemeinsam mit Powerwall-Visualisierungen interagieren möchten, eignen sich Tastatur und Maus nur bedingt zur Interaktion mit den dargestellten Daten. Nachteile der Maus- und Tastatursteuerung können durch Freihandgesten vermieden werden. Zusätzlich ermöglichen Freihandgesten einen natürlichen Umgang mit den dargestellten Visualisierungen.

In dieser Diplomarbeit wird anhand zweier Szenarien zur Powerwall-gestützten Datenanalyse ein Interaktionskonzept zur Gestensteuerung von Powerwalls entwickelt. Um die Anwendbarkeit dieses Konzepts zu untersuchen wird ein Prototyp implementiert, der eines der beiden Szenarien realisiert. Zusätzlich wird eine Benutzerstudie konzipiert und als Pilotstudie durchgeführt, um das Interaktionskonzept zu evaluieren. Darüber hinaus sollen mit dieser Studie weitere intuitive Gesten zur Interaktion mit Powerwall-Visualisierungen identifiziert werden. Ebenso wird ein Framework konstruiert, mit dem das Interaktionskonzept zur Gestensteuerung mit anderen Interaktionskonzepten zu multimodalen Interaktionsumgebungen kombiniert werden können.

Abstract

One feature of large, high resolution displays is to present large datasets for analysis e. g. to a group of people. Especially for multiple users analyzing data, mouse and keyboard are only partially suited interacting with powerwall visualizations. To avoid disadvantages from mouse and keyboard interaction, in-air-gestures can be used to interact with the displayed data. Additionally gesture interaction provides a natural interface to powerwall visualizations.

In this diploma thesis a concept to be able to interact with large, high resolution displays is developed based on two scenarios in which a powerwall is used for data analysis. To research the applicability of such an interaction concept a software prototype is implemented. This prototype supports one of the previously named scenarios. Also a user study is designed to evaluate the concept and identify more gestures for powerwall interaction. Additionally a framework is constructed that allows embedding the concept of powerwall interaction in a multi-modal environment with other interaction devices.

Inhalt

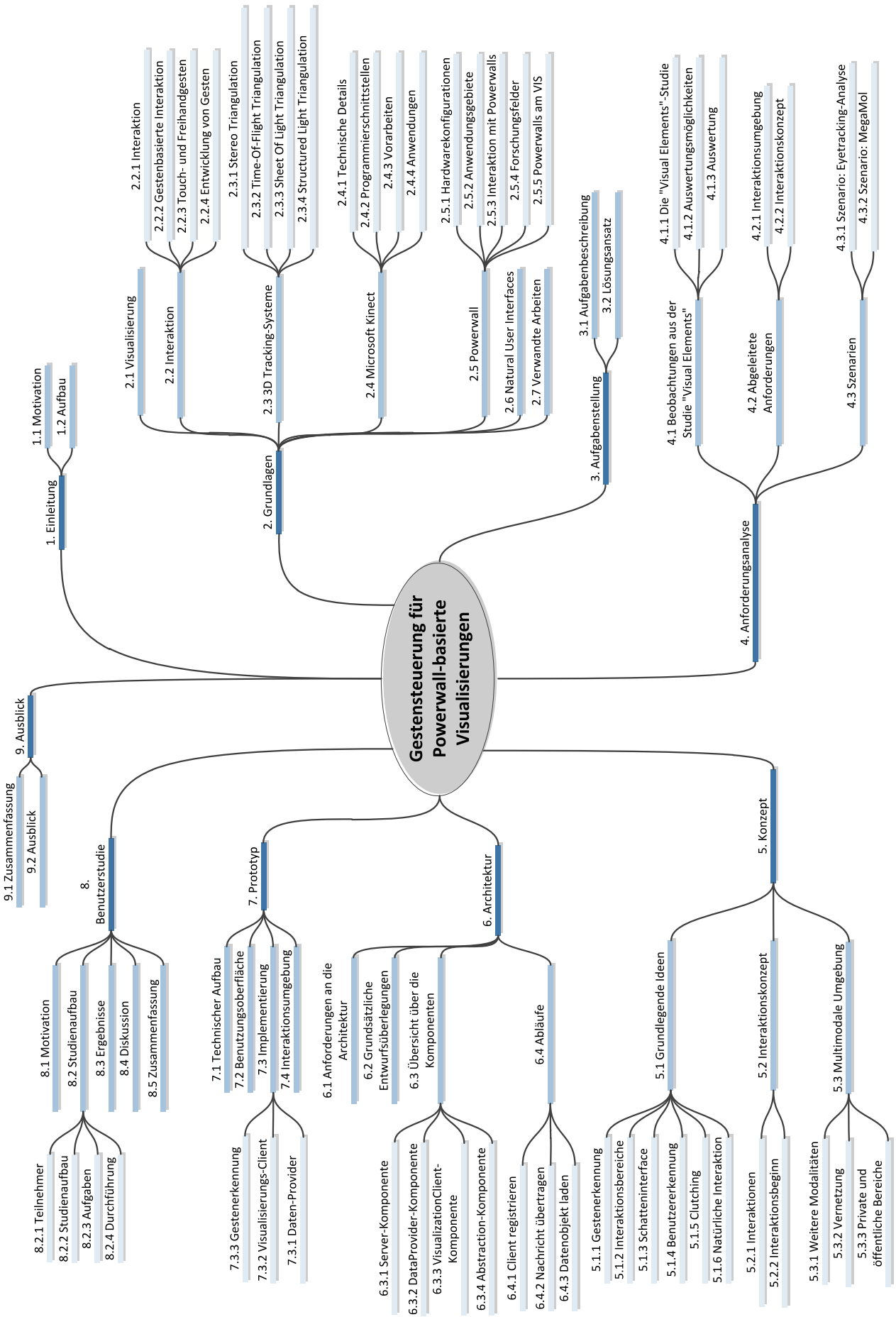
Kurzfassung 1
Abstract 1
Inhalt als Mind-Map 7
Abbildungsverzeichnis 9
Tabellenverzeichnis 15
Quellcodeverzeichnis 17
1 Einleitung 19
1.1 Motivation 19
1.2 Aufbau 20
2 Grundlagen zur Gestensteuerung Powerwall-basierter Visualisierungen 21
2.1 Visualisierung 21
2.1.1 Wissenschaftliche Visualisierungen und Informationsvisualisierung 22
2.2 Interaktion 24
2.2.1 Interaktion 24
2.2.2 Gestenbasierte Interaktion 26
2.2.3 Touch- und Freihandgesten 27
2.2.4 Entwicklung von Gesten 28
2.3 3D Tracking Systeme 30
2.3.1 Stereo Triangulation 30
2.3.2 Time-Of-Flight und Inferometrie 31
2.3.3 Sheet Of Light Triangulation 31
2.3.4 Structured Light Triangulation 32
2.4 Microsoft Kinect 33
2.4.1 Technische Details 33
2.4.2 Programmierschnittstellen 34
2.4.3 Vorarbeiten 35
2.4.4 Anwendungen des Kinect-Sensors 36
2.5 Powerwall 38
2.5.1 Hardwarekonfigurationen 38
2.5.2 Anwendungsgebiete 40
2.5.3 Interaktion mit Powerwalls 41
2.5.4 Forschungsfelder 43
2.5.5 Powerwalls am Institut für Visualisierung und Interaktive Systeme 44
2.6 Natural User Interfaces 45

2.7 Verwandte Arbeiten 47
3 Aufgabenstellung und Lösungsansatz 51
3.1 Aufgabenbeschreibung 51
3.2 Lösungsansatz 53
4 Ergebnisse der Anforderungsanalyse 55
4.1 Beobachtungen aus der Studie „Visual Elements“ 55
4.1.1 Die „Visual Elements“-Studie 55
4.1.2 Auswertungsmöglichkeiten 56
4.1.3 Auswertung 59
4.2 Abgeleitete Anforderungen 61
4.2.1 Anforderungen an eine Interaktionsumgebung 61
4.2.2 Anforderungen an ein Interaktionskonzept 63
4.3 Szenarien 65
4.3.1 Szenario: Eyetracking-Analyse 65
4.3.2 Szenario: MegaMol 70
5 Konzept 75
5.1 Grundlegende Ideen 75
5.1.1 Gestenerkennung 75
5.1.2 Interaktionsbereiche 75
5.1.3 Schatteninterface 76
5.1.4 Benutzererkennung 77
5.1.5 Clutching 78
5.1.6 Natürliche Interaktion 79
5.2 Interaktionskonzept 79
5.2.1 Interaktionen 80
5.2.2 Interaktionsbeginn 85
5.3 Einbettung in eine multimodale Umgebung 85
5.3.1 Weitere Interaktionsmodalitäten 86
5.3.2 Vernetzung 87
5.3.3 Private und öffentliche Bereiche 87
6 Architektur 89
6.1 Anforderungen an die Architektur 89
6.2 Grundsätzliche Entwurfsüberlegungen 90
6.3 Übersicht über die Komponenten 92
6.3.1 Server-Komponente 93

6.3.2 DataProvider-Komponente 95
6.3.3 VisualizationClient-Komponente 97
6.3.4 Abstraction-Komponente 103
6.4 Abläufe 107
6.4.1 Client registrieren 107
6.4.2 Nachricht übertragen 108
6.4.3 Datenobjekt laden 110
7 Prototyp 113
7.1 Technischer Aufbau 113
7.2 Benutzungsoberfläche 114
7.3 Implementierung 117
7.3.1 Daten-Provider 117
7.3.2 Visualisierungs-Client 122
7.3.3 Gestenerkennung 125
7.4 Multimodale Interaktionsumgebung 128
8 Benutzerstudie 129
8.1 Motivation 129
8.2 Methode 129
8.2.1 Teilnehmer 130
8.2.2 Studienaufbau 130
8.2.3 Aufgaben und Hypothesen 131
8.2.4 Studiendurchführung 133
8.3 Ergebnisse 135
8.3.1 Abschnitt 1: Gestenfindung 135
8.3.2 Abschnitt 2: Beurteilung der Gesten im Prototyp 138
8.4 Diskussion 141
8.4.1 Abschnitt 1: Gestenfindung 141
8.4.2 Abschnitt 2: : Beurteilung der Gesten im Prototyp 142
8.5 Zusammenfassung 145
8.5.1 Zusammenfassung der Pilotstudie 145
8.5.2 Optimierung des Studiendesigns für die Benutzerstudie 146
9 Zusammenfassung und Ausblick 149
9.1 Zusammenfassung und Bewertung 149
9.1.1 Zusammenfassung 149
9.1.2 Bewertung 151

9.2 Ausblick 152
9.2.1 Verbesserung der Handerkennung 152
9.2.2 Erweiterungen der Interaktion 154
9.2.3 Studie zur multimodalen Interaktionsumgebung 156
Literaturverzeichnis 157
Anhang A: Allgemeiner Fragebogen 164
Demografische Angaben 164
Angaben zur Ausbildung 164
Vorkenntnisse in der Gesteninteraktion 164
Anhang B: Fragebogen zur Gestenfindung 165
Interaktion: Objekt verschieben 165
Anhang C: Ausgangs- und Zieldarstellungen des ersten Studienabschnitts 166
Tutorial: Objekt rotieren 166
Aufgabe 1.1. Objekt verschieben 166
Aufgabe 1.2. Objekt löschen 166
Aufgabe 1.3. Objekt skalieren 167
Aufgabe 1.4. Objekt duplizieren 167
Aufgabe 1.5. Objekte koppeln 167
Aufgabe 1.6. Ansicht in einem Objekt wechseln 167
Aufgabe 1.7. Objektinhalt verändern 168
Anhang D: Fragebogen zur Beurteilung des Prototyps 169
Interaktion: Objekt Verschieben 169
Anhang E: Fragebogen zum Gesamtsystem 170

Inhalt als Mind-Map



Abbildungsverzeichnis

Abbildung 1: Visualisierungspipeline (Ertl, 2010). 22
Abbildung 2: Visualisierung von Strömungsdaten um ein Fahrzeug in einem Windkanal (Ertl, 2010). 23
Abbildung 3: Beispiel aus der Informationsvisualisierung. Strukturierung der Inhalte von Dateien mit dem Stichwort „idea“. Je mehr Dokumente ein mit „idea“ verwandtes Stichwort beinhalten, desto näher an der Mitte wird dieses Stichwort in einem Bogensegment dargestellt. Je dunkler die Farbe des Segments ist, desto häufiger kommt das jeweilige Stichwort in den Dokumenten vor (Collins, 2006). 24
Abbildung 4: Interaktion zwischen Benutzer und Computer. Der Benutzer gibt Informationen in den Computer ein, der durch Berechnungen ein Ergebnis erzeugt. Das Ergebnis wird vom den Benutzer interpretiert und daraus neue Eingabeinformationen für die nächste Berechnung abgeleitet. Vergleiche (Dix, Finlay, Abowd, & Beale, 2004). 24
Abbildung 5: Erweiterung Visualisierungs-Pipeline um die Interaktion des Benutzers mit Visualisierungen (vgl. Abbildung 1). Aus der Interpretation der Visualisierungen leitet der Benutzer neue Parameter für die Schritte "filtering", "mapping" und "rendering" ab und führt die Änderungen durch Interaktionen aus. 25
Abbildung 6: Gestentaxonomie nach Karam und Schraefel (Karam & Schraefel, 2005). 27
Abbildung 7: Erkennen einer Touchgeste in den drei Schritten "registration" (1), "continuation" (2) und "termination" (3) (Wigdor & Wixon, 2011). 28
Abbildung 8: Bei der Stereo Triangulation werden übereinstimmende Punkte in mehreren Bildern gesucht und deren Abstand zur Kamera mithilfe der Disparität der Punkte und dem Abstand der Kameralinsen berechnet (Kabayama & Trabasso, 2002). 30
Abbildung 9: Zwei Kameras des OptiTrack-Systems im VR-Labor des Informatikgebäudes der Universität Stuttgart (links). Marker zur Positionserkennung der 3D-Maus (rechts). 31
Abbildung 10: Bei der Time-Of-Flight-Methode wird die Zeit vom Aussenden eines Laserstrahls bis zum Eintreffen beim Sensor gemessen um die Distanz zum Objekt zu bestimmen. 31
Abbildung 11: Bei der Sheet Of Light Triangulation wird ein Lichtstreifen auf das zu untersuchende Objekt projiziert. Eine Kamera erfasst die Verformung des Lichtstreifens durch die Objektoberfläche. 32
Abbildung 12: Links wird ein Punktemuster in die Szene projiziert. Rechts wird dieses Muster durch ein Objekt verzerrt (Kinect Hacking 103: Looking at Kinect IR Patterns, 2010). 32
Abbildung 13: Der Microsoft Kinect 3D Sensor verfügt über eine Infrarotkamera und eine Farbkamera zur 3D-Bilderkennung und ein Mikrofonarray, um Geräusche lokalisieren zu können (Davison, 2011). 33
Abbildung 14: Erkennungsmodi der Kinect for Windows. Im Near Mode erkennt der Sensor Objekte, die sich bis zu 40cm nah am Sensor befinden (Microsoft, 2012). 34
Abbildung 15: Tiefenbild einer Szene, aufgenommen mit dem Kinect-Sensor (links). Je heller ein Bereich dargestellt wird, desto näher befindet sich dieser Bereich am Sensor. Rechts wurde eine Person in der Szene erkannt und markiert. 35
Abbildung 16: Durch den Kinect-Sensor erkannte Skelette von Personen. Links: Darstellung im OpenNI-Framework, rechts: mit dem Microsoft Kinect for Windows SDK. 36
Abbildung 17: Hand- und Fingererkennung mit dem Kinect-Sensor. Es werden die Kontur der Hand (gelb), der Mittelpunkt der Handfläche (grün), die Fingerspitzen (rot) und die Richtungsvektoren der Finger dargestellt (Stegmueller, 2012). 36
Abbildung 18: Kinect Sports - die Bewegungen des Spielers beim Diskuswurf werden auf den Avatar in der virtuellen Sportarena übertragen. 37

Abbildung 19: Dance Central – der Spieler imitiert die Tanzbewegungen eines virtuellen Tanzlehrers. 38
Abbildung 20: Mehrere Monitore an einem Bürocomputer erweitern den Arbeitsbereich. 39
Abbildung 21: Tiled LCD Displays. Links der MultiTouch-Tabletop "lambdaTable", rechts die Powerwall "lambdaVision" (Renambot). 39
Abbildung 22: VisBlock - Powerwall aus 20 aneinander gefügte Bildprojektionen (VisWall, 2012). 40
Abbildung 23: Powerwall im VR-Labor des Informatikgebäudes der Universität Stuttgart. 44
Abbildung 24: Schematische Darstellung der Powerwall im VISUS-Gebäude. Zehn Beamer projizieren ein Bild auf eine 5,97 x 2,25 Meter große Verbundglasscheibe (VISUS, 2012). 45
Abbildung 25: Beispiel eines Natural User Interface. Mit Körperbewegungen kann der Benutzer Farbe auf die Leinwand bringen (Microsoft, 2009). 46
Abbildung 26: Der Nintendo Wii Controller vermittelt - eingebaut in ein Lenkrad - ein natürliches Fahrgefühl. 47
Abbildung 27: Ein Plexiglaszylinder wird vom Microsoft Surface als Interaktionsobjekt erkannt. 47
Abbildung 28: Einsatz einer Powerwall im Projekt "Leeds Virtual Microscope" (links). Rechts interagiert ein Mitarbeiter mit einer Konstruktionszeichnung auf der ABB-Powerwall. 48
Abbildung 29: Links ist die Anwendung einer Powerwall in Entwickler-Meetings skizziert. Rechts fügt sich die Powerwall in eine multimodale Interaktionsumgebung ein. 49
Abbildung 30: Bildschirmausschnitt bei der Analyse der "Visual Elements" Studie. 51
Abbildung 31: Aufgabe aus dem ersten Teil der Studie "Visual Elements" - Die Punkte P1, P2 und P3 mussten von den Probanden abgelesen werden. 55
Abbildung 32: Aufgaben aus dem zweiten Teil der Studie "Visual Elements" - Aus den Abbildungen musste die größte Farbfläche bestimmt werden. 56
Abbildung 33: Dreiecke, die im dritten Teil der Studie "Visual Elements" verglichen werden mussten. 56
Abbildung 34: Darstellung der Blickpunkte aller Probanden beim Ablesen zweier Punkte aus einem Koordinatensystem in einer Heatmap. Rot dargestellte Bereiche wurden von Probanden häufig oder lange fixiert, grün dargestellte Bereiche wurden kaum oder nur flüchtig fixiert. 57
Abbildung 35: Scan-Path eines Probanden beim Vergleich zweier Dreiecke. Die Reihenfolge der Fixationen des Probanden wird durch die blaue Linie dargestellt. 57
Abbildung 36: Scan-Paths zweier Probanden über drei Areas of Interest. oben: Gaze Duration Sequence Diagram, mitte: Fixation Point Diagram, unten: Gaze Duration Distribution Diagram. 59
Abbildung 37: Import der Eyetracking-Daten in das System der multimodalen Interaktionsumgebung. 65
Abbildung 38: Die ausgewählten Aufgaben aus der Eyetracking-Studie werden im Daten-Explorer der Powerwall angezeigt. 66
Abbildung 39: Mit einer Geste wird eine Aufgabe aus dem Daten-Explorer auf die Powerwall bewegt. 66
Abbildung 40: Mit einer Geste wird eine Ansicht auf der Powerwall dupliziert. 67
Abbildung 41: Durch eine Wischgeste wird die Visualisierung in einer Ansicht gewechselt. 67
Abbildung 42: In den Ansichten werden die Areas of Interest (AOIs) dargestellt. 67
Abbildung 43: Mit Gesten werden die Achsen im Gaze Duration Diagram sortiert. 68
Abbildung 44: Mit einer Verschiebegeste wird ein Proband aus dem Diagramm in den Papierkorb geschoben und gelöscht. 68
Abbildung 45: Die Probanden werden mit Gesten in zwei Gruppen eingeteilt. 69
Abbildung 46: Mit einer Geste wird von der Ansicht ein Screenshot erstellt. 69

Abbildung 47: Auf einem Notebook werden die Daten für die Präsentation geladen. 71
Abbildung 48: Über eine Geste werden die Proteindaten aus dem Notebook in die multimodale Umgebung integriert. 71
Abbildung 49: Aus dem Daten-Explorer wird eine MegaMol-Ansicht auf der Powerwall geöffnet. 72
Abbildung 50: Mit einer Geste wird die MegaMol-Ansicht dupliziert. 72
Abbildung 51: Mit Gesten zum Verschieben und Zoomen werden die Ansichten positioniert. 72
Abbildung 52: Mit verschiedenen Gesten wird die Darstellung des Proteins beeinflusst. 73
Abbildung 53: Mit einer Geste wird die Darstellung des gesamten Proteins geändert. 73
Abbildung 54: Mit Greif- und Zoomgesten wird der Betrachtungswinkel des Proteins beeinflusst. 74
Abbildung 55: Interaktionsbereiche im Erkennungsbereich der Kinect (schwarzes Trapez). Im Fern-Interaktionsbereich (grün) können Übersichtsaufgaben durchgeführt werden, im Nah-Interaktionsbereich (blau) können Daten innerhalb der Ansichten manipuliert werden. 75
Abbildung 56: Spiegelbildliche Darstellung des Benutzers als Schatten mit Interaktionsobjekten (Festplatte und Papierkorb). 76
Abbildung 57: Schatteninterface ohne gestartete Interaktionserkennung. Bewegungen des Benutzers werden nicht als Gesteninteraktionen interpretiert (links). Rechts hat der Benutzer beide Hände über die Linie bewegt und kann Interaktionen auslösen. 78
Abbildung 58: Im Nah-Interaktionsbereich erkannte geöffnete Hand (links). Rechts wurde eine Greifgeste erkannt und der Benutzer kann eine Interaktion durchführen. 79
Abbildung 59: Beispiel der Interaktion "Ansicht verschieben" - mit der Bewegung in Pfeilrichtung wird die Ansicht von links oben nach rechts unten verschoben. 81
Abbildung 60: Interaktion "Ansicht löschen". Durch das Verschieben der Ansicht in den Papierkorb wird die Ansicht gelöscht. 81
Abbildung 61: Interaktion "Ansicht skalieren". Auswahl der zu skalierenden Ansicht mit beiden Händen (links). Rechts wird die Ansicht durch Auseinanderbewegen der Hände vergrößert. 82
Abbildung 62: Interaktion "Ansicht duplizieren". Zu duplizierende Ansicht wird mit beiden Händen aktiviert (links) und anschließend das Duplikat zur Seite herausgezogen (rechts). 82
Abbildung 63: Interaktion "Ansichten koppeln". Oben werden die zu koppelnden Ansichten mit je einer Hand ausgewählt und die Hände zusammengeführt. Anschließend sind die beiden Ansichten miteinander verbunden (unten). 83
Abbildung 64: Interaktion "Visualisierung ändern". Links wird in der Ansicht eine Ball-Visualisierung dargestellt. Mit einer Wischgeste wird die dargestellte Visualisierung gewechselt (rechts). 83
Abbildung 65: Interaktion "Auswahl treffen". In einem Radialmenü werden vier Elemente zur Auswahl dargestellt (links). Durch eine Bewegungsgeste in die entsprechende Richtung wählt der Benutzer ein Element aus (rechts). 83
Abbildung 66: Durch eine Greifgeste im Nah-Interaktionsbereich kann der Inhalt einer Ansicht manipuliert werden. 84
Abbildung 67: Piktogramm einer multimodalen Interaktionsumgebung. In dieser Umgebung ermöglichen Kamera-Sensoren eine Gesteninteraktion mit einer Powerwall. Zusätzlich werden Tabletop-Computer zur Interaktion mit den Daten verwendet. 86
Abbildung 68: Ein Plexiglaszylinder wird als Tangible Objekt zur Interaktion erkannt. 86
Abbildung 69: Entwurfsskizze für das Kommunikationsframework bestehend aus einem Server (mitte), einem Daten-Provider Client (rechts) und zwei Visualisierungs-Clients (oben und unten). Die Komponenten sind über ein Netzwerk verbunden und kommunizieren über Web Services. 91
Abbildung 70: Komponentendiagramm des Kommunikationsframeworks. Oben ist die Serverkomponente mit den Schnittstellen zum Nachrichtenaustausch zwischen den Clients

dargestellt. Rechts unten die Komponente "DataProvider", mit der Daten zur Visualisierung zur Verfügung gestellt werden können. Links unten die VisualizationClient-Komponente, die Daten zur Interaktion darstellen kann. Die Abstraction-Komponente (links oben) stellt Basisklassen zur Erweiterung des Frameworks dar. Grau dargestellte Komponenten müssen für jede Anwendung separat implementiert werden. 93
Abbildung 71: Klassenstruktur der Server-Komponente (Abbildung 70 oben rechts). Der Teil "ServerMonitor" kann für jedes Interaktionssystem unterschiedlich implementiert werden. 94
Abbildung 72: Klassenstruktur der DataProvider-Komponente (Abbildung 70 unten rechts). Der Teil "Custom Data Provider" muss für jede Datenquelle speziell implementiert werden....95
Abbildung 73: Klassenstruktur der VisualizationClient-Komponente (Abbildung 70 unten links). Der Teil "Program" visualisiert Daten und ermöglicht die Interaktion. Für jeden Gerätetyp in der Interaktionsumgebung muss ein VisualizationClient und das dazugehörige Programm implementiert werden. 98
Abbildung 74: Plugin-Struktur für Visualisierungen, die in einem VisualizationClient verwendet werden können. Der Platzhalter „Program“ steht für die Anwendung der VisualizationClient-Komponente, welche die Visualisierungs-Plugins verwenden soll. 100
Abbildung 75: Klassenstruktur zum Datentransfer im Kommunikationsframework und zur Datenfilterung. 103
Abbildung 76: Klassenstruktur der Nachrichten, die durch das Kommunikationsframework übertragen werden können. Beispielhaft ist je eine Nachrichtenklasse zum Filtern (FilterDataMessage), zum Datentransfer (LoadObjectMessage) und zur Dateninteraktion (NewViewMessage) dargestellt. 104
Abbildung 77: Sequenzdiagramm zum Ablauf beim Registrieren eines VisualizationClients beim Server. Über die Klasse ClientApplication und den WebService des VisualizationClients wird der Client beim MessageBroker des Servers registriert. 107
Abbildung 78: Sequenzdiagramm zum Ablauf beim Versenden einer Nachricht an alle Clients in der Interaktionsumgebung. Über die Klasse ClientApplication und den WebService des VisualizationClients wird die Nachricht zum MessageBroker des Servers übertragen, der die Nachricht an alle verbundenen Clients weiterleitet. 108
Abbildung 79: Aktivitätsdiagramm zur Verarbeitung einer Nachricht im MessageBroker der Server-Komponente. Es wird entschieden, ob die Nachricht auf dem Server verarbeitet oder an welche Clients die Nachricht weitergeleitet wird. 109
Abbildung 80: Sequenzdiagramm zum Laden eines Datenobjekts von einem Daten-Provider. Der VisualizationClient sendet eine Nachricht an den MessageBroker des Servers, der beim Daten-Provider das geforderte Objekt abruft und über eine Antwortnachricht an den VisualizationClient zurücküberträgt....111
Abbildung 81: Mit einer Plexiglasplatte ist der Kinect-Sensor auf einer Höhe von 1,60 m über dem Boden befestigt (links). Die Interaktionsbereiche vor der Powerwall sind durch Klebebandstreifen am Fußboden markiert (rechts). Ab 90 cm Abstand zur Powerwall beginnt der Nah-Interaktionsbereich, ab 120 cm der Fern-Interaktionsbereich. 113
Abbildung 82: Struktureller Aufbau des Prototyps zur Visualisierung und Interaktion mit Eyetracking-Daten. Über das Kommunikationsframework kommunizieren die Serverkomponente, der Eyetracking-Daten-Provider und der Visualisierungs-Client. An den Computer, auf dem der Visualisierungs-Client installiert ist, sind die Powerwall und ein Microsoft Kinect-Sensor angeschlossen. 114

Abbildung 83: Schatteninterface für den Prototyp zur Analyse von Eyetracking-Studien. Links ohne Interaktionsn, rechts mit erkannter Interaktion und Hilfestellungen. 115
Abbildung 84: Papierkorb-Symbol des Schatteninterfaces. Links inaktiv, rechts aktiv. Wird eine Ansicht auf dem Papierkorb abgelegt, wird sie gelöscht. 115
Abbildung 85: Beispielvisualisierungen für Eyetracking-Daten. Bilddarstellung des Stimulus (A), Darstellung der Fixationen auf einem Stimulus in einer Heat-Map (B) und einem Gaze-Duration-Sequence-Diagramm (C). 116
Abbildung 86: Eyetracking Datenmodell für den Daten-Provider des Prototyps. Die Pfeile zwischen den Klassen stellen 1:N-Beziehungen dar. 118
Abbildung 87: Klassendiagramm des Daten-Providers für Eyetracking-Daten. Im Programm wird ein Service erstellt, der über die Schnittstelle IDataProviderService Daten aus dem Eyetracking-Datenmodell zur Verfügung stellt. 119
Abbildung 88: Programmablauf beim Start des Powerwall-Prototyps zur Analyse von Eyetracking-Studien. Zuerst wird eine Verbindung zum Server hergestellt (A), anschließend werden die Übersichtsinformationen über alle Stimuli vom Daten-Provider abgerufen (B) und schließlich die Stimuli abgerufen und auf der Powerwall angezeigt (C). 123
Abbildung 89: Zustandsautomat zur Unterscheidung zwischen Nah- und Ferninteraktion. Je nach Distanz zum Sensor wechselt der Zustandsautomat zwischen den Interaktionsbereichen *Fern-Interaktion* und *Nah-Interaktion*. In jedem Zustand werden entsprechende Hilfestellungen zur Interaktion angezeigt. 125
Abbildung 90: Zustandsautomat zur Erkennung von Gesteninteraktionen mit Powerwall-Visualisierungen im Fern-Interaktionsbereich. 126
Abbildung 91: Zustandsautomat zur Erkennung von Gesteninteraktionen mit Powerwall-Visualisierungen im Nah-Interaktionsbereich. 128
Abbildung 92: Struktureller Aufbau der multimodalen Interaktionsumgebung mit Powerwall-Interaktion und Tabletop-Computer. Über das Kommunikationsframework können sich Visualisierungs-Clients auf Tabletop-Computern mit dem Visualisierungs-Client der Powerwall austauschen. 128
Abbildung 93: Versuchsaufbau der Benutzerstudie zur Findung und Bewertung von Gesteninteraktionen. 131
Abbildung 94: Ausgangsdarstellung der Beispielvisualisierung (links) und Zieldarstellung (rechts) bei der Interaktion "Rotieren". 134
Abbildung 95: Interaktionsaufgaben im zweiten Studienabschnitt. Zu jeder Interaktion sollte die implementierte Geste bewertet werden. 134
Abbildung 96: Gemittelte Einschätzung der Studienteilnehmer der Gesten aus den Aufgaben 1.1 bis 1.7. in den Eigenschaften „kompliziert“ und „ermüdend“ auf einer Punkteskala von 0 bis 20. 137
Abbildung 97: Bewertung der Aufgaben 2.1 bis 2.7 nach den Eigenschaften „passend“, „intuitiv“ und „leicht zu lernen“. 138
Abbildung 98: Bewertung der Aufgaben 2.1 bis 2.7 nach den Eigenschaften „kompliziert“, „umständlich“ und „peinlich. 139
Abbildung 99: Bewertung der Aufgaben 2.1 bis 2.7 nach Kriterien des NASA TLX. 140
Abbildung 100: Beurteilung der Teilkonzepte "Interaktionsbereiche", "Schatteninterface" und "Hilfestellungen". 141
Abbildung 101: Vergleich der Beurteilungen der Gesteninteraktionen in der Eigenschaft "kompliziert". Beurteilung im ersten Studienabschnitt (blau) und im zweiten Studienabschnitt (rot). 143

Abbildung 102: Vergleich der Beurteilung der Gesteninteraktionen in der Eigenschaft „ermüdend“ bzw. „physical Demand“. Beurteilung im ersten Studienabschnitt (blau) und im zweiten Studienabschnitt (rot). 144
Abbildung 103: Optimiertes Feedback zur Interaktion "Ansicht wechseln". Sobald die zweite Hand über das Objekt bewegt wird, werden oben, unten, links und rechts die Ansichten eingeblendet, zu denen gewechselt werden kann. Mit einer Wischgeste nach links kann die erste Ansicht auf der rechten Seite in das Objekt gezogen werden. 147
Abbildung 104: Handerkennung durch Differenzbild mit einem statischen Hintergrund (Izadi, et al., 2011). 153
Abbildung 105: Hand- und Fingererkennung mithilfe eines Abstandsprofils. 153
Abbildung 106: Erkennung zweier interagierender Hände. RGB-Bild, Tiefenbild, Hände im Tiefenbild, 3D-Modelle der Hände, Überlagerung der erkannten Hände mit den Modellen (v.l.n.r.) (Oikonomidis, Kyriazis, & Argyros, 2012).....154

Tabellenverzeichnis

Tabelle 1: Piktogramme zur Darstellung der Interaktionen mit Powerwall-Visualisierungen. 80
Tabelle 2: Teilnehmer an der Qualitativen Pilotstudie zur Gestenfindung und Gestenbewertung. Neben demografischen Angaben wurde die Verwendung von Gesten erkennenden Spielekonsolen und Touch-Geräten erhoben. 130
Tabelle 3: Aufgaben im ersten Studienabschnitt. Zu diesen sieben Interaktionen sollten die Teilnehmer Gesteninteraktionen entwickeln. 131
Tabelle 4: Anordnung der Interaktionen von „leicht“ nach „schwer“. Die Interaktion „Objekt löschen“ wurde von den Teilnehmern als leichteste, die Interaktion „Objektinhalt manipulieren“ als schwerste eingeschätzt. 137
Tabelle 5: Legt man die Antworten der Teilnehmer der Pilotstudie zugrunde, müsste so die Anordnung der Interaktionen nach Schwierigkeit aussehen. Dieses Ergebnis weicht von den Angaben der Teilnehmer in Aufgabe 1.8 (vgl. Tabelle 4) ab. 142

Quellcodeverzeichnis

Quellcode 1: Implementierung eines Daten-Providers in einer Konsolenanwendung. 97
Quellcode 2: XML-Markup in einer Anwendungskonfigurationsdatei, mit der Visualisierungs-Plugins für die Anwendung aktiviert werden.....100
Quellcode 3: Beispiel einer XML-Konfiguration eines Visualisierungs-Plugins. 101
Quellcode 4: Beispielcode zum Erstellen eines Visualisierungs-Clients und der Verwendung von Plugins zur Visualisierung. 103
Quellcode 5: Verbindungszeichenfolge in der Anwendungskonfigurationsdatei für das ADO.NET EntityFramework zum Zugriff auf die Eyetracking-Daten in einem SQL Server. 120
Quellcode 6: Verbindung zur Serverkomponente in der Anwendungskonfigurationsdatei des Eyetracking-DataProviders. 122
Quellcode 7: XML-Markup für die Visualisierungs-Plugins des Eyetracking-Prototyps. Die Visualisierungen werden beim Programmstart geladen. 122

1 Einleitung

The most profound technologies are those that disappear.
They weave themselves into the fabric of everyday life
until they are indistinguishable from it.

Mark Weiser

1.1 Motivation

„You are the controller“ (deutsch: „du bist der Controller“) – mit diesem Slogan bewirbt Microsoft das neueste Zubehör für die Spielekonsole XBOX 360 (Microsoft, 2010). Mit der XBOX Kinect Kamera sollen Gamepads und andere Controller mit Knöpfen und Steuerkreuz der Vergangenheit angehören: statt konventionelle Controller zu verwenden, wird eine Kamera ober- bzw. unterhalb des Bildschirms vor den Spielern positioniert. Die Kamera zeichnet alle Körperbewegungen des Spielers auf, die sie vor dem Fernsehgerät ausführen und übersetzt die Bewegungen in Aktionen des Charakters im Konsolenspiel. So erhält der Spieler mit den Bewegungen seiner Hände, Arme und Füße die Kontrolle über die Spielfigur.

Gleichzeitig werden seit dem Jahr 2000 jährlich weltweit zwischen 1 und 2 Exabytes an neuen Daten aus wissenschaftlichen Simulationen, Bildern, Videos und anderen Informationen generiert (Lyman & Varian, 2000). Um mit dieser Masse und Vielfalt an Daten umgehen zu können, werden immer neue Visualisierungsformen entwickelt. Spezielle Visualisierungen werden dabei zur Anzeige großer Mengen unstrukturierter Daten wie Textdokumenten oder Videos benötigt (Keim, 2002). Darüber hinaus ist zur Anzeige großer Datenmengen ausreichend Platz für die Darstellung nötig, über den Desktop-Bildschirme nicht mehr verfügen (Fekete & Plaisant, 2002). Großflächige, hochauflösende Displays (Powerwalls) bieten mit einem Vielfachen der Auflösung von Desktop-Bildschirmen mehr Darstellungsfläche. Da Powerwalls immer öfter eingesetzt werden, liegen Forschungsschwerpunkte auf der Entwicklung neuer Darstellungen für Powerwalls und der Interaktion mit großen, hochauflösenden Displays (Czerwinski, Robertson, Meyers, Smith, Robbins, & Tan, 2006).

Eine Inspiration zur innovativen Interaktion mit Daten liefert der Science-Fiction-Thriller „Minority Report“: Tom Cruise analysiert in der Rolle des Polizei-Chiefs John Anderton mit Hand- und Armbewegungen Video-, Foto- und anderes Informationsmaterial, um potentiellen Mördern auf die Spur zu kommen (Spielberg, 2002). Auch wenn dieses Szenario lediglich eine Fiktion ist, wirft es doch die Frage auf ob sich die Freihand-Gesteninteraktion auch für eine Anwendung außerhalb der Spielewelt eignet. Lassen sich beispielsweise wissenschaftliche Simulationen und andere große Informationsmengen auf großen Displays effizient darstellen und mithilfe von Körperbewegungen manipulieren? Eine solche Interaktionsumgebung fügt sich in die Vision von Mark Weiser ein, in der sich Computer und die Interaktion mit Computern in der Umgebung des Benutzers integrieren und eine „natürlichen“ Umgang mit Daten und Visualisierungen ermöglichen (Weiser, 1991).

Da das Gebiet der Freihandgesteninteraktion mit Powerwalls noch kaum erforscht ist, soll in dieser Diplomarbeit ein Konzept zur Gesteninterkation mit Visualisierungen komplexer Daten auf einem großen Display entwickelt werden. Dazu werden mithilfe einer Benutzerstudie passende Gesten gesucht, die eine Interaktion mit Powerwall-Visualisierungen ermöglichen. Darüber hinaus soll eine multimodale Interaktionsumgebung geschaffen werden, die neben der Gesteninteraktion mit Powerwall-Visualisierungen beispielsweise auch die Interaktion durch Tabletop-Computer unterstützt.

1.2 Aufbau

Diese Arbeit führt mit Kapitel 2 in die Grundlagen der Visualisierung und der Interaktion ein. Es werden speziell die gestenbasierte Interaktion sowie die Kategorisierung und Entwicklung von Gesten vorgestellt. Zur Erkennung von Freihandgesten werden unterschiedliche Tracking Systeme erläutert und anschließend die Technologie des Micrsoft Kinect-Sensors zur Personenerkennung, dem Skelett-Tracking und der Handerkennung analysiert. Schließlich werden Hardwarekonfigurationen und Anwendungsgebiete von Powerwalls, sowie aktuelle Forschungsfelder auf dem Gebiet der großen hochauflösenden Displays und das Konzept der Natural User Interfaces beschrieben.

Kapitel 3 erläutert die Aufgabenstellung und die einzelnen Schritte der Entwicklung des Lösungskonzepts. Darüber hinaus werden die bisherigen Vorarbeiten erläutert.

Im vierten Kapitel werden Anforderungen an ein System zur Gesteninteraktion mit Powerwall-Visualisierungen formuliert. Zur Findung dieser Anforderungen werden Beobachtungen aus der Auswertung einer Eyetracking-Studie am Institut für Visualisierung und Interaktive Systeme der Universität Stuttgart herangezogen, bei der eine Powerwall zum Einsatz kam. Um eine spätere Auswertung der gefundenen Anforderungen zu ermöglichen, werden zwei Szenarien entwickelt, in denen ein gestengesteuertes System Verwendung findet. Das erste Szenario zeigt das Konzept zur Gesteninteraktion mit Powerwall-Visualisierungen bei der Auswertung einer Eyetracking-Studie, das zweite Szenario zur Interaktion mit wissenschaftlichen Visualisierungen von Molekülen.

Basierend auf den Grundlagen aus Kapitel 2 und den Anforderungen in Kapitel 4 wird in Kapitel 5 ein detailliertes Konzept zur Gesteninteraktion mit Visualisierungen auf einer Powerwall vorgestellt. Dieses Konzept enthält Paradigmen zum Auslösen und Beenden von Interaktionen, sowie bestimmte Gesten zur Interaktion mit Powerwall-Visualisierungen. Ebenfalls werden im Sinne eines Natural User Interfaces Hilfestellungen für den Benutzer formuliert.

In Kapitel 6 wird eine Architektur vorgestellt, die zur Realisierung eines Prototyps zur Gesteninteraktion mit Powerwall-Visualisierungen verwendet werden kann. Gleichzeitig ermöglicht diese Architektur die Integration der Powerwall-Interaktion in eine multimodale Interaktionsumgebung.

Der Prototyp, der im Rahmen dieser Diplomarbeit entwickelt wurde, um das in Kapitel 5 vorgestellte Konzept beispielhaft umzusetzen, ist in Kapitel 7 dokumentiert. Der Prototyp implementiert die Möglichkeit mit Eyetracking-Visualisierungen durch Freihandgesten zu interagieren.

Kapitel 8 beschreibt den Aufbau und die Durchführung einer Benutzerstudie, in der das Interaktionskonzept aus Kapitel 5 und dessen Umsetzung im Prototyp bewertet werden sollen. Zusätzlich sollen von den Probanden Gesten zur Interaktion mit Powerwall-Visualisierungen vorgeschlagen werden. Im Rahmen dieser Diplomarbeit wurde eine Pilotstudie mit vier Probanden durchgeführt. Die Ergebnisse dieser Vorabstudie sind ebenfalls in Kapitel 8 dokumentiert.

Abschließend fasst das Kapitel 9 diese Diplomarbeit zusammen und bewertet deren Ergebnisse. Darüber hinaus wird ein Ausblick gegeben, wie die Weiterentwicklung des Konzepts zur Gesteninteraktion mit Powerwall-Visualisierungen gestaltet werden kann.

2 Grundlagen zur Gestensteuerung Powerwall-basierter Visualisierungen

Um ein umfassendes Verständnis zur Gestensteuerung von Visualisierungen auf Powerwalls zu schaffen, wird in diesem Kapiteln auf folgende grundlegende Konzepte eingegangen: Zunächst werden die Begriffe „Visualisierung“ und „Interaktion“ und deren Zusammenhänge erläutert. Anschließend wird aus dem allgemeinen Begriff der „Interaktion“ die konkrete Interaktionsform mittels Gesten herausgearbeitet und erklärt.

Damit die Interaktion mit Visualisierungen auf einer Powerwall durch Gesten realisiert werden kann, werden anschließend die im Rahmen dieser Arbeit benötigten Hardware- und Softwarekomponenten beschrieben.

2.1 Visualisierung

1987 wurde der Begriff der Visualisierung in einem Bericht des Workshops „Visualization in Scientific Computing“ der National Science Foundation folgendermaßen beschrieben:

„Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe *their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. In many fields it is already revolutionizing the way scientists do science. […] The goal of visualization is to leverage existing scientific methods by providing new scientific insight through visual methods.”* (McCormick, DeFanti, & Brown, 1987)

„Der Vorgang der Visualisierung ist eine Berechnungsmethode. Sie transformiert symbolische Informationen in geometrische und ermöglicht Forschern somit Simulationen und Berechnungen zu beobachten. Visualisierung bietet die Möglichkeit das Unsichtbare zu sehen, bereichert den Prozess wissenschaftlicher Entdeckungen und erlaubt tiefgründige und auch unerwartet Einblicke. In vielen Bereichen revolutioniert sie bereits die wissenschaftliche Forschungsarbeit. […] Das Ziel von Visualisierungen ist es, bestehende Methoden durch visuelle Einblicke zu erweitern.“

Die Konstruktion einer Visualisierung aus zunächst unsichtbaren Daten kann durch die sogenannte „Visualisierungspipeline“ dargestellt werden (siehe Abbildung 1).

Als Basis für jede Visualisierung dienen Daten, die beispielsweise durch Simulationen, Messungen oder aus bestehenden Datenbanken beschafft werden. Da die Betrachtung der Rohdaten zu unübersichtlich ist, sollen diese Daten grafisch dargestellt werden. Im ersten Verarbeitungsschritt, der Datenerfassung (data acquisition) wird entschieden, welche Daten aus welchen Quellen für eine Visualisierung infrage kommen. Dabei wird die Repräsentation der Daten auf eine gemeinsame Datenstruktur vereinheitlicht. Es entsteht die Menge der Rohdaten zur Visualisierung (raw data). Der nächste Schritt (filtering) wählt aus dieser aggregierten und homogenen Datenmenge diejenigen Daten aus, die in der Visualisierung dargestellt werden sollen (visualization data). Anschließend werden den einzelnen Komponenten der Daten visuelle Repräsentationen wie Form, Farbe oder Position zugeordnet (mapping). Damit stehen alle Parameter der darzustellenden Informationen in fest (renderable representation). Der letzte Schritt, das Rendering, stellt diese Informationen visuell auf einem bestimmten Ausgabegerät dar (visualizations).

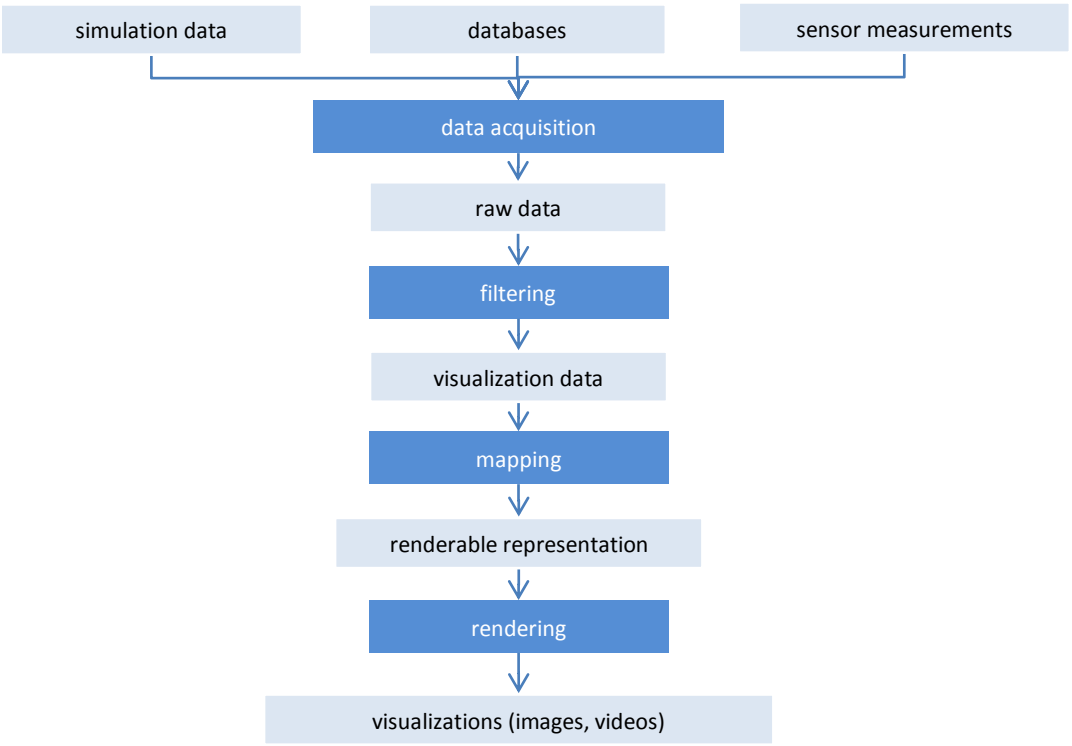


Abbildung 1: Visualisierungspipeline (Ertl, 2010).

In der Regel ist die Visualisierungspipeline Teil eines Analysekreislaufs. Die entstandene Visualisierung wird begutachtet und bewertet. Es entstehen neue Anforderungen an die visuelle Darstellung – andere Farben, andere Positionen, andere Größen oder sogar die Darstellung anderer Daten. Aufgrund dieser Anforderungen werden bei Bedarf in der Datenbeschaffung (data acquisition) grundlegend neue Daten beschafft, die Parameter der Verarbeitungsschritte (filtering, mapping und rendering) angepasst und die Visualisierungs-Pipeline erneut durchlaufen (vergleiche (The Visualization Handbook, 2005)).

2.1.1 Wissenschaftliche Visualisierungen und Informationsvisualisierung

Ein Teilgebiet der Visualisierung beschäftigt sich mit der visuellen Darstellung wissenschaftlicher Daten. Wissenschaftliche Daten besitzen eine feste Struktur: Zum einen ist zu jedem Messwert die Position in einem zwei- oder dreidimensionalen Raum bekannt, an dem diese Messung vorgenommen wurde. Zum anderen sind die Art und Dimension der Messwerte bekannt, die an diesen Positionen erhoben wurden. Dies könnte zum Beispiel die Stärke und Richtung der Strömung in einem Windkanal an beliebig vielen Messpunkten im Raum des Windkanals sein. Die gemessenen Werte werden hierbei als dreidimensionaler Vektor gespeichert.Da die Messwerte kontinuierliche Daten abbilden, kann zwischen diesen Werten interpoliert werden (The Visualization Handbook, 2005).

Durch Raumkoordinaten strukturierte Messdaten erlauben die Interpolation von Werten zwischen mehreren Messpunkten, an Positionen wo keine Messung stattgefunden hat. Die Interpolation findet in der Visualisierungs-Pipeline im Verarbeitungsschritt „filtering“ statt. Im selben Verarbeitungsschritt ist es möglich bestimmte Bereiche bei der Visualisierung auszulassen, oder beispielsweise auf ganze Dimensionen des Vektors zu verzichten. Im Schritt „mapping“ können die Messdaten grafischen Repräsentationen zugeordnet werden: zum Beispiel kann die Windrichtung anhand eines Pfeiles und die Windstärke durch die Länge oder Färbung dieses Pfeiles dargestellt

werden. Wird ein dreidimensionales Bild in dieser Visualisierung erstellt, könnte ein solcher Pfeil seinen Ursprung im zugehörigen Messpunkt haben. Richtung und Länge wird durch die Messwerte bestimmt. Im Rendering-Prozess wird das so konstruierte Bild auf einem Ausgabemedium, beispielsweise einem Bildschirm, dargestellt.
Eine auf diese Weise konstruierte Grafik von Strömungen um ein Fahrzeug in einem Windkanal ist in Abbildung 2 abgebildet.

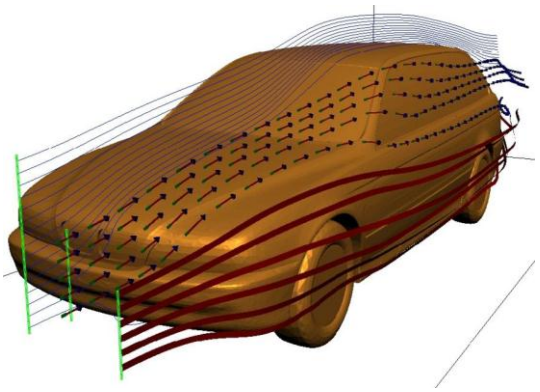


Abbildung 2: Visualisierung von Strömungsdaten um ein Fahrzeug in einem Windkanal (Ertl, 2010).

Neben den wissenschaftlichen Visualisierungen existiert das Feld der Informationsvisualisierung. Der bedeutende Unterschied zwischen diesen beiden Visualisierungsgebieten besteht in der Struktur der Daten, die visualisiert werden sollen. Im Gegensatz zu den Daten, welche die Grundlage für wissenschaftliche Visualisierungen bilden, müssen Daten zur Informationsvisualisierung keine fest definierte Position im 2D- oder 3D-Raum haben. Jedes einzelne Datum kann darüber hinaus unterschiedliche Eigenschaften und somit eine unterschiedlich große Struktur haben. Ein Vergleich zwischen solchen Daten ist nicht immer möglich. Ebenso ist es nicht möglich zwischen einzelnen Datenpunkten zu interpolieren, denn es ist nicht gewährleistet, dass zwischen den einzelnen Daten ein räumlicher Zusammenhang besteht (The Visualization Handbook, 2005).

Heutzutage werden jährlich schätzungsweise 1 Exabyte (1 Million Terabytes) solcher unstrukturierten Daten generiert (The Visualization Handbook, 2005). Hierzu zählen beispielsweise Texte, Dokumente, Webseiten und Bücher sowie Bilder, Videos oder Audiodaten.
Das primäre Ziel der Informationsvisualisierung besteht darin die Daten zunächst in der Datenerfassung (data acquisition) und der Filterung (filtering) in einen Zusammenhang zu bringen, Verbindungen und Strukturen zu finden. Anschließend besteht die Herausforderung darin, für die Daten im Schritt „mapping“ eine passende Darstellungsform zu finden. Da die zu visualisierenden Informationsdaten beispielsweise keine physikalische Bedeutung haben (wie die gemessenen Windgeschwindigkeiten und Windrichtungen im Windkanal), sind passende Darstellungsformen schwer zu finden. Diese Schwierigkeiten wirken sich auch auf die Interaktion zur Erstellung und Analyse dieser Visualisierungen aus.

Abbildung 3 zeigt beispielhaft die Visualisierung von Dateiinhalten auf einer Festplatte, die das Suchwort „idea“ beinhalten. Hierbei werden im Verarbeitungsschritt „data acquisition“ die Textinhalte aus den verschiedenen Dateiformaten auf der Festplatte extrahiert. Im Schritt „filtering“ werden alle Dateien verworfen, die nicht das gesuchte Wort „idea“ im Text beinhalten. Die verbleibenden Dateien werden weiter strukturiert und zu Stichwortgruppen zusammengefasst. Je nach Häufigkeit der Stichwörter werden die Gruppen auf Kreissegmente abgebildet (Arbeitsschritt „mapping“) und im letzten Schritt „rendering“ in unterschiedlicher Farbe und Größe dargestellt.

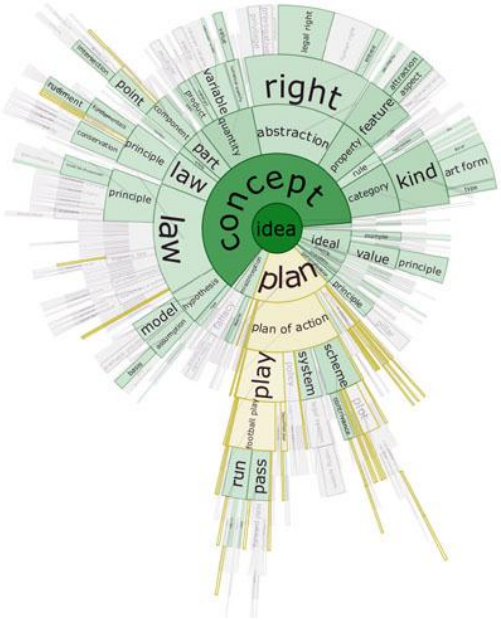


Abbildung 3: Beispiel aus der Informationsvisualisierung. Strukturierung der Inhalte von Dateien mit dem Stichwort „idea“. Je mehr Dokumente ein mit „idea“ verwandtes Stichwort beinhalten, desto näher an der Mitte wird dieses Stichwort in einem Bogensegment dargestellt. Je dunkler die Farbe des Segments ist, desto häufiger kommt das jeweilige Stichwort in den Dokumenten vor (Collins, 2006).

2.2 Interaktion

In diesem Abschnitt werden zunächst die Interaktion allgemein und die gestenbasierte Mensch-Computer-Interaktion beschrieben. Anschließend werden die Besonderheiten von Touch- und Freihandgesten vorgestellt und schließlich Anforderungen bei der Entwicklung von Gesteninteraktionen herausgearbeitet.

2.2.1 Interaktion

Das Wort „Interaktion“ leitet sich aus dem lateinischen „inter agere“ ab. „Inter“ bedeutet übersetzt „zwischen“ und „agere“ „handeln“ oder „verhandeln“. Bezogen auf Computersysteme besteht die Verhandlung zwischen Programm und Benutzer aus der Kommunikation zum Austausch von Eingabe- und Ausgabeinformationen, um ein gemeinsames Ziel zu erreichen (Dahm, 2006). Der Benutzer liefert Eingabeinformationen in Form von Daten und Parametern, daraus berechnet das Programm ein Ergebnis und informiert den Benutzer über spezielle Ausgaben. Das Ergebnis der Berechnung wird vom Benutzer interpretiert und so neue Informationen als Eingabe für den Computer gefunden (siehe Abbildung 4).

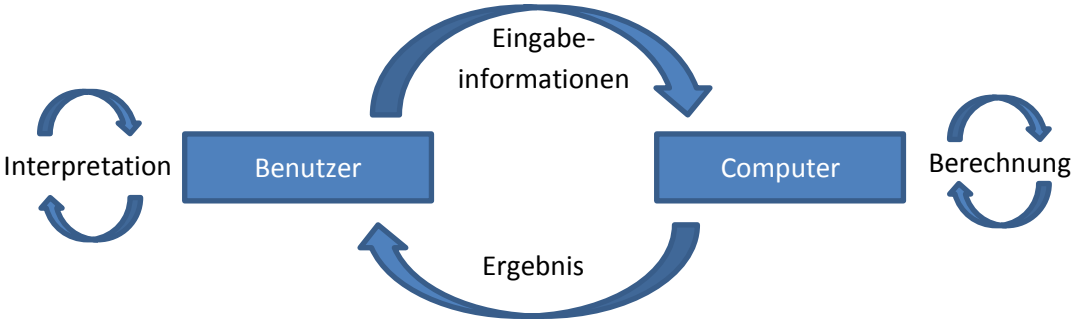


Abbildung 4: Interaktion zwischen Benutzer und Computer. Der Benutzer gibt Informationen in den Computer ein, der durch Berechnungen ein Ergebnis erzeugt. Das Ergebnis wird vom den Benutzer interpretiert und daraus neue Eingabeinformationen für die nächste Berechnung abgeleitet. Vergleiche (Dix, Finlay, Abowd, & Beale, 2004).

Als Interaktion mit einer Visualisierung kann also bezogen auf den Analysekreislauf folgendermaßen gesehen werden: Der Benutzer greift in den Schritten „data acquisition“, „filtering“, „mapping“ und „rendering“ in die Parameter der Visualisierung ein. Der Computer liefert das visuelle Ergebnis, worauf der Benutzer die Parameter wiederum verändern kann und bestimmt, welche Daten an welcher Position in welcher Form dargestellt werden. Abbildung 5 zeigt die Visualisierungspipeline, die um die Interaktionsmöglichkeiten des Benutzers erweitert wurde.

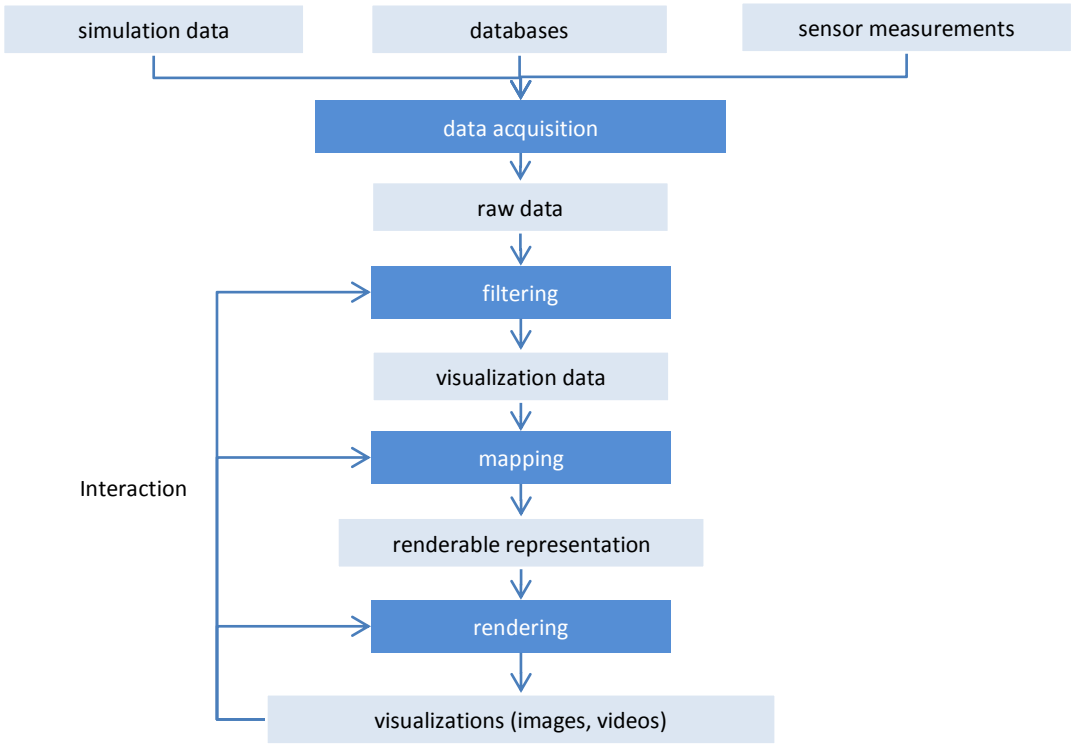


Abbildung 5: Erweiterung Visualisierungs-Pipeline um die Interaktion des Benutzers mit Visualisierungen (vgl. Abbildung 1). Aus der Interpretation der Visualisierungen leitet der Benutzer neue Parameter für die Schritte "filtering", "mapping" und "rendering" ab und führt die Änderungen durch Interaktionen aus.

Es können zwei Stile der Interaktion unterschieden werden: der indirekte Interaktionsstil und der direkte Interaktionsstil. Diese beiden Interaktionsstile werden im Folgenden vorgestellt:

Bei der Dateneingabe am Computer – und somit auch zur Interaktion mit Computerprogrammen – haben sich Maus und Tastatur zum gängigen Standard entwickelt. Mit einer Tastatur lassen sich bequem Texte oder Zahlen eingeben, die Maus ist zur Auswahl, Markierung und Positionierung von Elementen geeignet. Um beispielsweise ein Icon oder ein Fenster auf dem Bildschirm zu verschieben, bewegt der Benutzer die Maus auf der Tischfläche in die gewünschte Richtung. Der Mauscursor und das Icon auf dem Bildschirm bewegen sich entsprechend mit. Auch bei der Texteingabe bedienen die Finger des Anwenders Tasten, die weit entfernt von der Stelle am Bildschirm sind, wo der Text schließlich erscheint. Daher spricht man hier von einem *indirekten Interaktionsstil* (Shneiderman, 2002). Eine Ausprägung des indirekten Interaktionsstils sind die sogenannten WIMP-Interfaces verwendet. Das Akronym WIMP steht für „Windows, Icons, Menus and Pointers“ (Green & Jacob, 1990). WIMP-Interfaces dominierten seither die Interaktion mit Computerprogrammen (Hinckley, 1996).

Direkte Interaktion hingegen bedeutet, dass auf das zu manipulierende Objekt direkt zugegriffen wird, oder der Anwender zumindest glaubt ohne Umwege damit zu interagieren. Auf Mediatoren wie

eine Maus, die eine Bewegung auf dem Tisch auf den Bildschirm des Computers überträgt, wird verzichtet. Ein frühes Beispiel für direkte Interaktion ist der Lichtstift (Sutherland, 1964). Mit diesem Stift kann auf ein Objekt, das auf einem Röhrenbildschirm angezeigt wird, gedeutet und so mit ihm interagiert werden. Im Gegensatz zur Maus zeigt der Benutzer hier direkt auf das zu manipulierende Objekt.
Neue und günstigere Technologien eröffnen ein weites Anwendungsfeld für direkte Manipulation. Berührungsempfindliche Oberflächen lassen Benutzer ohne Zeigegeräte wie Maus oder Lichtstift direkt auf angezeigte Objekte zeigen und manipulieren (Shneiderman, 1983). Moderne Touchscreens erkennen sogar mehrere Berührungen zeitgleich und stellen den Benutzern dadurch mehr Interaktionsmöglichkeiten zur Verfügung (Wu & Balakrishnan, 2003). Bei geeigneter Größe können auch mehrere Personen solche Geräte gleichzeitig bedienen. SmartPhones und Tablet-PCs, die diese Technik nutzen, erfreuen sich großer Beliebtheit: im Jahr 2012 werden voraussichtlich 15,9 Millionen Smartphones in Deutschland verkauft werden. Im Vergleich zum Jahr 2008 wäre dies eine Verdreifachung des Absatzes von Smartphones (statista, 2012).

2.2.2 Gestenbasierte Interaktion

Die direkte Manipulation mit beispielsweise Touchscreens oder 3D-Kameras erfolgt zum Beispiel durch sogenannte „Gesten“. Dan Saffer definiert in seinem Buch „Designing Gestural Interfaces“ eine Geste folgendermaßen:

„A gesture, […], is any physical movement that a digital system can sense and respond to without the aid of a traditional pointing device such as a mouse or stylus. A wave, a head nod, a touch, […], and even a raised eyebrow can be a gesture.” (Saffer, 2009)

„Eine physische Bewegung, die ein digitales System erkennen und darauf reagieren kann ohne, dass ein klassisches Zeigegerät wie eine Maus oder ein Stylus verwendet werden. Ein Wink, Kopfnicken, eine Berührung, […], selbst eine gehobene Augenbraue kann eine Geste sein.“

Die Definition einer Geste nach Saffer lässt erkennen, welche Vielfalt hinter dem Begriff „Geste“ stehen kann. Die reine Anwesenheit von Personen kann als implizit ausgeführte Geste verstanden werden (Schmidt, 2000), wenn beispielsweise ein Bewegungssensor das Licht in einem Raum steuert oder eine automatische Türe öffnet. Auch auf öffentlichen Toiletten führen wir – bewusst oder unbewusst – explizite Gesten aus, wenn wir die Hand unter den Seifenspender oder den Wasserhahn halten, die mittels Infrarotsensoren erkennen, wann sie Seife ausgeben oder das Wasser fließen lassen sollen.

Um die Vielfalt der Gesten und Gesteninteraktionen zu klassifizieren entwickelten Karam und Schraefel eine Taxonomie über Gesten (Karam & Schraefel, 2005). In dieser Taxonomie werden Gesten in folgende Dimensionen klassifiziert (Abbildung 6):

- Gesten können in bestimmten **Anwendungsfeldern** (Application Domains) ausgeführt werden. Beispielsweise auf Mobilgeräten, in einer virtuellen Realität, in einem Computerspiel oder einer multimodalen Umgebung.

- Es existieren verschiedene **Technologien** (Enabling Technologies), die das Erkennen und Verwenden von Gesten ermöglichen. Eine Kategorie bilden die klassischen Technologien wie Maus, Tastatur oder Touchscreens. Technologien, die den Benutzer „wahrnehmen“ müssen, um Gesten erkennen zu können werden in einer zweiten Kategorie zusammengefasst. Hierzu

zählen die Computer Vision, die den Benutzer durch Bilderkennung „wahrnimmt“ oder Audio-Schnittstellen, die den Benutzer „hören“ können.

- Auch unterscheiden sich Gesteninteraktionen durch die **Systemantwort** (System Response)**,** die durch sie ausgelöst werden. In der Taxonomie sind hier visuelle Displays, Audioausgabe und CPU Kommandos aufgeführt.

- Gesten können unterschiedlichen **Stilen** (Gesture Styles) zugeordnet werden. Von deiktischen Gesten, die sich auf das Zeigen einer Position oder einer Richtung beschränken, über manipulative Gesten bis hin zur Zeichensprache.

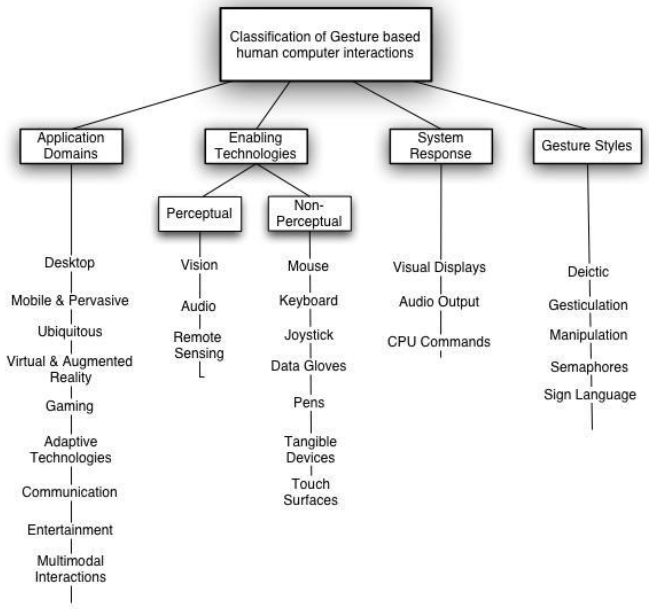


Abbildung 6: Gestentaxonomie nach Karam und Schraefel (Karam & Schraefel, 2005).

2.2.3 Touch- und Freihandgesten

Da ein Touchscreen nur auf Berührungen reagiert, sind der Interkation hier nur Gesten auf einer zweidimensionalen Ebene möglich, bei denen der Benutzer die Oberfläche des Bildschirms berührt. Im anderen Fall sind Gesten mit bis zu sechs Freiheitsgraden im dreidimensionalen Raum möglich. Diese Gesten werden als Freihand- oder In-Air-Gesten bezeichnet (Saffer, 2009).

Wigdor und Wixon beschreiben in (Wigdor & Wixon, 2011) ihre Erfahrungen bei der Entwicklung der Microsoft PixelSense Plattform. Dabei geben sie auch nützliche Tipps und wichtige Anforderungen bei der Entwicklung von gestengesteuerten Systemen. Als führende Entwickler bei Microsoft Surface legen sie ihren Fokus dabei auf Touch-Gesten. Alle Prinzipien der Gestenentwicklung lassen sich auch auf In-Air-Gesten anwenden. In-Air-Gesten dürfen dabei aber nicht als „Touch-Gesten nur ohne Touch“ betrachtet werden. Mit der Berührung ist klar, wann das System eine Geste zu erkennen hat und wann eine Geste beendet ist. Bei In-Air-Gesten fehlt dieses Merkmal und führt zur zentralen Frage:

> Welche Bewegungen sind als vom Benutzer beabsichtigte Gesten zu deuten und welche nicht?

Dieses Problem wird „Live-Mic“ genannt. Zur Lösung dieses Problems nennen Wigdor und Wixon zwei Möglichkeiten. Spezielle Bewegungen können für Gestenkommandos reserviert werden. Dies schränkt den Benutzer allerdings in der Auswahl seiner Bewegungen zur Kommunikation mit Mitbenutzern ein. Alternativ verwendet man eine Bestimmte Geste als sogenannte „Clutch“, eine Kupplung, die dem System zeigt: Jetzt folgt eine weitere Geste, die als Interaktion mit dem System zu interpretieren ist. (Wigdor & Wixon, 2011)
Ein Beispiel für ein System mit Gesteninteraktion, das solche Kupplungen verwendet ist Richard Bolts Projekt „Put-That-There“. In diesem Projekt ist die Kupplung ein Sprachbefehl: Nach den Kommandos „that“ und „there“ erwartet das System eine Zeigegeste auf ein Objekt bzw. eine neue Position (Bolt, 1980).

2.2.4 Entwicklung von Gesten

Nach Wigdor und Wixon bestehen Gesten aus den drei Abschnitten „registration“, „continuation“ und „termination“ (Wigdor & Wixon, 2011). Der erste Abschnitt kennzeichnet den Beginn einer Geste. Hier erkennt das System, dass der Benutzer die Absicht hat eine Geste auszuführen, die erkannt werden soll. Die „continuation“ umfasst die eigentliche Durchführung der Geste. Sie kann als Parametrisierung der Geste bezeichnet werden, wodurch das System die Bedeutung der Geste bestimmt. Der letzte Abschnitt kennzeichnet das Ende der Geste. Anschließend muss das System eine Antwort auf die erkannte Geste liefern.

In Abbildung 7 ist beispielhaft die Erkennung der Touchgeste „Rename“ dargestellt. Sobald der Benutzer mit einem Finger ein Element berührt, erkennt das System den Beginn einer Gesteninteraktion („registration“, Nr. 1 in Abbildung 7). Anschließend bewegt der Benutzer den Finger auf der Touchoberfläche und das Element folgt dieser Bewegung („continuation“, Nr. 2 in Abbildung 7) solange, bis das System die Geste „Rename“ erkannt hat („termination“, Nr. 3 in Abbildung 7).

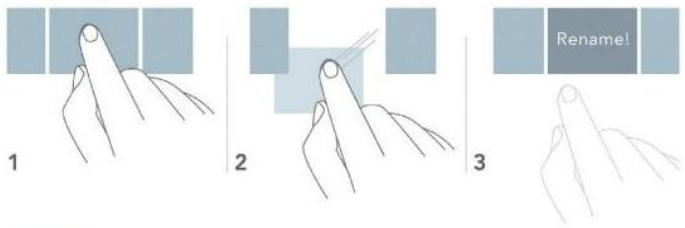


Abbildung 7: Erkennen einer Touchgeste in den drei Schritten "registration" (1), "continuation" (2) und "termination" (3) (Wigdor & Wixon, 2011).

Die Herausforderung liegt hierbei darin, möglichst früh erkennen zu können, welche Geste vom Benutzer ausgeführt wird, um schnelles Feedback (Shneiderman, User Interface Design, 2002) geben zu können. Gesten mit gleichen oder ähnlichen Anfängen müssen schnellstmöglich unterschieden werden können, um entsprechend schnell darauf reagieren zu können. Andernfalls entsteht der Eindruck beim Benutzer, dass das System überhaupt nicht reagiert oder eingefroren ist. Dies würde den Benutzer frustrieren und daran hindern das System in Zukunft weiter zu verwenden.

Die wenigsten Systeme, die eine Gesteninteraktion unterstützen, kommen mit einer einzelnen Geste aus. Die Gesamtheit aller Gesten, die ein System zur Gesteninteraktion anbietet, wird als „Gestensprache“ bezeichnet (Wigdor & Wixon, 2011). Um eine solche Interaktionssprache intuitiv

und schnell erlernen zu können, werden die folgenden Ansprüche an eine solche Sprache gestellt: Sie muss konsistent, kommutativ, negierbar und reziprok sein.

- *Konsistenz*: Eine Geste hat nur eine Bedeutung und erzielt immer dieselbe Systemantwort
- *Kommutation*: Die Reihenfolge einzelner Gesten kann vertauscht werden aber das Ergebnis bleibt gleich. Beispielsweise erzielt das Verschieben und anschließendes Skalieren eines Bildes dasselbe Ergebnis wie zuerst Skalieren und dann Verschieben.
- *Negation*: Aktionen durch Gesten müssen rückgängig gemacht werden können und das System in seinen vorigen Zustand zurückversetzen. Hierbei ist ausdrücklich nicht die Implementierung einer „Rückgängig"-Schaltfläche gemeint. Zu jeder Geste soll es eine andere Gesteninteraktion geben, welche die erste rückgängig macht Zum Beispiel könnte mit einer Geste ein Element aus einer Liste auf eine freie Fläche herausgezogen werden. Die Negation wäre dann, dieses Objekt wieder in die Liste zurückzubewegen.
- *Reziprok*: Im Gegensatz zur Negation werden nur Teilaspekte eines Objekts in den Ausgangszustand zurückversetzt. Reziprok in diesem Sinne wäre beispielsweise ein Bild zunächst in der Horizontalen zu strecken. Dabei wird das Seitenverhältnis verändert. Wird das Bild anschließend in der Vertikalen gestreckt, kann das Seitenverhältnis in den Ausgangszustand zurückversetzt werden.

Zusätzlich zu diesen formalen Ansprüchen an Gesten und Gesten-Sprachen schränken weitere Faktoren die Auswahl und Entwicklung von Gesten ein:

- Die **Sensoren**, die zur Erkennung von Gesten verwendet werden sollen, bestimmen, welche Gesten vom System erkannt und beantwortet werden können.
- Der **menschliche Körper** unterliegt ebenfalls Einschränkungen. Nicht jede denkbare Geste kann von Menschen ausgeführt werden. Gesten, die beispielsweise über den Bewegungsradius von Gelenken hinausgehen, könnten zwar für bestimmte Anwendungen passend sein, aber von den Benutzern nicht durchgeführt werden (Karam & Schraefel, 2005).
- Die Interpretation und Bedeutung von Gesten hängt stark vom **kulturellen Hintergrund** der Personen ab, die diese Gesten ausführen. Entsprechend unterschiedlich können die Erwartungen an Reaktionen auf Gesten sein. Die Geste „Kopfnicken" wird im europäischen Raum in der Regel als Bejahung oder Zustimmung interpretiert. In anderen Kulturkreisen bedeutet sie allerdings das genaue Gegenteil (Karam & Schraefel, 2005).
- Neben dem kulturellen Hintergrund spielt auch die individuelle **Psychologie** der Menschen eine wichtige Rolle. Nicht jeder Anwender möchte auch jede Geste ausführen. In einer Studie konnte Microsoft Research zeigen, dass zum Beispiel auffällige und ausladende Gesten oder Gesten, die mit dem gesamten Körper auszuführen sind, von einer Mehrzahl an Personen sehr ungern ausgeführt werden. Ebenfalls vermieden die Probanden Sprachsteuerung, da unnatürlich laut und deutlich ausgesprochene Satzfragmente als peinlich empfunden wurden. Daher sind dezente, aber dennoch deutliche und eindeutige Gesten, vorzuziehen (Bragdon, DeLine, Hinckley, & Morris, 2011).

- Dan Saffer empfiehlt darüber hinaus, Gesten entsprechend der **Komplexität** der Aufgaben, die durch sie ausgelöst werden soll, zu verwenden. Einfache oder kleine Aufgaben, die häufig ausgeführt werden, sollen durch einfache oder kurze Gesten ausgelöst werden. Komplizierte oder große Aufgaben hingegen durch entsprechend längere oder umfangreichere Gesten (Saffer, 2009).

2.3 3D Tracking Systeme

Damit ein Computer Objekte im dreidimensionalen Raum erkennen und verfolgen kann, wurden verschiedene Systeme entwickelt. Aus zweidimensionalen Sensorinformationen, beispielsweise Bildaufnahmen von Kameras, können durch unterschiedliche Verfahren Informationen über die abgelichtete dreidimensionale Welt berechnet werden. Im Folgenden werden hier die gängigsten Verfahren vorgestellt (Arman & Aggarwal, 1993): Stereo Triangulation, Time-Of-Flight, Sheet Of Light Triangulation und Structured Light Triangulation. Anschließend werden Vor- und Nachteile der jeweiligen Methode sowie Anwendungsbeispiele genannt.

2.3.1 Stereo Triangulation

Bei der „Stereo-Triangulation“ werden gleichzeitig mehrere Bilder derselben Szene aus verschiedenen Blickwinkeln aufgenommen. Dabei muss der Abstand zwischen den Kamerapositionen der verschiedenen Aufnahmen bekannt sein (siehe B in Abbildung 8). In diesen Bildern werden korrespondierende Punkte gesucht. In Abbildung 8 korrespondieren beispielsweise die Punkte (x_1,y_1) in Bild 1 und (x_2,y_2) in Bild 2. Aus der Disparität dieser Punkte und dem Abstand der Kameras bei der Aufnahme der Bilder lässt sich die Distanz zur Kamera des Objekts bestimmen, zu dem die Punkte gehören.

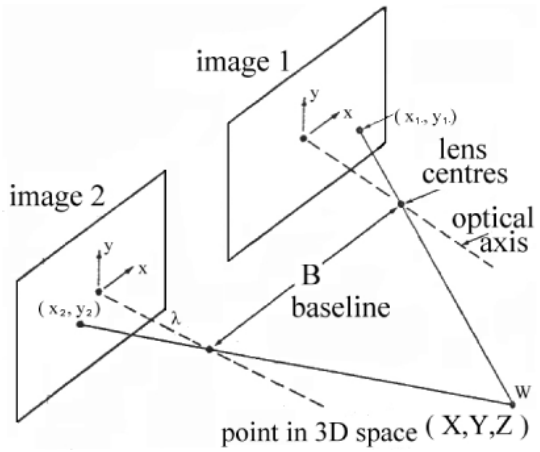


Abbildung 8: Bei der Stereo Triangulation werden übereinstimmende Punkte in mehreren Bildern gesucht und deren Abstand zur Kamera mithilfe der Disparität der Punkte und dem Abstand der Kameralinsen berechnet (Kabayama & Trabasso, 2002).

Dieses Verfahren eignet sich gut um Tiefeneindrücke im nahen Sichtfeld der Kamera zu ermitteln. Um korrespondierende Punkte einfacher erkennen zu können, werden spezielle Marker verwendet, die in den aufgenommenen Bildern leichter gefunden werden können. Diese Technologie macht sich beispielsweise das OptiTrack-System der Firma NaturalPoint Inc. zunutze.

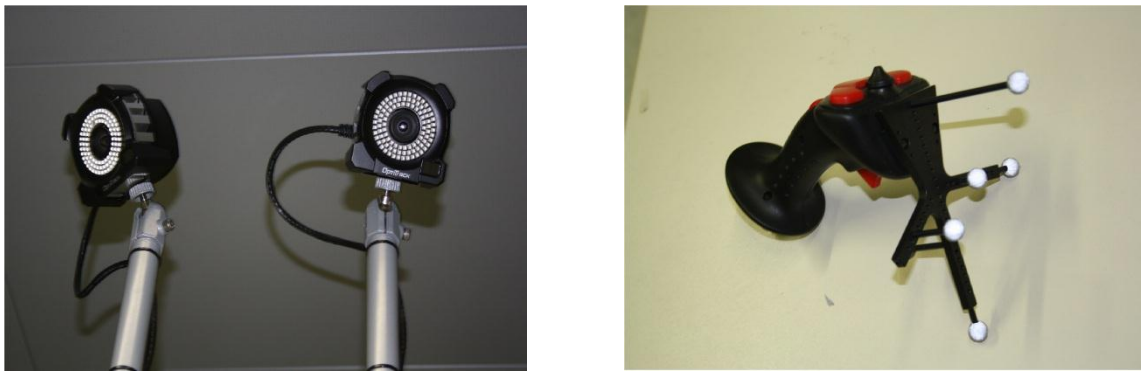


Abbildung 9: Zwei Kameras des OptiTrack-Systems im VR-Labor des Informatikgebäudes der Universität Stuttgart (links). Marker zur Positionserkennung der 3D-Maus (rechts).

2.3.2 Time-Of-Flight und Inferometrie

Die Time-Of-Flight Methode funktioniert wie ein Radar. Statt Radiowellen wird Licht von einer Lichtquelle, beispielweise einem Laser, in die Szene abgestrahlt. Ein Sensor misst, wie lange der ausgesendete Strahl braucht, um von einem Objekt im Raum reflektiert und zum Sensor zurückgeworfen zu werden. Je nach Dichte der beleuchteten Punkte kann so ein hochauflösendes Höhenprofil oder Tiefenbild der Szene erstellt werden.

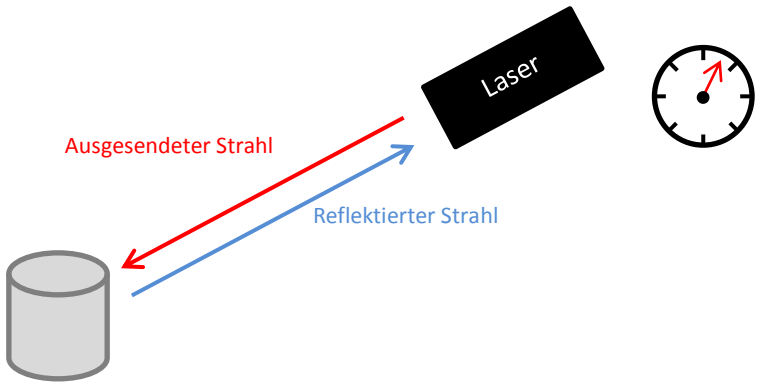


Abbildung 10: Bei der Time-Of-Flight-Methode wird die Zeit vom Aussenden eines Laserstrahls bis zum Eintreffen beim Sensor gemessen um die Distanz zum Objekt zu bestimmen.

Ähnlich wie bei der Time-Of-Flight Methode wird die Reflektion von Licht bei der Interferometrie angewendet: Von einer Lichtquelle wird kohärentes Licht – also Lichtstrahlen mit derselben Wellenlänge – in die Szene abgestrahlt. Wird ein solcher Lichtstrahl von einer Oberfläche reflektiert, wird er gleichzeitig in der Phase verschoben. Durch diese Phasenverschiebung lässt sich die Distanz der reflektierenden Fläche zum Kamerasensor errechnen. (Droeschel, Stückler, & Behnke, 2011)

2.3.3 Sheet Of Light Triangulation

Bei dieser Variante wird ein Lichtstreifen au seiner Lichtquelle auf die Szene projiziert. Wäre die Lichtquelle gleichzeitig eine Kamera, würde das reflektierte Licht hier als eine gerade Linie erkannt werden. Eine an jedem anderen Punkt angebrachte Kamera „sieht“ diese Linie als verformte Kurve. Die Verformung hängt von der Oberfläche des beleuchteten Objekts und dem Betrachtungswinkel der Kamera ab. Ist der Abstand von Lichtquelle und Kamera bekannt, kann die Distanz zu den reflektierenden Punkten des Objekts oder der Szene berechnet werden.

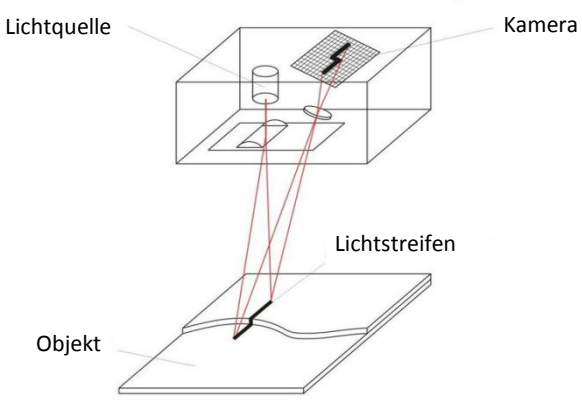


Abbildung 11: Bei der Sheet Of Light Triangulation wird ein Lichtstreifen auf das zu untersuchende Objekt projiziert. Eine Kamera erfasst die Verformung des Lichtstreifens durch die Objektoberfläche.

Der Lichtstreifen beleuchtet nur einen schmalen Teil der Szene oder des Objekts. Damit ein vollständiges Tiefenbild erstellt werden kann, muss entweder die Lichtquelle über das Objekt oder das Objekt unter der Lichtquelle bewegt werden. Am Fraunhofer Institut für Integrierte Schaltungen wurde ein solcher Scanner entwickelt, unter dem Prüfobjekte mit einer Geschwindigkeit von 1 m/s bewegt werden können (Fraunhofer IIS, 2010).

2.3.4 Structured Light Triangulation

Mit einer Abwandlung der Sheet Of Light Triangulation kann auf die Bewegung von Lichtquelle oder Prüfobjekt bzw. Szene verzichtet werden. Eine statische Lichtquelle projiziert ein Muster aus parallelen Lichtstreifen auf die Szene. Die Kamera erfasst gleichzeitig mehrere von den Oberflächen der Szene verformte Kurven. Nimmt man an, dass sich das Höhenprofil der Szene in kleinen Bereichen nur wenig ändert und das projizierte Streifenmuster ist dicht genug, kann zwischen den Kurven interpoliert werden. Somit ist mit einer einzelnen Aufnahme die Berechnung eines Tiefenbildes möglich. Wählt man Infrarotlicht zur Projektion des Streifenmusters, kann das durch die Kamera aufgezeichnete Farbbild ohne Störungen zu weiteren Verarbeitungsschritten der Bilderkennung herangezogen werden. Bei diesem Verfahren ist nur eine Kamera nötig.

Statt eines Streifenmusters kann auch ein Punktemuster verwendet und in die Szene projiziert werden. Das durch den Sensor erkannte Punktemuster wird mit dem ursprünglichen Muster verglichen und aus der Differenz ein Tiefenbild berechnet (Garcia & Zalevsky, 2007).

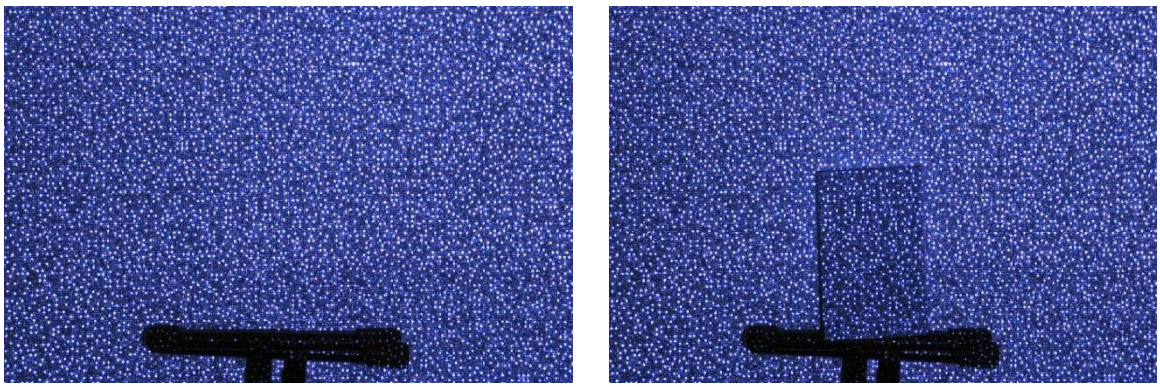


Abbildung 12: Links wird ein Punktemuster in die Szene projiziert. Rechts wird dieses Muster durch ein Objekt verzerrt (Kinect Hacking 103: Looking at Kinect IR Patterns, 2010).

2.4 Microsoft Kinect

Dieses Kapitel stellt den Kinect-Sensor von Microsoft vor. Zunächst werden technische Merkmale, anschließend die verfügbaren Programmierschnittstellen und Bibliotheken vorgestellt, die bei der Entwicklung des Prototyps Anwendung finden.

2.4.1 Technische Details

Der Kinect-Sensor (siehe Abbildung 13) verfügt über folgende technische Merkmale:

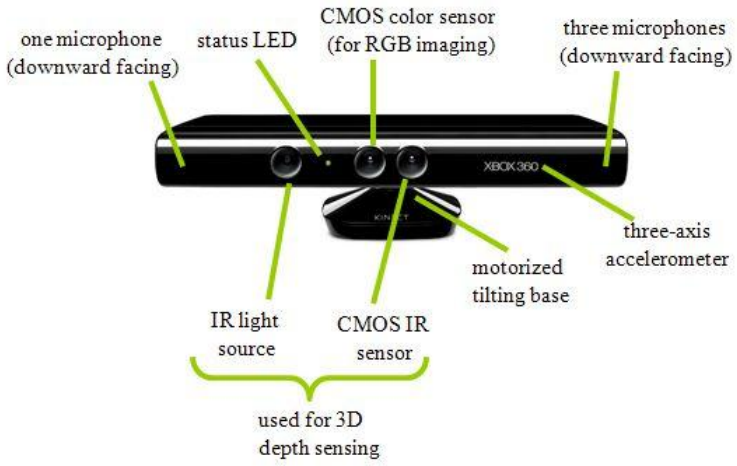


Abbildung 13: Der Microsoft Kinect 3D Sensor verfügt über eine Infrarotkamera und eine Farbkamera zur 3D-Bilderkennung und ein Mikrofonarray, um Geräusche lokalisieren zu können (Davison, 2011).

Eine RGB-Kamera erzeugt einen Video-Stream mit einer maximalen Auflösung von 640x480 Pixeln bei einer Farbtiefe von 32-Bit mit einer Bildrate von 30 Bildern pro Sekunde. Mit derselben Bildrate kann auch ein Video-Stream mit 320x240 Pixeln und 16-Bit Farbtiefe abgerufen werden.

Zusätzlich liefert eine Infrarot-Kamera einen Video-Stream mit Tiefeninformationen. Der Erkennungsbereich dieses Sensors liegt zwischen 80 cm und 4 m Entfernung und liefert einen monochromen Video-Stream mit einer Auflösung von 320x240 Pixeln. In diesem Stream sind Tiefeninformationen mit 2^{13} Abstufungen enthalten.

Der Sichtwinkel des Sensors beträgt 57° in der Horizontalen und 43° in der Vertikalen. Mit einem integrierten Motor kann der Sensor den Winkel in der Vertikalen um 27° in beide Richtungen verschieben. Dies ergibt eine trapezförmige Interaktionsfläche von ca. 6m² im Erkennungsbereich des Sensors.

Ein Array von vier Mikrofonen erlaubt neben der Aufzeichnung von Sprache und Geräuschen eine Lokalisation von Geräuschquellen und eine verbesserte Unterdrückung von Hintergrundgeräuschen. Es kann auf einen 16-Bit Audio-Stream mit 16 kHz zugegriffen werden.

Mittels des eingebauten Chipsatz der Firma PrimeSense werden die aufgezeichneten Informationen bereits im Sensor aufbereitet. So „erkennt“ der Sensor bereits bis zu 6 Personen in seinem Sichtbereich und liefert zu diesen Personen weitere Informationen. Diese Informationen können in Form von Skeletten in einem eigenen Data-Stream abgerufen werden.

Kinect for Windows

Im Gegensatz zu der seit Februar 2010 erhältlichen XBOX Kinect ist seit dem 1. Februar 2012 eine Variante der Kinect erhältlich, die für die Verwendung mit einem Windows-Computer ausgelegt ist. Die technischen Merkmale der Kinect for Windows stimmen bis auf ein zusätzliches Feature mit der XBOX Kinect überein.

Die Kinect for Windows verfügt zusätzlich über einen sogenannten „near mode“. In diesem Modus erkennt die Kinect Objekte ab einem Abstand von 40 cm zum Sensor zuverlässig. Die Grenze der XBOX Kinect liegt bei 80 cm (siehe Abbildung 14). Dafür reduziert sich die maximale Entfernung zur Erkennung von vier auf drei Meter. Zwischen dem „near mode“ und dem „default mode“ kann umgeschaltet werden. Der „default mode“ entspricht dem Erkennungsbereich der XBOX Kinect (Microsoft, 2012).

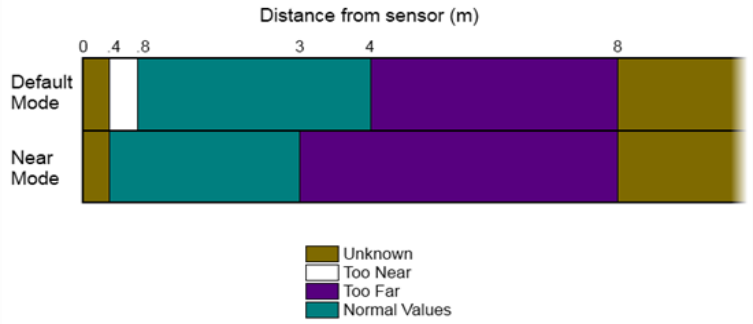


Abbildung 14: Erkennungsmodi der Kinect for Windows. Im Near Mode erkennt der Sensor Objekte, die sich bis zu 40cm nah am Sensor befinden (Microsoft, 2012).

2.4.2 Programmierschnittstellen

Dieser Abschnitt stellt zwei Programmierschnittstellen für den Microsoft Kinect-Sensor vor. Die Open-Source Bibliothek „OpenNI“ und das Microsoft „Kinect for Windows Software Developer Kit”.

OpenNI Framework

Bereits vor der Veröffentlichung der Kinect entwickelte sich ein Wettbewerb darum, die Kinect auch für andere Geräte als die Xbox 360 zu verwenden. Noch im Monat der Veröffentlichung wurde die Organisation OpenNI von mehreren Firmen, darunter auch die Herstellerfirma der Kinect PrimeSense, gegründet. Ziel dieser Organisation ist die Entwicklung eines Frameworks, das als Industriestandard für Interaktionsgeräte wie die Kinect dienen kann (Villaroman, Rowe, & Swan, 2011).

Mit den Treibern der Firma PrimeSense und dem OpenNI-Framework können Computeranwendungen auf die Sensordaten der Kinect zugreifen und diese verarbeiten. Damit ist eine Gestenerkennung und eine Gestensteuerung für den Computer möglich. Dies wurde vor allem von Forschung, Wissenschaft und Privatpersonen genutzt. Einige dieser Ergebnisse werden im Kapitel 2.3.3 vorgestellt, da diese auch für die in dieser Arbeit erstellten Lösung verwendet werden.

Kinect for Windows SDK

Aufgrund des Enthusiasmus, den Forscher und Entwickler der Kinect und den frei verfügbaren Frameworks entgegenbrachten, entschloss sich Microsoft dazu, ebenfalls eine Programmierschnittstelle für die Kinect anzubieten. Das „Kinect for Windows software development

kit (SDK)“ wurde im Rahmen eines „Code Camp“ im Juni 2011 in einer Beta-Version veröffentlicht. Zu Beginn dieser Arbeit stand das Kinect SDK bereits im Beta2-Stadium zur Verfügung (Microsoft, 2011). Am 1. Februar 2012 wurde schließlich „Kinect for Windows“ der Öffentlichkeit zugänglich gemacht. Bis dahin war die Verwendung der Microsoft-Schnittstelle nur für den nicht-kommerziellen Einsatz gestattet. Mit der Version kann diese Schnittstelle auch für kommerzielle Zwecke genutzt werden.

2.4.3 Vorarbeiten

Entwickler, die sich neu mit der Microsoft Kinect beschäftigen, können auf ein breites Spektrum von frei Verfügbaren Bibliotheken und Beispielen zurückgreifen. Es ist kaum noch nötig die Algorithmen zur Bilderkennung selbst zu implementieren. Im Folgenden werden einige Schnittstellen und Beispiele vorgestellt, die auch bei der Entwicklung des Prototyps für diese Diplomarbeit Anwendung finden werden: Personenerkennung, Skelett-Tracking, Hand- und Fingererkennung.

Personenerkennung

Um eine hohe Informationsrate für Programmierer zu ermöglichen, werden bereits im Kinect-Sensor selbst die aufgezeichneten Videostreams vorverarbeitet. Hardwareseitig implementierte Algorithmen berechnen ein Tiefenbild, wie in Abbildung 15 links dargestellt. Die verschiedenen Grauwerte eines Pixels beschreiben, wie weit ein Objekt an dieser Stelle im Raum erkannt wurde. Je heller der Pixel dargestellt wird, desto näher befindet sich das erkannte Objekt am Sensor.

Mithilfe dieser Tiefeninformationen wird ebenfalls im Sensor eine Personenerkennung durchgeführt. Bis zu sechs Personen können in einem Bild erkannt werden (Abbildung 15, rechts).

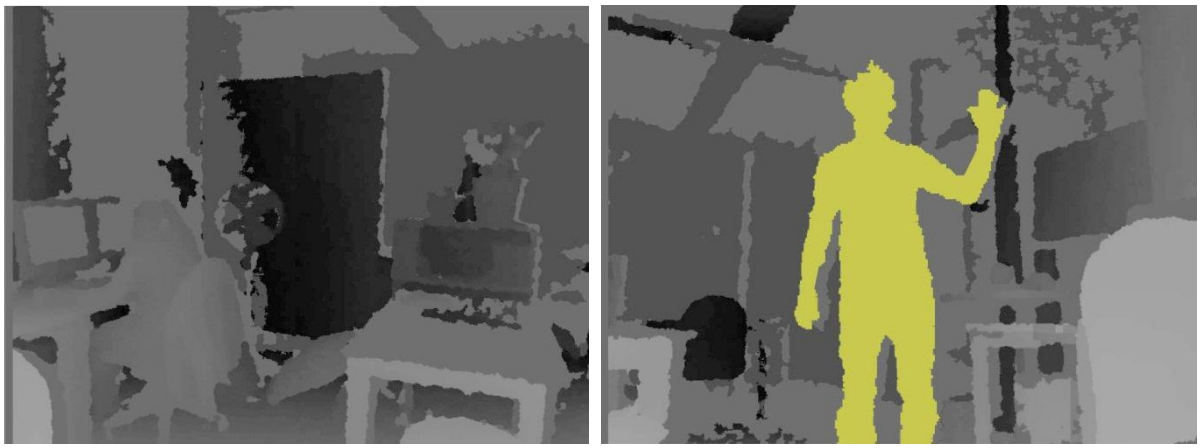


Abbildung 15: Tiefenbild einer Szene, aufgenommen mit dem Kinect-Sensor (links). Je heller ein Bereich dargestellt wird, desto näher befindet sich dieser Bereich am Sensor. Rechts wurde eine Person in der Szene erkannt und markiert.

Skelett-Tracking

Um Körperbewegungen der vom Kinect-Sensor erkannten Personen - beispielsweise für eine Gestenerkennung - nachvollziehen zu können, ist ebenfalls die Erkennung eines minimalistischen Skeletts implementiert. Zu jeder vom Sensor erkannten Person wird die Position aller wichtigen Gelenke bestimmt und daraus ein virtuelles Skelett erstellt. In diesem Skelett sind von folgenden Gelenken und markanten Punkten die Position und Distanz zur Kinect enthalten: Kopf, Schultern, Ellenbogen, Handgelenke, Handspitzen, Hüftgelenke, Knie, Knöchel und Fußspitzen. Mit diesen Informationen lassen sich beispielsweise Posen von Personen erkenn und Bewegungen verfolgen, aus denen Gesten abgeleitet werden können.

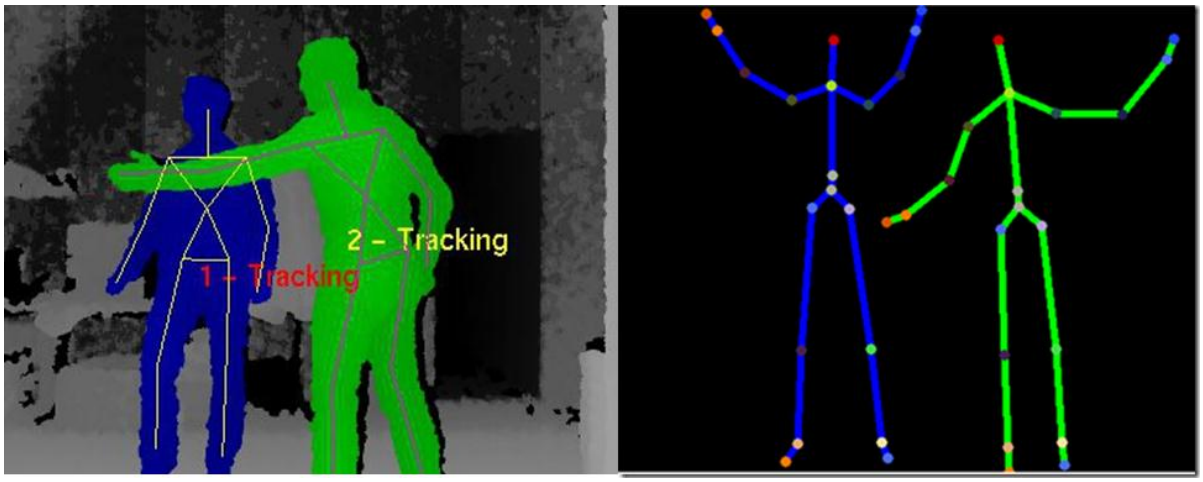


Abbildung 16: Durch den Kinect-Sensor erkannte Skelette von Personen. Links: Darstellung im OpenNI-Framework, rechts: mit dem Microsoft Kinect for Windows SDK.

Hand- und Fingererkennung

Mit dem Microsoft Kinect-Sensor ist auch eine Hand- und Fingererkennung möglich. Dieses Feature ist nicht in der Hardware des Sensors implementiert, sondern softwareseitig realisiert werden. Informationen zur Handerkennung lassen sich aus den Skelettdaten der erkannten Personen ableiten. Im Skelett werden die Punkte der Handgelenke und Fingerspitzen einer Person gespeichert.

Die Finger einer verfolgen Person werden nicht mit dem Skelett erkannt. Es sind Bibliotheken frei verfügbar, die aus den Datenstreams des Kinect-Sensors eine Fingererkennung durchführen können. Eine Bibliothek zur Hand- und Fingererkennung ist „CandescentNUI“, welche auf der OpenSource-Plattform codeplex.com verfügbar ist (Stegmueller, 2012). Mit dieser Bibliothek können aus dem Tiefenbild des Kinect-Sensors Konturen von Händen gefunden und Finger an einer Hand erkannt werden. Zusätzlich zur Position einer erkannten Hand können die Anzahl der ausgestreckten Finger, sowie deren Richtungsvektoren abgerufen werden.

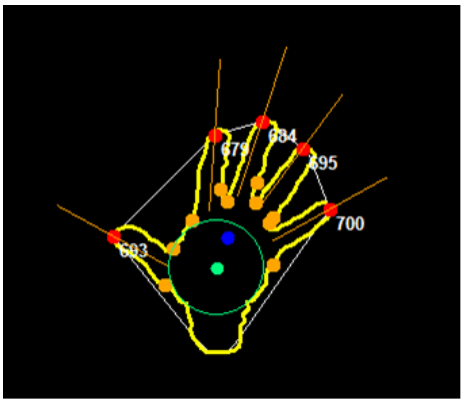


Abbildung 17: Hand- und Fingererkennung mit dem Kinect-Sensor. Es werden die Kontur der Hand (gelb), der Mittelpunkt der Handfläche (grün), die Fingerspitzen (rot) und die Richtungsvektoren der Finger dargestellt (Stegmueller, 2012).

2.4.4 Anwendungen des Kinect-Sensors

Dieser Abschnitt stellt bestehende Anwendungen des Microsoft Kinect Sensors vor. Zunächst werden Anwendungen in Kombination mit der XBOX 360 vorgestellt. Anschließend werden Anwendungen für die Kinect for Windows präsentiert, die den Kinect-Sensor außerhalb der Spielewelt einsetzen.

Anwendungen für die XBOX 360

Der Kinect-Sensor wurde für die Verwendung mit der Spielekonsole XBOX 360 entwickelt. Bisher sind über 80 Videospiele für die XBOX Kinect erschienen (Kinect Games, 2012). Dabei sind zwei verschiedene Spieltypen zu erkennen:

- Spiele, die den Kinect-Sensor nutzen, um die Bewegungen des Spielers aufzuzeichnen und auf den Avatar im Spiel zu übertragen.
- Spiele, die mithilfe des Kinect-Sensors Bewegungen des Spielers analysieren und überprüfen, ob sie bestimmten Vorgaben entsprechen

Im ersten Spieletyp steuert der Spieler mit seinen Körperbewegungen einen Charakter durch eine virtuelle Welt und lässt ihn Aktionen durchführen. Typische Bewegungen sind: laufen, springen, rennen, sich ducken oder schwimmen. Ein Beispiel für diesen Spieletyp ist „Kinect Sports“, in dem der Spieler olympische Disziplinen wie Diskuswurf, 100-Meter-Sprint, Hürdenlauf und Weitsprung absolvieren muss (siehe Abbildung 18).

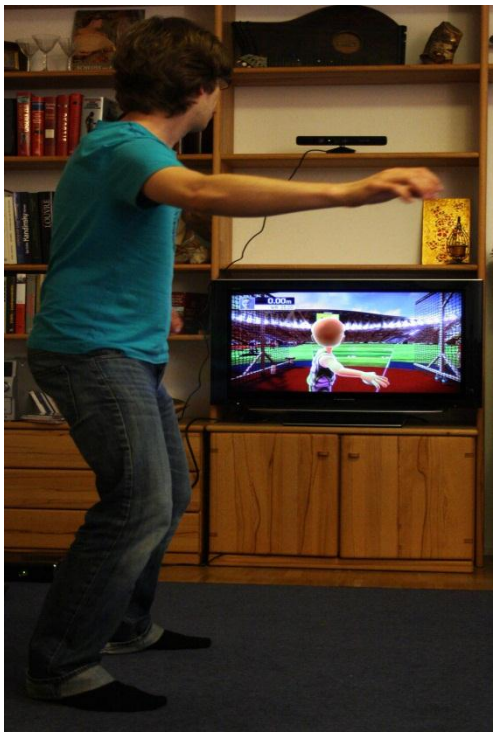


Abbildung 18: Kinect Sports - die Bewegungen des Spielers beim Diskuswurf werden auf den Avatar in der virtuellen Sportarena übertragen.

Der zweite Spieletyp lässt den Spieler Bewegungen imitieren, die von einer virtuellen Person demonstriert werden. Je genauer die Bewegungen und das Timing mit den Vorgaben des Bewegungsmodells übereinstimmen, desto mehr Punkte werden dem Spieler gutgeschrieben. Dieses Konzept verwendet beispielsweise das Spiel „Dance Central“. Ein virtueller Tanzlehrer führt dem Spieler Bewegungen zu aktueller Musik vor, die nachgetanzt werden müssen (siehe Abbildung 19).

Abbildung 19: Dance Central – der Spieler imitiert die Tanzbewegungen eines virtuellen Tanzlehrers.

Anwendungen für die Kinect for Windwos

Es gibt bereits Programme, die den Kinect-Sensor außerhalb der Spielewelt einsetzen, auch wenn diese noch nicht kommerziell vertrieben werden. Ein Beispiel ist eine Anwendung, die in Operationssälen von Krankenhäusern eingesetzt werden kann. Um sich im Operationsgebiet zurechtzufinden und beispielsweise tumoröse Gewebe zu identifizieren, kann ein Chirurg auf zuvor angefertigte Röntgen-, CT- oder MRT-Aufnahmen zurückgreifen, die ein dreidimensionales Abbild des Patienten darstellen. Seither musste ein Chirurg Assistenzpersonal beauftragen die Darstellung der Aufnahmen auf dem Bildschirm seinen Wünschen entsprechend anzupassen. Würde er selbst eine Tastatur oder Maus bedienen, ginge die Sterilität verloren und die Operation würde durch neue Hygienemaßnahmen unnötig in die Länge gezogen. Um dem Operateur selbst die Möglichkeit zu geben, mit den Visualisierungen am Computer zu interagieren ohne unsteril zu werden, wurde eine Anwendung mit der Microsoft Kinect entwickelt (Ruppert, Reis, Amorim, de Moraes, & da Silva, 2012).

2.5 Powerwall

Als „Powerwall“ wird ein großflächiges, hochauflösendes Display bezeichnet. Im Englischen wird der Begriff „Large High-Resolution Display“ verwendet. In den folgenden Abschnitten werden verschiedene Hardwarekonfigurationen für Powerwalls, Forschungsfelder auf dem Gebiet der hochauflösenden Displays und einige Anwendungsszenarien vorgestellt (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006). Anschließend werden einige Herausforderungen bei Interaktionskonzepten mit Powerwalls präsentiert und abschließend die Powerwalls beschrieben, die zur Durchführung dieser Diplomarbeit am Institut für Visualisierung und Interaktive Systeme zur Verfügung stehen.

2.5.1 Hardwarekonfigurationen

Die Hardware eines Standard-PCs ist in den meisten Fällen nicht für den Betrieb eines großflächigen und hochauflösenden Displays aus. Mittlerweile wurden unterschiedliche Hardwarekonfigurationen für hochauflösende Displays entwickelt. Im Folgenden werden die Techniken „Multi-Monitor Desktop“, „Tiled LCD Panels“ und „Projector Arrays“ dargestellt.

Multi-Monitor Desktop

Heutige Betriebssysteme erlauben das Betreiben von mindestens einem weiteren Monitor neben dem Standardbildschirm. Dabei wird beispielsweise der Windows-Desktop um die Auflösung des zweiten Monitors in einer bestimmten Richtung erweitert. Bei handelsüblichen Grafikkarten gehört die Unterstützung dieser Funktionalität bereits zum Standard (siehe Abbildung 20). Da solche Konfigurationen ohne spezielles Fachwissen aufgebaut werden können, finden sie vor allem an Desktop-PCs im privaten Bereich und im Büroalltag Anwendung (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006). Spezielle Grafikkarten und Technologien wie EyeFinity des Grafikkartenherstellers ATI ermöglichen mittlerweile das Betreiben von bis zu sechs Bildschirmen (AMD, 2012).

Abbildung 20: Mehrere Monitore an einem Bürocomputer erweitern den Arbeitsbereich.

Tiled LCD Panels

Ähnlich der Technik des Multi-Monitor Desktops ist das Prinzip der Tiled LCD Panels. Hier bildet ein Set von mehreren LCD-Displays in einer Matrixanordnung die Powerwall. Die Displays können dabei flach horizontal als Tisch oder als vertikal als Wand angeordnet werden. Ebenso sind gekrümmte Anordnungen möglich. Im Electronic Visualization Labortory der University of Illinios at Chicago werden sowohl eine Tisch-Powerwall aus 15 Tiles (Krumbholz, Leigh, Johnson, Renambot, & Kooima) und ein flaches Wand-Display bestehend aus 55 LCD Displays (Renambot, Jeong, Jagodic, Johnson, & Leigh) entwickelt. Ebenfalls wird dort an einem Display aus mehreren LCD Displays gearbeitet, das einen 360°-Rundumblick ermöglichen soll (Johnson, 2010).

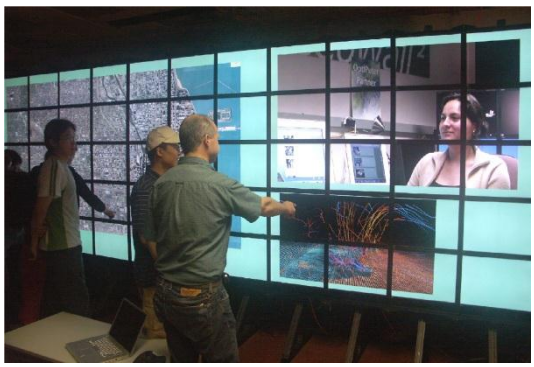


Abbildung 21: Tiled LCD Displays. Links der MultiTouch-Tabletop "lambdaTable", rechts die Powerwall "lambdaVision" (Renambot).

Die Verwendung von LCD-Displays bietet gegenüber der Verwendung von Projektoren, wie sie im nächsten Abschnitt beschrieben werden, diverse Vorteile: Großflächige LCD-Displays sind vergleichsweise günstig zu beschaffen und benötigen weniger Platz, da auf einen Projektionsabstand

verzichtet werden kann. Darüber hinaus sind LCD-Displays einfacher gegeneinander auszurichten und in der Fardarstellung abzugleichen.
Diesen Vorteilen ist ein gravierender Nachteil entgegenzusetzen: bei nebeneinander angeordneten LCD-Displays gehen die einzelnen Darstellungsflächen nicht nahtlos ineinander über. Die Rahmen der Displays unterbrechen das dargestellte Bild. Dies ist vor allem bei Textdarstellungen ungünstig (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006).

Projector Arrays

Derartige Brüche zwischen den Teilbildern, aus denen eine Powerwall zusammengesetzt ist, treten bei der Verwendung von Projektoren nicht auf. Bei einer Konfiguration mit Projektoren können die einzelnen Geräte so positioniert werden, dass sich die Projektionen teilweise überlappen. So entsteht ein weicher Übergang zwischen den Einzelbildern. Als Beispiel für Projector Arrays führt (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006) den VisBlock an, eine kompakte Konstruktion, die eine beliebige Anordnung von mehreren Projektoren erlaubt.

Abbildung 22: VisBlock - Powerwall aus 20 aneinander gefügte Bildprojektionen (VisWall, 2012).

Anders als bei den LCD-Displays sind die Bilddarstellungen nicht an die Abmessungen der Geräte gebunden. Kleine Projektoren können bei einer größeren Distanz zur Projektionsfläche ein größeres Bild darstellen. Ebenso kann das Bild auf beliebige Oberflächen projiziert werden. Wie im Vergleich mit LCD-Displays bereits beschrieben, benötigen Projector Arrays allerdings vergleichsweise viel Platz. Gleichzeitig sind Anschaffung und Unterhaltung von Projektoren sind deutlich teurer als bei LCD-Displays (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006).

2.5.2 Anwendungsgebiete

Hochauflösende und großflächige Displays haben vielfältige Anwendungsmöglichkeiten. In (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006) wurden diese unter anderem zu folgenden Anwendungsgebieten zusammengefasst:

- **Command and Control**
  In vielen Kommandozentralen und Koordinationsstellen von beispielsweise Militär, Flugsicherheit oder auch Telekommunikation sind heutzutage Powerwalls installiert. Die amerikanische Luftwaffe entwickelte die interaktive DataWall, um Situationsanalysen und Entscheidungsfindung in Gefechtssituationen zu unterstützen.

- **Vehicle Design**
  Bei der Konstruktion von Fahrzeugen können Designer auf die Besonderheit von Powerwalls zurückgreifen, Modelle in originalgetreuem Maßstab darzustellen und im Team gleichzeitig an einer Darstellung arbeiten. Gleichzeitig können in VR-Laboren Fahreigenschaften simuliert und getestet werden.

- **Scientific Visualization**
  Daten in verschiedensten Größendarstellungen betrachten und fast beliebig viele Daten gleichzeitig darstellen zu können, sind Argumente Powerwalls zur Darstellung wissenschaftlicher zu verwenden. Bespiele hierfür sind Simulationen einer Supernova oder übergroße Darstellungen von Molekülstrukturen.

- **Collaboration**
  Als Präsentations- und Diskussionsfläche können hochauflösende Displays in Meetings oder Gruppenarbeiten Anwendung finden. An der Universität Konstanz wurde hierzu beispielsweise eine Powerwall konzipiert, auf der während einer Präsentation Objekte mit einem Laser-Pointer manipuliert werden können (König, Bieg, Schmidt, & Reiterer, 2007).

- **Immersive Applications**
  Eine CAVE („CAVE Automatic Virtual Environment“) besteht aus mindestens vier Wänden, auf die Bilder einer virtuellen Realität projiziert werden. Zusätzlich können ebenfalls der Fußboden und die Decke beleuchtet werden. Ein Benutzer kann sich in dieser virtuellen Umgebung bewegen oder die Position der virtuellen Kamera mit einem Joystick steuern. In einer solchen Umgebung wurden beispielsweise Trainingsszenarien für Feueralarme entwickelt.

- **Public Information Dispslays**
  Durch die immer günstiger werdende Hardware verdrängen große digitale Displays zunehmend gedruckte Werbung an öffentlichen Plätzen. Durch Projektoren können beliebig geformte Objekte zu Projektionsflächen werden.

2.5.3 Interaktion mit Powerwalls

Benutzungsschnittstellen zur Interaktion mit Programmen, die auf handelsüblichen Bildschirmen dargestellt werden, entwickeln Programmierer seit vielen Jahren. Im Gegensatz dazu gibt es kaum Erfahrung mit großen hochauflösenden Displays. In (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006) konnten bereits einige Herausforderungen bei der Interkation mit großen Displays identifiziert werden:

- **Reaching distant objects**
  Je näher ein Anwender vor einem großflächigen Display steht, desto schwieriger ist für ihn alle dargestellten Objekte zu erfassen und mit ihnen zu interagieren. Diese Schwierigkeit wächst mit der Größe des Displays. Beispielsweise ist das Verschieben eines Objekts von der rechten Seite des Displays an den linken Rand eine schwere Aufgabe, wenn das Display entsprechend groß ist.

- **Tracking the cursor**
  Wird eine Anwendung auf einer Powerwall mit Maus und Tastatur gesteuert, verwenden Benutzer bei größeren Displays immer höhere Mauszeigergeschwindigkeiten. Dies erschwert die Verfolgung des Cursors. Zusätzlich ist das Auffinden eines Mauszeigers auf einem entsprechend großen Display schwer.

- **Crossing bezels**
  Wie in Kapitel 2.5.1 beschrieben, können zur Konstruktion einer Powerwall mehrere LCD-Displays nebeneinander angeordnet werden. In dieser Konstruktion behindern die Rahmen, welche die einzelnen Displays begrenzen, die Interaktion mit der Powerwall. Beispielsweise gibt es Unterschiede zwischen der erwarteten Mausbewegung über eine Displaygrenze und der Anzeige des Cursors, denn unterhalb der Displayrahmen existiert keine Anzeigefläche.

- **Transitioning between interactions**
  Interaktionen mit großen Displays können in zwei Paradigmen unterteilt werden: Aufgaben, die Arbeiten mit Details der dargestellten Informationen beinhalten, werden in kurzer Distanz zum Display durchgeführt. Andere Aufgaben, beispielsweise Übersichtsaufgaben, werden eher in großer Distanz ausgeführt. Es bedarf bei der Interkation mit Powerwalls also eines nahtlosen Übergangs zwischen diesen beiden Aufgabenarten.

Darüber hinaus führt (Czerwinski, Robertson, Meyers, Smith, Robbins, & Tan, 2006) zwei weitere Probleme auf. Diese treten vor allem dann auf, wenn die Interaktion mit Fenstern, wie sie auf Standard-Bildschirmen gängig ist, auf ein großflächiges Display übertragen wird. Sie lassen sich aber auch auf die fensterlose Darstellung von Daten und Objekten übertragen.

- **Window management problems**
  Großflächige Displays erschweren die Positionierung von neu erstellten Fenstern: Es ist schwer zu bestimmen, wo neue Fenster erscheinen und beispielsweise Dialogboxen dargestellt werden sollen.

- **Task management problems**
  Auf einem großen Display können mehr Fenster gleichzeitig dargestellt werden als auf dem Standardbildschirm eines Desktop-PCs. Um den Überblick zu behalten, bedarf es neuer Mechanismen, welche eine Zuordnung von Fenstern zu Aufgaben und eine Aufgabenverwaltung ermöglichen.

Eine weitere Herausforderung bilden große Displays, wenn sie von einer Gruppe von Anwendern verwendet werden. Je nachdem, wie das große Display in die Gruppenarbeit integriert wird, können ihm unterschiedliche Bedeutungen zuteilwerden. Es kann zu einem öffentlichen Raum für alle Beteiligten, aber auch nur zu halböffentlichen Ablageflächen für Gruppenmitglieder werden. Dies kann sowohl die Interaktion mit dem Display als auch die angezeigten Daten beeinflussen (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006).

2.5.4 Forschungsfelder

Basierend auf der Analyse der technischen Konfigurationen von Powerwalls, der Sammlung von Anwendungsgebieten und den Herausforderungen in der Interaktion, lassen sich laut (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006) folgende Top-Ten Forschungsfelder ableiten:

1. **Truly seamless tiled displays**
   Das Kalibrieren mehrerer Projektoren und Displays ist eine langwierige Aufgabe beim Aufbau einer Powerwall. Unterschiedlich kräftige Farben und Helligkeitsunterschiede können den Eindruck eines durchgängigen Bildes zerstören.

2. **Stereoscopic large high-resolution displays**
   Mit mehreren Projektoren können für jedes Auge unterschiedliche Bilder projiziert werden um einen dreidimensionalen Eindruck beim Betrachter zu erzeugen. Eine Herausforderung ist, mehreren Personen, die an unterschiedlichen Positionen im Raum stehen, einen idealen 3D-Effekt zu präsentieren.

3. **Easily reconfigurable large high-resolution displays**
   Da das Kalibrieren einer Powerwall viel Zeit in Anspruch nimmt, ist das Umgestalten eines beispielsweise flachen Displays zu einer gekrümmten Displayfläche zu aufwändig. Zukünftige hochauflösende Displays sollten einfach umzugestalten oder sogar transportabel sein.

4. **High-performance cluster rendering**
   Großflächige und hochauflösende Displays benötigen spezielle Hardware und Rechnerarchitekturen um große Grafiken in Echtzeit anzeigen zu können.

5. **Scalability**
   Die Mehrzahl zusammengesetzter Displays beschränkt sich auf weniger als zwanzig Kacheln. Zukünftig sollen deutlich mehr Kacheln, wie bei der LambaVision Powerwall, zum Standard werden.

6. **Design and evaluate large high-resolution display groupware**
   Großflächige Displays sind prädestiniert zur Verwendung bei Teamaufgaben. Im Gegensatz zu etablierten Groupware-Anwendungen für Desktop-PCs bieten Powerwalls zusätzliche Herausforderungen.

7. **Effective interaction techniques**
   Maus und Tastatur eignen sich nur bedingt zur Steuerung einer Powerwall. Es wird bereits an anderen Techniken der Interaktion geforscht. Einige Beispiele sind: Gesteninterkation, Laser-Pointer oder Spracherkennung.

8. **Perceptually valid ways of presenting information on the large displays**
   Große Anzeigeflächen weisen Besonderheiten in Bezug auf die visuelle Wahrnehmung durch den Betrachter auf: in großer Distanz zu einer Powerwall können kaum Details der dargestellten Informationen wahrgenommen werden. Befindet man sich nah genug an der Projektionsfläche, können Objekte „am anderen Ende“ der Powerwall nicht mehr betrachtet werden.

9. **Empirical evidence for the benefits of large high-resolution displays**
   Eine evidenzbasierte Grundlage für die Gestaltung von Anwendungen auf großflächigen Displays würde die Entwicklung derartiger Anwendungen vorantreiben. Hersteller gingen damit ein geringeres Risiko bei Neuentwicklungen ein.

10. **Integrating large high-resolution displays into a seamless computing environment**
    Eine Herausforderung besteht darin, verschiedene Geräte in eine gemeinsame Arbeitsumgebung zu integrieren. Wie bereits beschrieben können Poweralls Gruppenaufgaben eingesetzt werden. Nun könnten persönliche Geräte der Gruppenteilnehmer in eine gemeinsame Umgebung integriert werden.

2.5.5 Powerwalls am Institut für Visualisierung und Interaktive Systeme

Zur Entwicklung des Prototyps für diese Arbeit stehen am Informatikinstitut der Universität Stuttgart insgesamt drei verschiedene Powerwalls zur Verfügung:

- Im VR-Labor des Informatikgebäudes ist eine vergleichsweise kleine Powerwall installiert. Sie wird von einem Rechner mit 3D-Grafikkarte gesteuert und verfügt über zweifache HD-Auflösung (3.840 x 1080 Pixel). Das Bild wird ohne Überlappung von zwei HD-Beamern auf die Projektionsfläche gestrahlt. Für den 3D-Effekt projiziert je ein weiterer HD-Beamer pro Auge ein zweites Bild. Die beiden Bilder werden durch Filter unterschiedlich polarisiert. Ein Betrachter benötigt eine Polarisationsbrille um für jedes Auge eines der beiden Bilder auszublenden und so den 3D-Effekt wahrnehmen zu können.

Abbildung 23: Powerwall im VR-Labor des Informatikgebäudes der Universität Stuttgart.

- Ebenfalls mit dieser Polarisationstechnik ist die zweite Powerwall des Informatikinstituts ausgestattet. Sie befindet sich im Hörsaal V38.02. In diesem Fall sorgen sechs HD-Beamer mit überlappenden Projektionen für eine übergangslose dreidimensionale Darstellung.

- Die Powerwall im VISUS-Gebäude stellt auf einer Projektionsfläche von 5,97 x 2,25 Meter werden bei einer Auflösung von 10.800 x 4.096 Pixel über 88 Millionen Bildpunkte dar. Ein einzelner Pixel wird dort mit einer Größe von etwa 0,56 mm² dargestellt. Zehn 4K-Projektoren, wie sie in digitalen Kinos üblich sind, projizieren das Bild mit einer Auflösung von je 4.096 x 2.400 hochkant auf eine Verbundglasscheibe. Damit Visualisierungen mit einer Bildrate von mindestens 30 Bildern pro Sekunde erreicht werden, werden die zehn Steuercomputer der Powerwall durch einen Rechnercluster mit 64 Render-Knoten unterstützt.
  Diese Powerwall ist, wie die beiden anderen, fähig dreidimensionale Visualisierungen

darzustellen. Ähnlich zu Rot-Grün-Brillen werden für jedes Auge kleine Farbbereiche blockiert und somit der 3D-Effekt erzielt. Bei dieser Variante der passiven 3D-Technologie kann der Betrachter jedoch den Kopf beliebig neigen ohne, dass der 3D-Effekt verloren geht (VISUS, 2012).

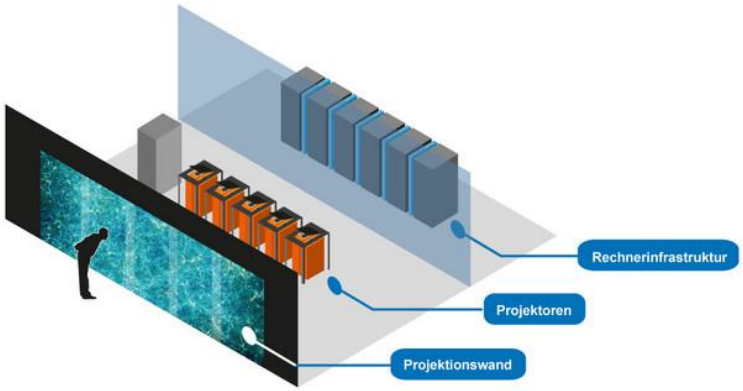


Abbildung 24: Schematische Darstellung der Powerwall im VISUS-Gebäude. Zehn Beamer projizieren ein Bild auf eine 5,97 x 2,25 Meter große Verbundglasscheibe (VISUS, 2012).

2.6 Natural User Interfaces

Die meisten Anwendungen, die heutzutage auf Computern ausgeführt werden, bedienen sich neben der Desktop-Metapher des WIMP-Konzepts. WIMP steht hierbei für „Windows, Icons, Menu, Pointing device“ (Jacob, et al., 2008). Hierbei werden beispielsweise mit Hilfe einer Maus, die über die Tischplatte geschoben wird, Fenster auf dem Computer-Bildschirm hin und her bewegt. Diese indirekte Interaktionsform entspricht nicht dem, was Benutzer aus der realen Welt gewohnt sind. Neue Geräte wie Touchscreens, Powerwalls und die Kinect bringen neue und direkte Interaktionsmöglichkeiten. Ohne Geräte, die zwischen der Hand des Benutzers und dem Computerdesktop vermitteln, können angezeigte Objekte direkt angetippt oder ausgewählt werden. Bewegt man die Hand, bewegen sich ausgewählte Objekte mit – genau, wie es auch in der „echten Welt“ der Fall wäre. Daher werden diese Benutzungsschnittstellen auch „Natural User Interfaces“ – kurz NUI – genannt (Wigdor & Wixon, 2011).

(Wigdor & Wixon, 2011) sehen die Imitation der Realität aber nur als einen kleinen Teil eines Natural User Interfaces an. Viel wichtiger bei einem NUI ist, dass es sich für jeden Benutzer „natürlich“ anfühlt. Sowohl Novizen, die das System zum ersten Mal verwenden, als auch Experten, die das System täglich benutzen, müssen sich problemlos mit der Schnittstelle auseinandersetzen können. Die Schnittstelle bedient die individuellen Bedürfnisse des Anwenders. Das System unterstützt einen Anfänger bei seinen zögerlichen Versuchen mit dem System umzugehen und schafft einen leichten Einstieg. Dabei erreicht dieser Anwender seine Ziele mit Zufriedenstellung. Gleichzeitig werden die Ansprüche eines Profis befriedigt, der das System mit hoher Effizienz und Geschwindigkeit bedienen kann. Die Entwicklung eines Novizen zum Vollprofi soll fließend und einfach sein. Der Anwender hat Spaß an der Interaktion mit dem System, egal auf welcher Höhe der Lernkurve er sich befindet.
Das Höchste Ziel eines NUI ist die Verschmelzung mit der Realität: sobald der Benutzer nicht mehr zwischen Imitation der Realität und der Kreation aus der virtuellen Welt unterscheiden kann, ist dieses Ziel erreicht.

Natural User Interfaces haben dabei oft sozialen Charakter. Computer mit Tastatur und Maus sind als Einzelarbeitsplätze gedacht. Touchscreens und Powerwalls hingegen erlauben Kollaboration in verschiedenen Aufgabenszenarios:

- *Gemeinsame Aufgabe* („highly coupled tasks“)
  Mehrere Anwender arbeiten gleichzeitig an derselben Aufgabe. Das Ergebnis hängt von ihrer Zusammenarbeit ab.

- *Aufgabenteilung* („lightly coupled tasks“)
  Mehrere Anwender teilen sich eine Aufgabe. Ihre Teilergebnisse werden zur Lösung der Aufgabe zusammengefügt.

- *Getrennte Aufgaben* („uncoupled tasks“)
  Mehrere Anwender arbeiten an unterschiedlichen Aufgaben, die nicht zusammenhängen

Mit dem Spiel „Paint Party“ hat Microsoft das Konzept seines „Natural User Interfaces“ erstmals auf der E3-Konferenz 2009 vorgestellt. Hier wurde die Kinect der Öffentlichkeit zum ersten Mal unter dem Codenamen „Project Natal“ präsentiert. „Paint Party“ verwandelt das Wohnzimmer des Spielers in ein Action-Painting Studio: Mit beiden Händen kann eimerweise Farbe auf einer überlebensgroßen digitalen Leinwand verteilet werden. Ähnlich wie bei einer Wasserschlacht – oder beim richtigen Actionpainting. Feinere Farbspritzer und -spuren lassen sich mit dem Wink einer Hand auf die Leinwand bringen. Mit gesprochenen Kommandos wie „Hellblau“ oder „Braun“ kann der Künstler ohne lästige Menüs aus seiner Farbpalette wählen. Es können sogar Schattenbilder, die vom Künstler oder mithilfe seiner Assistenten dargestellt werden, als Schablonen verwendet und in das fertige Bild integriert werden (Microsoft, 2009).

Abbildung 25: Beispiel eines Natural User Interface. Mit Körperbewegungen kann der Benutzer Farbe auf die Leinwand bringen (Microsoft, 2009).

„Natural User Interface“ ist aber nicht gleichbedeutend mit „ganz ohne Controller“. Bereits 2006 landete der Spielekonsolenhersteller Nintendo mit der Wii einen großen Erfolg. Ein Controller, der mit entsprechenden Sensoren ausgerüstet ist, erkennt Drehung, Haltung und Beschleunigung des Arms eines Spielers. Diese Bewegungen werden auf das jeweilige Spiel übertragen. Im Gegensatz zur Kinect hat der Spieler hier etwas in der Hand, mit dem sich Gesten ausführen lassen, wie sie auch in der Realität Anwendung finden. Jeder hat schon einmal Tennis gespielt oder zumindest im Fernsehen ein Tennisspiel verfolgt. Dieses Wissen reicht aus, um den Controller der Wii wie einen Tennisschläger halten und bewegen zu können. Schon kann man ohne großen Lernaufwand den Spielspaß genießen. Um das Spiel noch realitätsnaher zu gestalten, gibt es für diverse Spiele der Wii

Zusatzteile, die an den Controller angebaut werden können. Man hält einen richtigen Tennischläger in der Hand oder bewegt ein rundes Lenkrad beim Steuern eines Fahrsimulators.

Abbildung 26: Der Nintendo Wii Controller vermittelt - eingebaut in ein Lenkrad - ein natürliches Fahrgefühl.

Auch Touchscreens werden zu den „Natrual User Interfaces“ gezählt. Jedes Smartphone besitzt ein Touch-Display, auf dem man durch Tippen Tasten drücken oder mit Wisch-Gesten durch Menus scrollen kann. In größeren Ausführungen stehen diese Interaktionsmöglichkeiten auf Tablets wie dem iPad zur Verfügung. Es gibt auch Bildschirme für den Bürocomputer, die eine Touch-Eingabe ermöglichen. Neue Technologien ermöglichen sogar, dass ganze Tischoberflächen als Touchscreen verwendet werden können. Ein Beispiel hierfür ist Microsofts Surface, der bereits in Hotels oder Banken Anwendung findet (Microsoft, 2012). Neben der Interaktion mit einem oder mehreren Fingern erkennt der Surface auch speziell markierte Objekte, die auf der Tischplatte abgelegt werden. Dies erlaubt eine der direktesten Formen der Interaktion: Objekte aus der realen Welt werden sowohl vom Menschen als auch vom Computer „verstanden“.

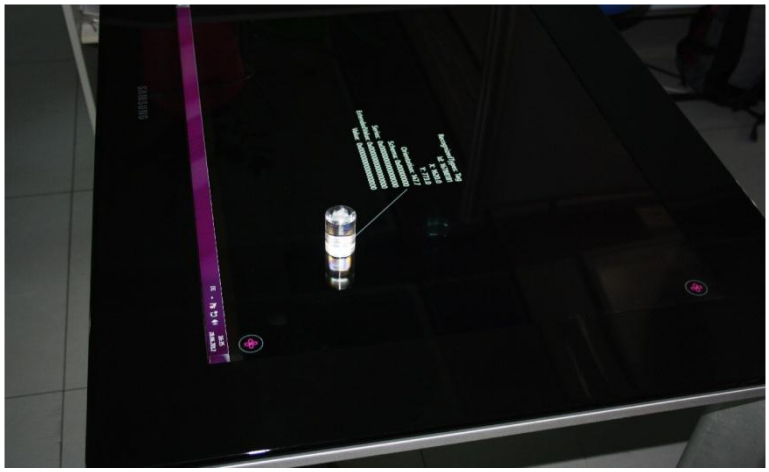


Abbildung 27: Ein Plexiglaszylinder wird vom Microsoft Surface als Interaktionsobjekt erkannt.

2.7 Verwandte Arbeiten

Dieser Abschnitt stellt Arbeiten vor, die thematisch mit dem Aufgabengebiet dieser Diploamrbeit verwandt sind. Zum einen werden Anwendungsbeispiele von großen, hochauflösenden Displays und Interaktionssysteme mit Powerwalls vorgestellt. Zum anderen werden Frameworks und Systeme zur multimodalen Interaktion beschrieben.

Powerwall-Anwendungen

Seither wurden große, hochauflösende Displays laut (Randell, et al., 2012) vor allem zu Präsentationszwecken und in den Geowissenschaften eingesetzt. In ihrem Projekt „Leeds Virtual

Microscope“ setzen (Randell, et al., 2012) die Powerwall-Technologie in der Lehre der Humanmedizin ein. In Seminaren können Medizinstudenten als Gruppe Gewebeproben auf einem Tiled Display bestehend aus 28 Segmenten betrachten und diskutieren, statt sich mehrere Mikroskope teilen zu müssen . Die Steuerung des dargestellten Bildausschnitts erfolgt hierbei mit einem Logitech Gamepad.

Die Firma ABB setzt die Powerwall-Technologie in Kombination mit Mobilgeräten bei der Konstruktion und dem Design technischer Komponenten ein (Fallman, Kruzeniski, & Andersson, 2005). Auf einer ca. 3m x 1,80m großen Projektionsfläche können Konstruktionszeichnungen und Blaupausen der Produkte der Firma ABB dargestellt, präsentiert und mit Pen-Tracking manipuliert werden. Darüber hinaus können in einem „aktiven Bereich“ vor der Powerwall Daten auf mobile Geräte übertragen und beispielsweise von Technikern mit in die Produktionshalle genommen werden.

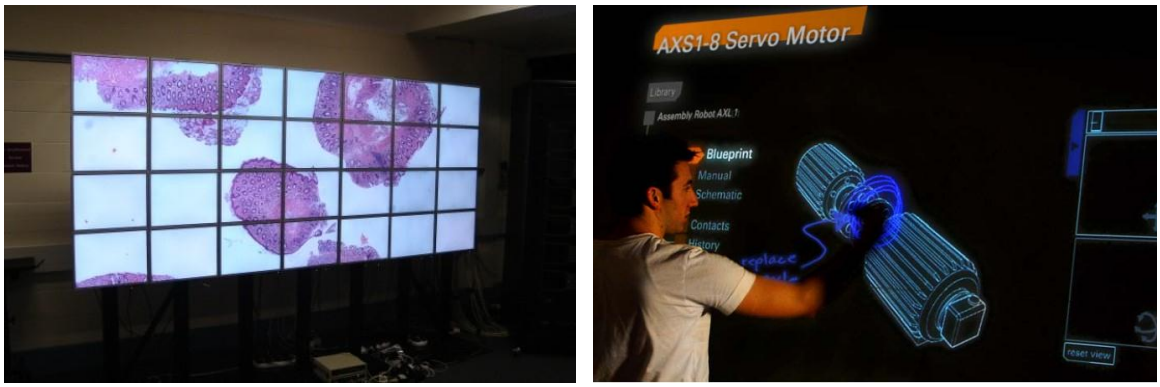


Abbildung 28: Einsatz einer Powerwall im Projekt "Leeds Virtual Microscope" (links). Rechts interagiert ein Mitarbeiter mit einer Konstruktionszeichnung auf der ABB-Powerwall.

Microsoft Research kombiniert bereits Powerwalls mit Gestenerkennung zur Unterstützung von Besprechungen zwischen Softwareentwicklern (Bragdon, DeLine, Hinckley, & Morris, 2011). Mithilfe von Mobilgeräten und Zeigegesten können Entwickler dargestellte Elemente auf der Powerwall markieren und manipulieren. Dies erleichtert den Diskussionsablauf, der nicht mehr durch Anweisungen aus dem Plenum, bestimmte Änderungen in der Präsentation vorzunehmen, unterbrochen werden muss.

Am Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung wurde ein SmartControlRoom entwickelt – eine multimodale Interaktionsumgebung mit Powerwall und interaktivem Tischcomputer für Lagezentren. Mithilfe von Freihandgesten können mehrere Benutzer gleichzeitig mit Karten und Objekten interagieren, die auf der Powerwall oder dem Tabletop-Computer dargestellt werden (Fraunhofer IOSB, 2012).

Diese verwandten Arbeiten fügen sich in die Vision von Mark Weiser ein (Weiser, 1991), in der Powerwalls als integraler Bestandteil in Büros die Interaktion mithilfe verschiedener Eingabemodalitäten ermöglichen. Gleichzeitig verschmelzen die Geräte und die Interaktion mit der Umgebung der Benutzer.

Eine solche multimodale Interaktionsumgebung im Büroalltag beschreiben (Streitz, Prante, Müller-Tomfelde, Tandler, & Magerkurth, 2002) in ihrem Projekt „Roomware“. In dieser Umgebung ist die

Interaktion mit Daten auf einer Powerwall, auf Tischcomputern, mit Mobilgeräten und sogar mit Computern ausgerüsteten Sesseln möglich.

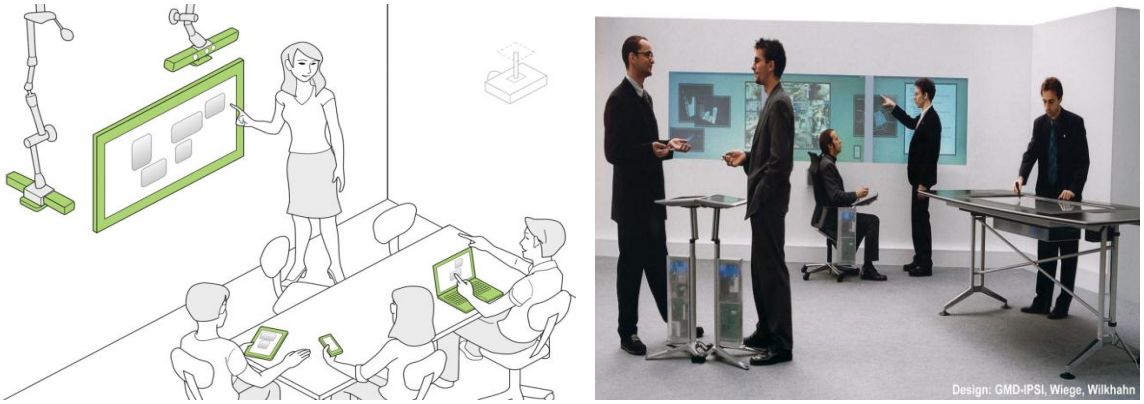


Abbildung 29: Links ist die Anwendung einer Powerwall in Entwickler-Meetings skizziert. Rechts fügt sich die Powerwall in eine multimodale Interaktionsumgebung ein.

Interaktionsframeworks

Um eine multimodale Interaktionsumgebung schaffen zu können, stellen (König, Rädle, & Reiterer, 2009) das Squidy-Framework vor. In diesem Framework können mithilfe von Aktivitätsdiagrammen verschiedene Eingabegeräte mit beliebigen Computern und Ausgabegeräten über ein Netzwerk verbunden werden. Die Verarbeitung der Eingabedaten, wie beispielsweise die Position eines Mauszeigers, kann ebenfalls in diesen Diagrammen beeinflusst werden. Das Squidy-Framework unterstützt keinen Austausch von Daten und Objekten, mit denen in der multimodalen Umgebung interagiert werden soll.

Eine weitere Middleware zur Erstellung von multimodalen Interaktionsumgebungen beschreiben (Gjerlufsen, Klokmose, Eagan, Pillias, & Beaudouin-Lafon, 2011) mit dem Konzept der Shared Substances. Alle Hardware- und Softwarekomponenten sowie Daten, über die ein Computer verfügt, der Teil der multimodalen Interaktionsumgebung ist, werden als Ressourcen betrachtet und können für andere zugänglich gemacht werden. Diese Ressourcenteilung erlaubt allerdings nicht, dass während der Nutzung des Frameworks Komponenten verändert oder aus der Umgebung entfernt werden können.

3 Aufgabenstellung und Lösungsansatz

Ziel dieser Diplomarbeit ist die Entwicklung eines Interaktionskonzeptes zur gestenbasierten Steuerung von Visualisierungen auf einem großflächigen Display. In diesem Kapitel wird die vollständige Aufgabenbeschreibung wiedergegeben und anschließend ein allgemeines Lösungskonzept vorgestellt.

3.1 Aufgabenbeschreibung

Die Aufgabenstellung dieser Diplomarbeit lässt sich in folgenden Punkten zusammenfassen:

- Entwicklung eines geeigneten Interaktionskonzepts zur gestenbasierten Steuerung einer Powerwall
- Einbettung in eine multimodale Interaktionsumgebung
- Entwicklung zweier Szenarien
- Implementierung eines Prototyps
- Evaluation des Prototyps

Im Folgenden werden kurz der Hintergrund und die Vorarbeiten zu dieser Arbeit genannt. Anschließend werden die oben genannten Punkte im Einzelnen genauer vorgestellt.

Hintergrund und Vorarbeiten zu dieser Arbeit

Am Institut für Visualisierung und Interaktive Systeme der Universität Stuttgart werden Eyetracking Studien durchgeführt. Zur Auswertung der Ergebnisse dieser Studien wurden in der Diplomarbeit „Visuelle Analyse von Eye-Tracking-Daten“ (Chen, 2011) neue Visualisierungstechniken erstellt. Bei der Anwendung dieser Visualisierungen zur Auswertung der Studie „Visual Elements“, die am Institut durchgeführt wurde, fiel auf, dass viele Programmfenster gleichzeitig verwendet werden (siehe Abbildung 30) und diese dementsprechend viel Platz auf dem Desktop einnehmen. Zusätzlich erschwert die Verwendung eines einzelnen Desktop-PCs die Zusammenarbeit mehrerer Mitarbeiter bei der Analysearbeit. Es entstand der Wunsch sowohl mehr Platz zur Darstellung der Visualisierungen zu erhalten, als auch mehr Raum für das Analyseteam zur Verfügung zu haben.

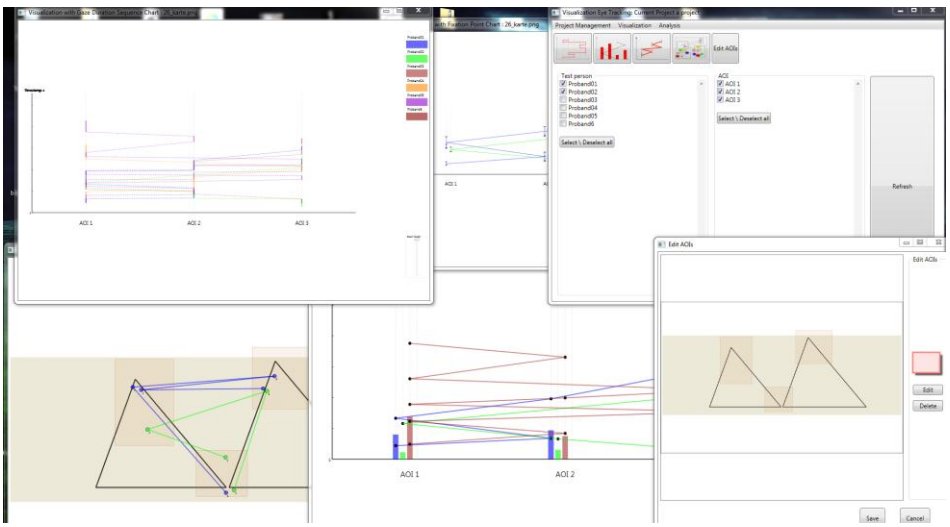


Abbildung 30: Bildschirmausschnitt bei der Analyse der "Visual Elements" Studie.

Wie in Kapitel 2.5 beschrieben, erlauben Powerwalls eine großflächige und hochaufgelöste Visualisierung von Daten. Die große Anzeigefläche ist besonders bei der Datenanalyse im Team von Vorteil. Also sollte eine Möglichkeit gefunden werden die Visualisierungen der Eyetracking-Daten auf einer Powerwall darstellen zu können.

Als die Studienanalyse mit der Powerwall fortgesetzt wurde, stellte sich heraus, dass die Interaktion mit den Visualisierungen durch Maus und Tastatur des Steuercomputers der Powerwall problematisch ist. Weil immer eine Person zum Computer gehen musste, um gewünschte Anpassungen an den Visualisierungen vorzunehmen, wurde der Analysefluss ständig unterbrochen. Es kam die Idee auf die Manipulation der Powerwall-Visualisierungen mithilfe von Gesten durchführen zu können, sodass keine Person die Analysegruppe vor der Powerwall verlassen muss, um Änderungen an den Darstellungen auf der Powerwall vornehmen zu können.

Entwicklung eines Interaktionskonzepts

Im Rahmen dieser Diplomarbeit soll ein Interaktionskonzept entwickelt werden, das die Steuerung von Visualisierungen auf einer Powerwall durch Freihandgestern ermöglicht. Diese Interaktionsschnittstelle soll nach Möglichkeit den Vorgaben eines Natural User Interfaces entsprechen. Zunächst wird das generelle Interaktionskonzept für die Interaktion eines Benutzers beschrieben. Da Powerwalls die Analyse von Daten durch eine Gruppe von Personen unterstützen, ist zusätzlich zu untersuchen, wie das Interaktionskonzept auf mehrere Benutzer gleichzeitig angewendet werden kann.

Einbettung in eine multimodale Interaktionsumgebung

Parallel zu dieser Arbeit wird im Rahmen der Diplomarbeit „Tabletop-Computer-basierte Steuerung für Powerwall-Visualisierungen“ (Püttmann, 2012) ein Konzept zur Interaktion mit Visualisierungen auf Basis von Touch-Gesten erarbeitet. In Kapitel 2.5.4 wurde dargestellt, dass sich hochauflösende Displays für die Integration in multimodalen Umgebungen eignen. Um die Interaktionsmöglichkeiten in beiden Arbeiten zu optimieren und zu erweitern, soll die Möglichkeit untersucht werden, das Konzept zur Interaktion mit Powerwall-Visualisierungen und das Konzept zur Interaktion mit Visualisierungen auf Tabletop-Computern in einer multimodalen Interaktionsumgebung zu kombinieren.

Entwicklung zweier Anwendungsszenarien

Das zu entwickelnde Interaktionskonzept soll auf möglichst viele Visualisierungstypen anwendbar sein. Um die Anforderungen an ein solches Interaktionskonzept formulieren zu können, werden beispielhaft zwei Szenarien in unterschiedlichen Gebieten der Visualisierung erstellt. Das erste Szenario stellt den Einsatz der multimodalen Interaktionsumgebung im Rahmen eines Anwendungsfalls auf dem Gebiet der informationswissenschaftlichen Visualisierung vor. Das zweite Szenario beschreibt die Anwendung des Interaktionskonzepts bei der Steuerung von wissenschaftlichen Visualisierungen.

Implementierung eines Prototyps

Das Konzept zur Interaktion mit Powerwall-Visualisierungen soll in Form eines Prototyps implementiert werden. Dabei muss der Prototyp eines der beiden zu entwickelnden Anwendungsszenarien umsetzen und auf einer Powerwall lauffähig sein. Der Prototyp soll

Schnittstellen zur Anbindung an die jeweilige Visualisierungssoftware bereitstellen. Sofern die Einbettung in eine multimodale Umgebung möglich ist, soll in Kombination mit dem Prototyp aus der Diplomarbeit „Tabletop-Computer-basierte Steuerung für Powerwall-Visualisierungen“ (Püttmann, 2012) eine Interaktionsumgebung geschaffen werden. Diese Umgebung wird dann Interaktionen mit Powerwall-Visualisierungen durch Freihandgesten und Touch-Interaktion mit Tabletop-Computern ermöglichen.

Evaluation

Abschließend ist gefordert, dass das im Rahmen dieser Diplomarbeit entwickelte Interaktionskonzept mithilfe des implementierten Prototyps in einer qualitativen Benutzerstudie evaluiert wird. Es muss überprüft werden, ob das Gesamtkonzept der Idee eines Natural User Interfaces entspricht und sich die implementierten Gesten für die jeweiligen Interaktionen eignen. Gleichzeitig können im Rahmen der Evaluation weitere Interaktionsgesten durch die Probanden vorgeschlagen werden.

3.2 Lösungsansatz

Zur Bearbeitung der Aufgabenstellung dieser Arbeit wird folgendermaßen vorgegangen:

1. **Formulierung der Szenarien**
   Anhand der Anforderungen werden die beiden Szenarien zur Interaktion mit Powerwall-Visualisierungen entwickelt. Das Szenario im Bereich der Informationsvisualisierung soll den Ablauf einer Eyetracking-Analyse mithilfe einer gestengesteuerten Interaktionsumgebung darstellen. Das zweite Szenario überträgt die Anforderungen auf das Gebiet der wissenschaftlichen Visualisierungen. Die Szenarien werden in Kapitel 4.3 vorgestellt.

2. **Anforderungsanalyse**
   Zunächst werden die Anforderungen an eine gestengesteuerte Interaktionsumgebung erhoben. Dies geschieht zunächst exemplarisch anhand der Beobachtung einer Situation, in der eine solche Interaktionsumgebung wünschenswert wäre. Hierzu wird auf dem Feld der informationswissenschaftlichen Visualisierung die Auswertung einer Eyetracking-Studie verfolgt. Ein Bericht über die Beobachtungen findet sich in Kapitel 4.1.1.
   In Kombination mit den Grundlagen aus Kapitel 2 werden daraus sowohl Anforderungen an die Gestensteuerung für Powerwall-Visualisierungen als auch an eine multimodale Interaktionsumgebung abgeleitet und in Kapitel 4.2 dokumentiert.

3. **Entwicklung des Interaktionskonzepts**
   Basierend auf den Anforderungen und den beiden Anwendungsszenarien wird ein allgemeines Interaktionskonzept zur Gesteninteraktion mit Powerwall-Visualisierungen erstellt. Dieses Interaktionskonzept wird in Kapitel 5 beschrieben.

4. **Design einer Architektur für die multimodele Interaktionsumgebung**
   In Zusammenarbeit mit der Diplomarbeit „Tabletop-Computer-basierte Steuerung für Powerwall-Visualisierungen“ (Püttmann, 2012) wird das Interaktionskonzept zur Gestensteuerung einer Powerwall mit dem Interaktionskonzept für Tabletop-Computer abgestimmt. Anschließend wird ein Kommunikationsframework entwickelt, dass den Einsatz mehrerer unterschiedlicher Geräte innerhalb der Interaktionsumgebung ermöglicht. Die Architektur dieses Kommunikationsframeworks und die Integration der Powerwall-Interaktion in dieses Framework sind in Kapitel 6 dokumentiert.

5. **Implementierung des Prototyps**
   Der Prototyp zur Gesteninteraktion mit Powerwall-Visualisierungen wird mithilfe des zuvor erstellten Kommunikationsframework implementiert. Dies erlaubt, frühzeitig die multimodale Interaktionsumgebung durch die Kombination von Powerwall-Interaktion und Tabletop-Computer-Interaktion zu testen. Zur Erkennung der Gesteninteraktion mit Powerwall-Visualisierungen werden Microsoft Kinect-Sensoren verwendet. Es werden die Interaktionen implementiert, die im Interaktionskonzept entwickelt wurden.
   Zusätzlich werden im Prototyp Komponenten verwendet, die bereits während der Anforderungsanalyse entwickelt wurden. Hierzu zählen ein Datenbankmodell für Eyetracking-Daten, eine Heatmap-Visualisierung für Daten aus dem Eyetracking-Datenmodell, sowie Vorarbeiten mit dem Kinect-Sensor.

6. **Evaluation des Interaktionskonzepts**
   Das Interaktionskonzept und dessen Implementierung im Prototyp sollen in einer Benutzerstudie bewertet werden. Gleichzeitig können mithilfe der Probanden weitere Gesten zur Interaktion mit Powerwall-Visualisierungen gefunden werden. Der Aufbau dieser Studie, die Durchführung einer Pilotstudie und die Ergebnisse der Pilotstudie sind in Kapitel 8 aufgeführt.

4 Ergebnisse der Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an das zu entwickelnde Interaktionskonzept vorgestellt. Zunächst werden Beobachtungen beschrieben, die beim Ablauf der Auswertung einer Eyetracking-Studie ohne ein gestenbasiertes Interaktionskonzept angestellt wurden. Daraus werden anschließend Anforderungen an das zu entwickelnde Interaktionskonzept abgeleitet. Um diese Anforderungen in einer Studie evaluieren zu können, werden abschließend zwei Szenarien vorgestellt, die das Interaktionskonzept anwenden.

4.1 Beobachtungen aus der Studie „Visual Elements“

Dieser Abschnitt stellt zunächst die Studie „Visual Elements“ vor, die am Institut für Visualisierung und Interaktive System der Universität Stuttgart durchgeführt wurde. Anschließend werden die Möglichkeiten aufgezeigt, mit denen diese Studie bisher ausgewertet werden konnte. Zum Abschluss wird der Verlauf der Auswertung der Studie „Visual Elements“ skizziert.

4.1.1 Die „Visual Elements“-Studie

Im Zeitraum von Mai bis Juli 2011 wurde am VIS die Studie „Visual Elements“ durchgeführt. Ziel dieser Studie war es für verschiedene Visualisierungstechniken die Augenbewegungen von Probanden aufzuzeichnen. Dieses Vorgehen wird als „Eye-Tracking“ bezeichnet (Duchowski, 2007). Hierfür wurden den Probanden Aufgaben aus drei verschiedenen Themengebieten gestellt, die anhand einer grafischen Darstellung beantwortet werden mussten. Die Probanden mussten Punkte aus einem Koordinatensystem ablesen, größte Farbflächen bestimmen und Dreiecke auf ihre mathematische Ähnlichkeit hin vergleichen. Zusätzlich zu den Augenbewegungen wurden die Antwort der Probanden auf die jeweilige Frage und die Dauer zur Lösung der Aufgabe erhoben.

Aufgabe 1: Koordinaten ablesen

Im ersten Teil der Studie wurden Probanden ein-, zwei- und dreidimensionale Koordinatensysteme mit unterschiedlichen Skalen präsentiert. Aus diesen Koordinatensystemen mussten einzelne Punkte, die durch rote Kreuze markiert waren, abgelesen werden.

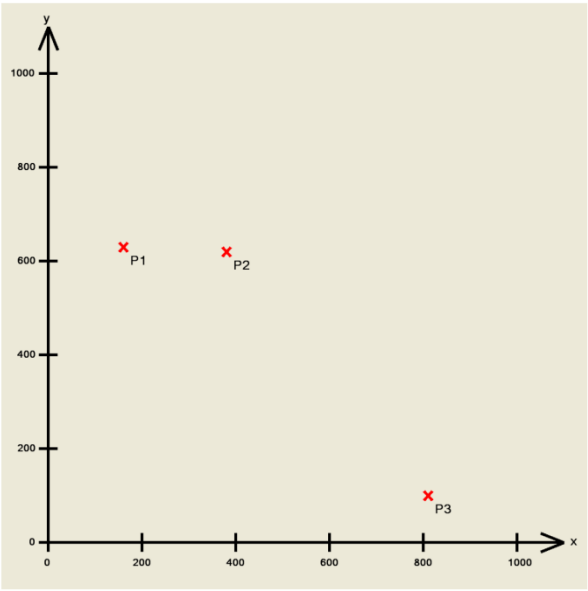


Abbildung 31: Aufgabe aus dem ersten Teil der Studie "Visual Elements" - Die Punkte P1, P2 und P3 mussten von den Probanden abgelesen werden.

Aufgabe 2: Größte Fläche bestimmen

Zum zweiten waren die Probanden aufgefordert auf einer grafischen Darstellung die Farbe der größten Fläche zu identifizieren. Dies waren entweder Kartendarstellungen, auf denen Länder unterschiedlich eingefärbt waren, oder Treemaps unterschiedlicher Größe.

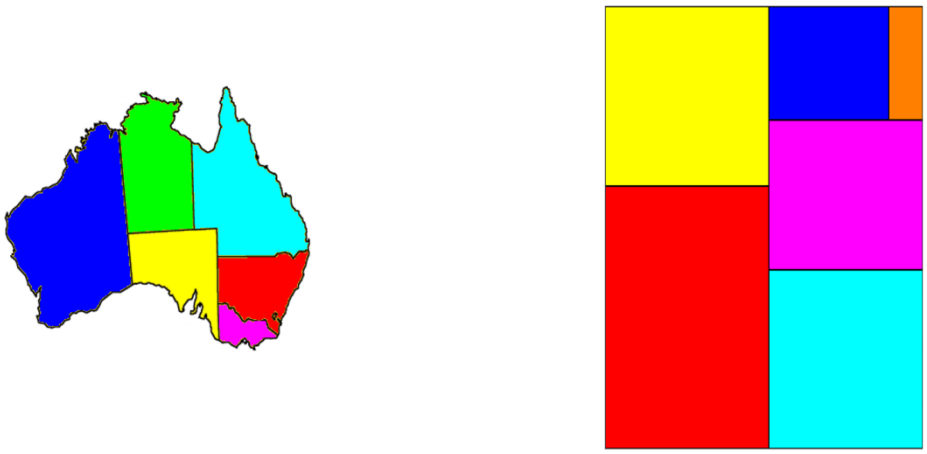


Abbildung 32: Aufgaben aus dem zweiten Teil der Studie "Visual Elements" - Aus den Abbildungen musste die größte Farbfläche bestimmt werden.

Aufgabe 3: Dreiecke vergleichen

Die dritte Aufgabe der Studie „Visual Elements“ bestand darin je zwei Dreiecke auf ihre mathematische Ähnlichkeit hin zu vergleichen.

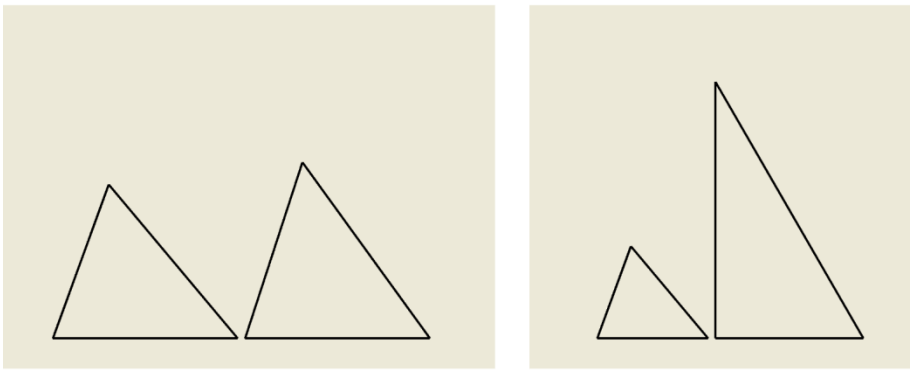


Abbildung 33: Dreiecke, die im dritten Teil der Studie "Visual Elements" verglichen werden mussten.

In jeder dieser Aufgaben wurden den Testpersonen 30 verschiedene Darstellungen zur Betrachtung präsentiert. Während der Bearbeitung dieser Aufgaben wurde die Zeit gestoppt, die die Probanden zur Bearbeitung der jeweiligen Aufgabe benötigten. Darüber hinaus wurden die Augenbewegungen der Probanden aufgezeichnet und die Antworten der Probanden festgehalten.

4.1.2 Auswertungsmöglichkeiten

Folgende grafische Visualisierungstechniken standen zur Auswertung der Studie „Visual Elements“ zur Verfügung: Heatmaps, Scan-Paths und Parallel Scan-Paths. Diese Techniken stellen die Blickpunkte (Fixationen) der Probanden während der Bearbeitung der Aufgaben dar und werden im Folgenden kurz vorgestellt.

Heatmap Visualisierung

Die Heatmap-Darstellung summiert die Fixationen eines Probanden über jeden Bildpunkt und stellt dies farblich in einer Hitzekarte dar. Bereiche, die ein Proband häufig oder lange betrachtet hat, werden rot dargestellt. Bereiche, die selten betrachtet wurden, werden hingegen grün dargestellt. Dazwischen sind Abstufungen von rot, orangeüber gelb nach grün möglich.
Abbildung 34 zeigt eine Darstellung der Blickpunkte aller Probanden beim Ablesen zweier Punkte aus einem Koordinatensystem. An jedem Pixel des Bilds wurden die Blicke der Probanden addiert und farbig dargestellt. Rot dargestellte Bereiche wurde häufig oder lange von Probanden geblickt. Grüne Bereiche wurden von Probanden selten betrachtet.

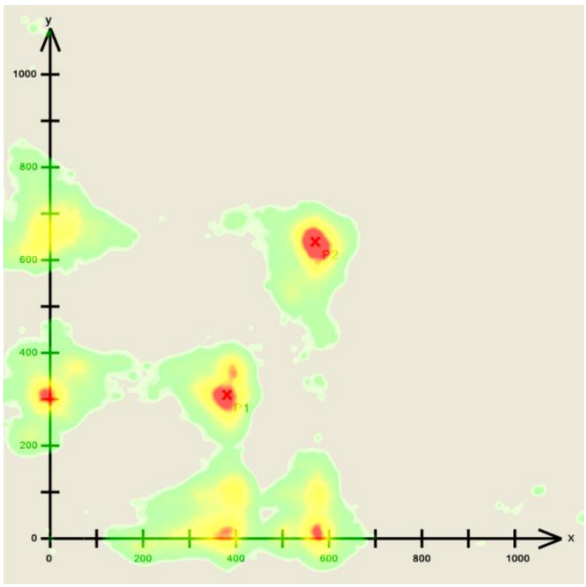


Abbildung 34: Darstellung der Blickpunkte aller Probanden beim Ablesen zweier Punkte aus einem Koordinatensystem in einer Heatmap. Rot dargestellte Bereiche wurden von Probanden häufig oder lange fixiert, grün dargestellte Bereiche wurden kaum oder nur flüchtig fixiert.

Scan-Path Visualisierung

Diese Visualisierung stellt die Fixationen der Probanden als Punkte dar. Die einzelnen Fixationen werden nach ihrer zeitlichen Abfolge miteinander über Linien verbunden. Diese Visualisierungsform wird als „Scan-Path“ bezeichnet und erlaubt die Bewegung der Blicke eines Probanden nachzuverfolgen.

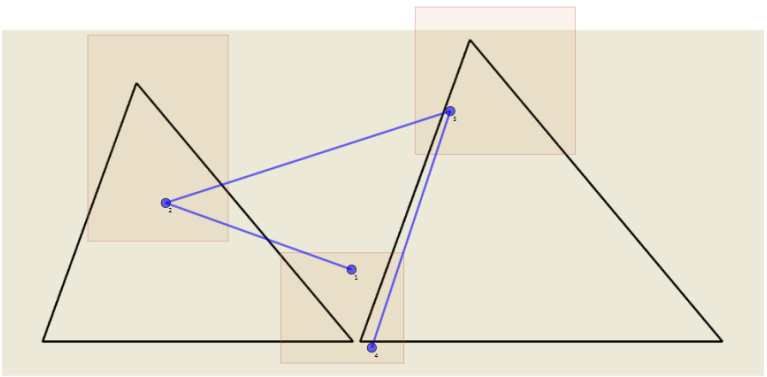


Abbildung 35: Scan-Path eines Probanden beim Vergleich zweier Dreiecke. Die Reihenfolge der Fixationen des Probanden wird durch die blaue Linie dargestellt.

Parallel Scan-Path Visualization

Über diese grafischen Darstellungen hinaus wurden im Rahmen der Diplomarbeit „Visuelle Analyse von Eye-Tracking Daten“ von Xuemei Chen weitere Visualisierungen entwickelt, die zur Auswertung der Studie „Visual Elements“ herangezogen werden konnten (Chen, 2011). Diese Visualisierungen bedienen sich dem Konzept der „Areas of Interest“ (AOI). AOIs sind Bereiche innerhalb des Betrachtungsfelds eines oder mehrerer Probanden, die von den Probanden lange oder häufig betrachtet wurden, und daher für die Auswertung von besonderem Interesse sind. Unter anderem werden hierzu die Anzahl der Fixationen innerhalb einer AOI, die Verweildauer innerhalb einer AOI und die Wechsel zwischen AOIs in einem System ähnlich paralleler Koordinaten dargestellt.
In vertikaler Richtung stellt eine Zeitachse den Verlauf der Betrachtung dar. Für jede AOI wird eine weitere Koordinatenachse parallel zur Zeitachse dargestellt. Auf Basis dieses Ansatzes wurden drei Varianten der Parallel Scan-Paths entwickelt (Raschke, Chen, & Ertl, 2012):

- **Gaze Duration Sequence Diagram**
  In diesem Diagrammtyp wird die Verweildauer innerhalb einer AOI als vertikale Farblinie auf der entsprechenden Koordinatenachse visualisiert. Horizontale Verbindungslinien zwischen Koordinatenachsen entsprechen einem Blickwechsel zwischen den beiden AOIs.

- **Fixation Point Diagram**
  Ähnlich wie beim Gaze Duration Sequence Diagram werden Fokuswechsel zwischen AOIs durch Verbindungslinien zwischen Koordinatenachsen dargestellt. Allerdings wird als Verbindungspunkt auf der Koordinatenachse der zeitliche Mittelpunkt der Verweildauer auf der AOI gewählt. Um diesen Mittelpunkt herum werden die einzelnen Fixationen innerhalb der AOI im zeitlichen Verlauf durch Punkte auf der Koordinatenachse markiert.

- **Gaze Duration Distribution Diagram**
  Zusätzlich zu den Fokuswechseln zwischen einzelnen AOIs wird in diesem Diagrammtyp die kumulierte Verweildauer der Fokussierung innerhalb einer AOI als Balkendiagramm dargestellt.

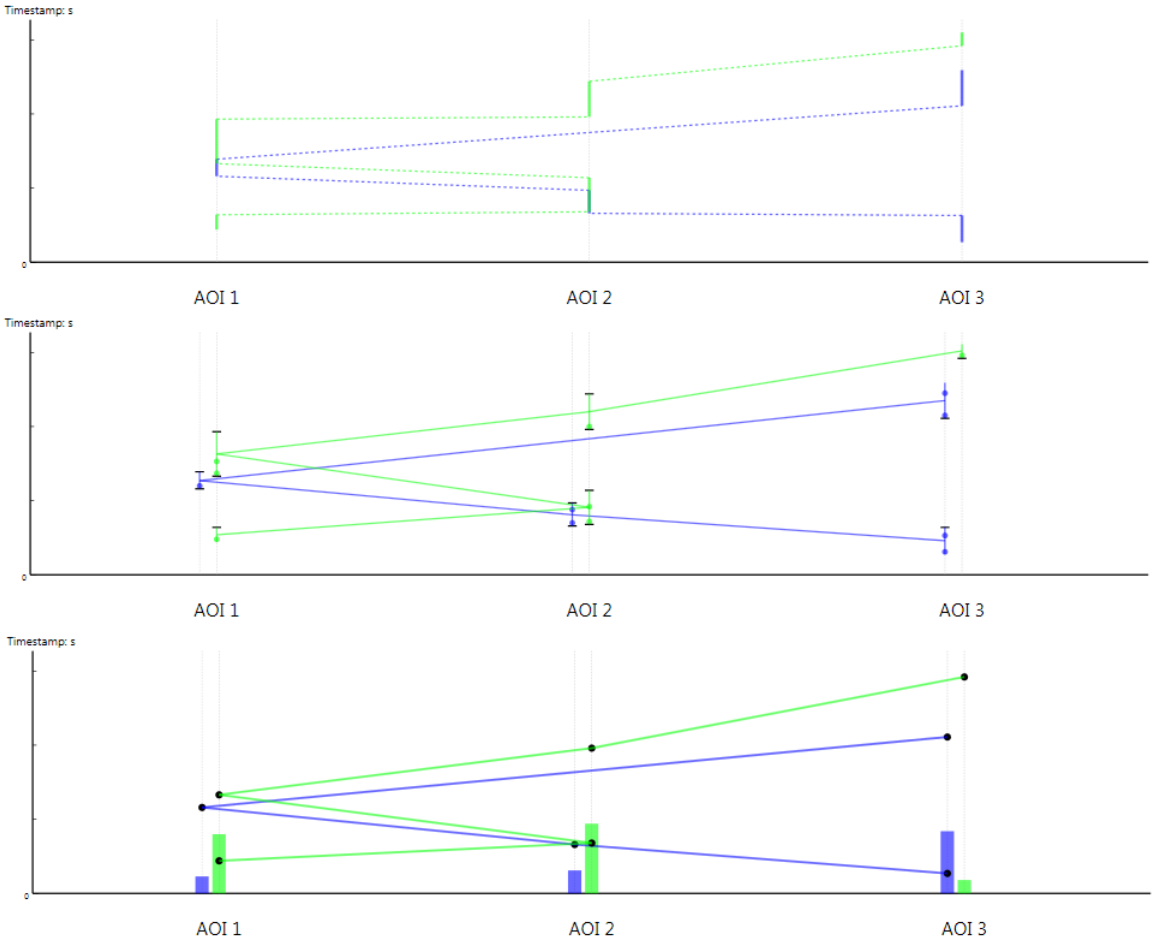


Abbildung 36: Scan-Paths zweier Probanden über drei Areas of Interest. oben: Gaze Duration Sequence Diagram, mitte: Fixation Point Diagram, unten: Gaze Duration Distribution Diagram.

4.1.3 Auswertung

Die Studie „Visual Elements“ wurde mit 30 Probanden durchgeführt. In der Auswertung wurde der Hypothese nachgegangen, dass mehrere Personen eine ähnliche Betrachtungsstrategie bei der Beantwortung der Fragen verfolgen. Hierzu wurden die in Kapitel 4.1.2 beschriebenen Visualisierungen verwendet. Bevor die eigentliche Auswertung beginnen konnte, mussten alle Daten, die bei der Studie erhoben wurden, in die Formate der verwendeten Programme konvertiert werden: Aus den Dateien, die vom EyeTracking zur Verfügung standen, mussten relevante Informationen extrahiert und in eine Datenbank für die Parallel Scan-Paths importiert und ebenfalls für die statistische Auswertung konvertiert werden. Um die Übersicht zu behalten, mussten Dateinamen gewählt werden, die eine spätere Zuordnung durch die auswertenden Personen ermöglichten.

Die Auswertung wurde zunächst von einer einzelnen Person an einem Desktop-PC durchgeführt. Zu Beginn wurde die Grafik, welche den Testpersonen dargestellt wurde, gesucht und in einem Bildbetrachtungsprogramm geöffnet. Anschließend wurde eine Heat-Map-Darstellung, die mit der Software des Eye-Traking-Systems generiert wurde, in einem neuen Fenster neben der Grafik positioniert. Auf dieser Heat-Map sind alle Blickpunkte aller Probanden kumuliert dargestellt. Aus diesen Informationen wurde abgeleitet, welche Bereiche auf der Grafik für die Probanden zur Lösung der gestellten Aufgabe von besonderem Interesse gewesen sein könnten. Daraufhin wurden im Programm der Parallel Scan-Paths entsprechende Areas of Interest angelegt. Nun mussten aus einer Liste alle Probanden ausgewählt werden, um diese dann in einem weiteren Fenster auf einem Gaze Duration Sequence Diagram darzustellen.

In dieser Darstellung konnte nun der Fragestellung nach Gemeinsamkeiten in der Betrachtungsstrategie nachgegangen werden. Hierzu wurden zunächst die 30 Scan-Paths optisch auf Gemeinsamkeiten verglichen und Merkmale wie gleiche Betrachtungsreihenfolge, gleicher Startpunkt oder ähnliche Dauer zur Lösung der Aufgabe gesucht. Anschließend wurden diese Gruppen in jeweils einem neuen Fenster getrennt dargestellt.
An diesem Punkt fiel auf, dass der Platz auf dem Bildschirm des Desktop-PCs nicht ausreicht, um alle Fenster gleichzeitig darzustellen. Es musste häufig nach den passenden Fenstern zur Beurteilung und zum Vergleich eines Merkmals gesucht werden. Dieser Nachteil zeigte sich insbesondere, als die ersten Ergebnisse mit Kollegen besprochen werden sollten.

Um mehr Platz zur Platzierung von Fenstern zur Verfügung zu haben und eine angenehmere Atmosphäre zur Teamarbeit zu schaffen, wurde beschlossen die Analyse an einer Powerwall fortzusetzen. Die erste Hürde hierbei lag darin, den Zustand der Analyseumgebung auf dem Desktop-PC auf der Powerwall wiederherzustellen. Bevor mit der Analyse fortgefahren werden konnte mussten alle Schritte, die am Desktop-PC durchgeführt wurden, noch einmal nachvollzogen werden. Alle Dateien mussten kopiert und erneut geöffnet werden. Anschließend mussten sämtliche Gruppierungs- und Filteraktionen aus dem Gedächtnis heraus ein zweites Mal ausgeführt werden, um dieselben Ansichten wie auf dem Desktop-PC zu erzeugen.

Nachdem diese Vorbereitungen getroffen wurden, konnte mit der Auswertung der Studie fortgefahren werden. Da viele Daten gleichzeitig angezeigt und somit direkt verglichen werden konnten, entstand schnell ein angeregtes Gespräch über diese Daten und deren Interpretation. Als gravierender Nachteil der Powerwall entpuppte sich allerdings die Interaktion mit den angezeigten Daten: um Veränderungen an der Darstellung der Daten durchzuführen – zum Beispiel um ein Fenster zu verschieben, ein weiteres Fenster anzuzeigen oder die dargestellte Informationsmenge zu ändern – musste jeweils ein Mitglied die Diskussionsrunde verlassen. Auch, wenn der Weg zum Steuercomputer der Powerwall nur wenige Meter betrug, unterbrach dies jedes Mal den Analysefluss. Die Gespräche mussten unterbrochen werden, bis derjenige, der zum Steuercomputer beordert wurde, dort angekommen, sich neu orientiert und die gewünschte Manipulation durchgeführt wurde. Es entstanden unproduktive Zwischengespräche darüber, welches Objekt genau verändert werden sollte und in welcher Weise. Zum Beispiel: „nein, das Fenster darunter“ oder „noch weiter links“.
Nachdem die gewünschte Manipulation ausgeführt wurde, musste der Analysefluss wieder aufgenommen werden. Hatte die Gruppe die Diskussion ohne die Person, die am Steuerrechner war, fortgesetzt, musste diese wieder auf den aktuellen Informationsstand gebracht werden. Im anderen Fall verfiel die Gruppe während der Manipulation in Gespräche über andere Themen und alle mussten den „roten Faden“ wieder aufnehmen.

Über die Distanz der Powerwall zum Steuercomputer hinaus erschwerte die Interaktion mit dem Programm der Parallel Scan-Paths den Analysefluss. Wurden beispielsweise Probanden auf mehreren Diagrammen gleichzeitig angezeigt, waren diese auf jedem Diagramm in einer unterschiedlichen Farbe dargestellt. Somit fielen Vergleiche der Betrachtungsmuster eines Probanden in verschiedenen Aufgaben oder Vergleiche mehrerer Probanden miteinander schwer. Ebenso waren die Schritte aufwändig, die am Steuercomputer durchgeführt werden mussten, um einzelne Probanden aus Darstellungen zu entfernen oder neue Darstellungsgruppen zu erstellen.

Nach Beendigung der Analyse einer Aufgabe mussten die Ergebnisse von Hand dokumentiert, in Form von Screenshots abgespeichert und sämtliche Fenster geschlossen werden. Andernfalls wäre die Verwirrung, die aufgrund der hohen Anzahl von Fenstern ohnehin schon bestand, noch größer geworden.

4.2 Abgeleitete Anforderungen

Aus den Grundlagen in Kapitel 2 und den in Kapitel 4.1 genannten Vorgaben, Hürden, Schwierigkeiten und Problemen lassen sich die folgenden Anforderungen an eine gestengesteuerte Interaktionsumgebung für Powerwall-Visualisierungen (Kapitel 4.2.1) und das Konzept zur Interaktion mit Powerwall-Visualisierungen (Kapitel 4.2.2) ableiten.

Um die Erfüllung dieser Anforderungen nicht im Voraus zu beeinflussen werden folgende Begriffe verwendet:

Ansicht: Einheit, in der Informationen angezeigt werden. Die Manipulation einer Ansicht hat keine Auswirkungen auf die darin enthaltenen Informationen. Das Äquivalent auf einem Desktop-PC sind Fenster.

Visualisierung: Grafische Form, in der Daten innerhalb einer Ansicht visualisiert werden. Dies könnte beispielsweise eine HeatMap oder eine tabellarische Visualisierung sein.

Darstellung: Untereinheit einer Visualisierung und konkrete Darstellung eines Datums. Beispielsweise ein roter Kantenzug für die Augenbewegung eines Probanden.

4.2.1 Anforderungen an eine Interaktionsumgebung

Während der Auswertung der Studie „Visual Elements“ an der Powerwall wurden von den Teilnehmern Ideen und Anregungen für eine Umgebung zur Interaktion mit der Powerwall gesammelt. Diese Ideen werden hier in der Reihenfolge aufgeführt, wie wichtig sie für die Teilnehmer der Studienauswertung erschienen:

1. **Es sollen viele Informationen gleichzeitig angezeigt werden können.**
   Der Bildschirm an einem Standard-PC bietet nicht genügend Platz, um es mehreren Personen zu ermöglichen, alle Daten bequem zu erkennen. Es soll ein möglichst großflächiges Display eingesetzt werden, damit eine Gruppe von Personen die Möglichkeit hat sämtliche Daten zu betrachten.

2. **Die Visualisierungen sollen von mehreren Personen problemlos betrachtet werden können.**
   Der Raum um einen Schreibtisch bietet in der Regel nicht genügend Platz für eine Gruppe von Personen, denen Ergebnisse einer Analyse präsentiert werden sollen, oder die an einer Gruppenanalyse teilnehmen.

3. **Ansichten sollen schnell und einfach erreichbar sein.**
   Insbesondere bei großen oder lang dauernden Analysen werden mit der Zeit viele verschiedene Ansichten geöffnet. Es soll möglich sein diese bei Bedarf schnell wieder zu finden und weiterzuarbeiten.

4. **Daten, die an mehreren Stellen angezeigt werden, sollen dieselbe Darstellung haben.**
   Werden dieselben Daten an verschiedenen Stellen oder in verschiedenen Ansichten angezeigt, soll erkennbar sein, dass diese Daten dieselbe Quelle besitzen. Zum Beispiel soll die Farbe, in der die Diagramme eines Probanden dargestellt werden, in allen Darstellungen dieselbe sein.

5. **Alle Ansichten in der Interaktionsumgebung sollen auf derselben Datenbasis arbeiten.**
   Es soll vermieden werden, dass in der Vorbereitung auf die Analyse oder sogar während der Analyse Zeit aufgewendet werden muss, um Daten zwischen verschiedenen Formaten zu konvertieren. Alle Visualisierungen und Interaktionen sollen auf einer homogenen Datenstruktur arbeiten. Ein einmaliger Datenimport vor Analysebeginn soll genügen.

6. **Mehrere Personen sollen in der Umgebung interagieren können.**
   Die Verwendung eines Standard-PC beschränkt die Interaktionen mit dem Programm auf eine Person, die Tastatur und Maus bedient. In einer Diskussion müssen die Teilnehmer sich entweder an den Eingabegeräten abwechseln oder sich gegenseitig Anweisungen geben. Es soll möglich sein, dass jeder Teilnehmer problemlos mit der Interaktionsumgebung interagiert – nach Möglichkeit sogar mehrere Benutzer gleichzeitig.

7. **Es sollen unterschiedliche Geräte verwendet werden können.**
   Die Interaktionsumgebung soll sich nicht nur auf ein großes Display beschränken. Es soll möglich sein, weitere Geräte in die Interaktionsumgebung zu integrieren. Beispielsweise ein Notebook, wie in Punkt 6 beschrieben. In Zusammenarbeit mit der Diplomarbeit „Tabletop-Computer-basierte Steuerung für Powerwall-Visualisierungen“ (Püttmann, 2012) sollen Tabletop-Geräte oder Tablets in die Interaktionsumgebung integriert werden können.

8. **Daten sollen zwischen Geräten, die der Interaktionsumgebung angehören, ausgetauscht werden können.**
   Werden mehrere Geräte und unterschiedliche Gerätetypen in der Interaktionsumgebung verwendet, sollen diese Geräte die Daten untereinander austauschen können. Dadurch wird vermieden, dass vor Beginn einer Analysesitzung die Geräte manuell auf denselben Datenstand gebracht werden müssen.

9. **Werden Daten auf mehreren Geräten gleichzeitig angezeigt, sollen die Ansichten synchron gehalten werden können.**
   Stellen beispielsweise mehrere Geräte dieselben Visualisierungen dar, sollen Interaktionen, die auf einem Gerät durchgeführt werden, auf die anderen Geräte übertragen und dort ebenfalls ausgeführt werden.

10. **Es soll möglich sein sich als Einzelperson oder Kleingruppe zurückzuziehen und Teilaspekte separat zu analysieren.**
    Neben einem öffentlichen Bereich, in dem alle Daten auf allen beteiligten Geräten synchronisiert angezeigt und von allen Teilnehmern bearbeitet werden können, sollen Bereiche definiert werden können, auf die nur Einzelpersonen oder eine Untergruppe zugreifen kann. Entstehen während der Analyse unterschiedliche Theorien, sollen diese in unterschiedlichen Gruppen ausgearbeitet und später wieder zusammengefügt werden können.

11. **Es soll möglich sein, Analysen vorzubereiten und die vorbereiteten Daten in die Interaktionsumgebung laden zu können.**
    Analysen werden in der Regel von Einzelpersonen vorbereitet. Hierbei soll eine Vorauswahl getroffen werden können, welche Daten in der bevorstehenden Analyse verwendet werden sollen. Diese vorbereitenden Maßnahmen sollen nicht in der späteren Analyseumgebung getroffen werden müssen.

4.2.2 Anforderungen an ein Interaktionskonzept

Ebenso wie Ideen zur Gestaltung einer Interaktionsumgebung entstanden sind, wurden auch Vorschläge und Anforderungen an das Interaktionskonzept mit Powerwall-Visualisierungen zusammengetragen.

1. **Interaktionen mit einem großflächigen Display sollen direkt am Display stattfinden.**
   Es soll nicht nötig sein sich von den Darstellungen auf dem großen Display zu entfernen, um mit ihnen zu interagieren, wie es beispielsweise bei Steuercomputern der Fall ist. Dadurch soll gewährleistet werden, dass die Diskussionsgruppe nicht verlassen werden muss, um beispielsweise Darstellungen zu ändern.

2. **Interaktionen sollen den Analysefluss nicht unterbrechen.**
   Wenn mit Visualisierungen interagiert wird, soll dieser Vorgang beinahe unterbewusst durchgeführt werden können. Die Manipulationen sollen sich in den Gesprächs- und Analysefluss einfügen und diesen unterstützen. Unterbrechungen sind nicht erwünscht.

3. **Die Interkation soll intuitiv sein**
   Im Sinne eines Natural User Interfaces soll die Interaktion intuitiv und ohne großen Aufwand erlernbar sein. Die durchführbaren Interaktionen sollen zielführend eine Analysesituation, wie sie in Kapitel 4.3 beschrieben wird, unterstützen und sowohl Anfänger als auch Experten ansprechen.

4. **Das System soll prompt reagieren**
   Ebenfalls unter Gesichtspunkten eines Natural User Interfaces soll das System flüssig ablaufen und ohne große Ladezeiten auskommen. Hierdurch soll vermieden werden, dass der Analyseprozess durch Wartezeiten unterbrochen wird.

5. **Es soll möglich sein, mehrere Datensätze auf einfache Weise zu vergleichen**
   Verschiedene Ansichten sollen schnell erreicht und arrangiert werden können, sodass ein Vergleich zwischen deren Inhalten möglich wird.

6. **Es soll möglich sein, sich einen Überblick über die Daten zu verschaffen und bei Bedarf auf Detailinformationen zuzugreifen**
   Das Interaktionskonzept soll Shneiderman's Visual Information Seeking Mantra: „Overview first, zoom and filter, then details on-demand" unterstützen (Shneiderman, The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations, 1996). In der Überblicksphase sollen diejenigen Interaktionen durchführbar sein, die benötigt werden um sich einen Überblick zu verschaffen, Interaktionen auf Detailebene sollen erst bei Bedarf erreichbar sein.

7. **Es soll möglich sein, schnell zwischen verschiedenen Visualisierungen zu wechseln.**
   Sofern es mehrere Visualisierungen für bestimmte Daten gibt, soll der Wechsel zwischen diesen Visualisierungen ohne großen Aufwand möglich sein. Darüber hinaus sollen diese Aktionen ebenfalls keine Wartezeiten verursachen.

8. **Neue Ansichten sollen schnell erstellt werden können.**
   Um den erreichten Zustand einer Visualisierung zu erhalten, aber dennoch am selben Punkt fortfahren zu können, sollen Ansichten beispielsweise dupliziert werden können. Ebenso soll es möglich sein eine neue Ansicht im ungefilterten Ausgangszustand ohne Umwege zu erstellen.

9. **Es soll möglich sein direkt in einer Visualisierung zu filtern.**
   Soll die dargestellte Menge der Daten verändert oder eingeschränkt werden, sollen diese Aktionen direkt aus der jeweiligen Visualisierung angestoßen werden können. Es soll nicht nötig sein zur Filterung eine neue Ansicht zu erstellen.

10. **Aktionen in einer Visualisierung sollen direkt erreichbar sein.**
    Interaktionen mit einer Visualisierung sollen entweder in der Visualisierung selbst, in unmittelbarer Nähe zur Visualisierung oder in unmittelbarer Nähe zum Benutzer angestoßen werden können.

11. **Es soll möglich sein Ansichten wiederzuverwenden.**
    Um zu vermeiden, dass im Laufe der Benutzung eine Vielzahl von Ansichten mit unterschiedlichen Visualisierungen gleichzeitig angezeigt wird oder im Hintergrund geöffnet ist, sollen diese Ansichten wiederverwendet werden können. Beispielsweise könnten Ansichten die Möglichkeit bieten, andere Daten zu laden oder die Visualisierung zu ändern.

12. **Gesten zur Interaktion sollen eindeutig sein**
    Der Benutzer soll in jeder Situation wissen können, welche Gesten ihm nun zur Interaktion zur Verfügung stehen und jederzeit in der Lage sein zu beschreiben, welchen Effekt welche Geste in diesem Moment hat.
    Gleichzeitig soll die Interaktionsumgebung jede Geste eindeutig erkennen und auswerten können. Ebenso soll die Interaktionsumgebung feststellen können, wann eine Bewegung als Geste verstanden werden soll.

13. **Gesten sollen sozialverträglich sein**
    Benutzer sollen Gesten gerne ausführen. Gesten sollen nicht peinlich oder unangenehm für den Ausführenden sein. Sie sollen nicht zu ausladend sein, um in einer Diskussionsgruppe, die eng beieinander steht, ausgeführt werden zu können.

14. **Gesten sollen übertragbar sein**
    Werden mehrere unterschiedliche Geräte wie beispielsweise eine Powerwall und Tabletop-Computer in einer gemeinsamen Interaktionsumgebung verwendet, sollen die Gesten von einem auf ein anderes Gerät übertragen werden können. Die Gestensprachen der verschiedenen Geräte sollen möglichst übereinstimmen.

15. **Gesten sollen logisch sein**
    Die Gestensprache, die zur Interaktion verwendet wird, soll logisch aufgebaut sein. Gesten, die ausgeführte Aktionen kompensieren, sollen erkennbar sein. Die Zuordnung von Gesten soll sinnvoll sein. Beispielsweise sollen deiktische Gesten zur Positionsbestimmung verwendet werden oder die Reaktion des Systems der Ausführungsrichtung einer Geste entsprechen.

4.3 Szenarien

In den folgenden beiden Abschnitten werden zwei Szenarien vorgestellt, in denen eine Interaktionsumgebung für Powerwall-Visualisierungen Anwendung findet. Diese Szenarien beschreiben mögliche Abläufe bei der Verwendung einer Interaktionsumgebung, wie sie in den Anforderungen in Kapitel 4.2 beschrieben sind. Darüber hinaus bestimmen sie den Umfang der Funktionalität des Prototyps, der im Rahmen dieser Diplomarbeit erstellt werden soll.

4.3.1 Szenario: Eyetracking-Analyse

Das erste Szenario stellt die Anwendung einer Interaktionsumgebung im Bereich der Informationsvisualisierung dar. Es beschreibt den Einsatz dieser Interaktionsumgebung bei der Auswertung einer Eyetracking-Studie. Dieses Szenario ist direkt aus den Erfahrungen bei der Auswertung der Studie „Visual-Elements“ entstanden und beschreibt einen alternativen Analyseablauf zu dem aus Kapitel 4.1. Zunächst bereitet der Versuchsleiter der Eyetracking-Analyse die Daten zur Analyse vor, bringt die Daten in eine multimodale Interaktionsumgebung und führt dort mit Kollegen die Analyse durch. Die Ergebnisse der Analyse können vom Versuchsleiter außerhalb der Interaktionsumgebung weiter verwendet werden.

Offline-Vorbereitung der Eyetracking-Analyse

Der Versuchsleiter einer Eyetracking-Studie sitzt am Abend nach Beendigung der Studie zuhause und bereitet die morgige Auswertung dieser Studie vor. Er importiert die Daten der Studie in das System der multimodalen Interaktionsumgebung, welches auf seinem Notebook installiert ist.
Nach dem Import wählt der Versuchsleiter drei Aufgaben aus, über die in der bevorstehenden Besprechung diskutiert werden sollen.

Abbildung 37: Import der Eyetracking-Daten in das System der multimodalen Interaktionsumgebung.

Kurz vor Beginn der Analysesitzung

Der Versuchsleiter betritt vor seinen Kollegen die multimodale Interaktionsumgebung und tritt mit seinem Notebook an die Powerwall heran.

Über ein Identity-Tag an der Rückseite des Notebooks, in dem IP-Adresse, Port und Identifikation kodiert sind, wird das Notebook erkannt und in die Interaktionsumgebung integriert. Ein Notebook-Symbol wird auf der Powerwall angezeigt. Neben dem Notebook-Symbol werden die drei am Vorabend ausgewählten Aufgaben angezeigt. Mit einer Geste schiebt der Versuchsleiter die Aufgaben auf eine freie Fläche der Powerwall. Anschließend werden diese Aufgaben in den Daten-Explorern aller Geräte der Interaktionsumgebung angezeigt.

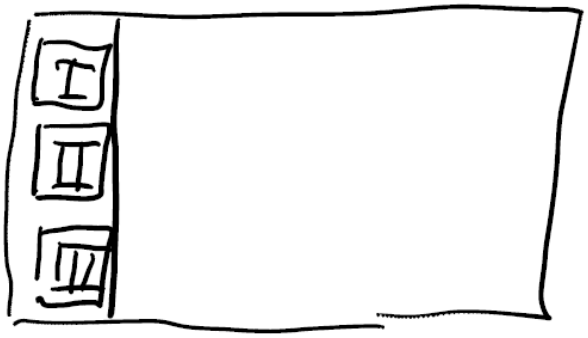


Abbildung 38: Die ausgewählten Aufgaben aus der Eyetracking-Studie werden im Daten-Explorer der Powerwall angezeigt.

Durchführung der Besprechung

Nachdem sich alle Teilnehmer der Besprechung in der Interaktionsumgebung eingefunden haben, beginnt der Versuchsleiter die Analyse. Er zieht mit einer Freihandgeste die erste Aufgabe aus dem Datenexplorer der Powerwall auf eine freie Fläche der Powerwall. Anschließend wird an dieser Stelle das Bild der Aufgabenstellung angezeigt. Mit einer weiteren Geste justiert der Versuchsleiter die Größe, in der das Bild angezeigt wird, um allen Kollegen eine einwandfreie Sicht zu ermöglichen. Anhand dieses Bildes erklärt der Versuchsleiter seinen Kollegen die Aufgabenstellung für die Probanden und seine These.

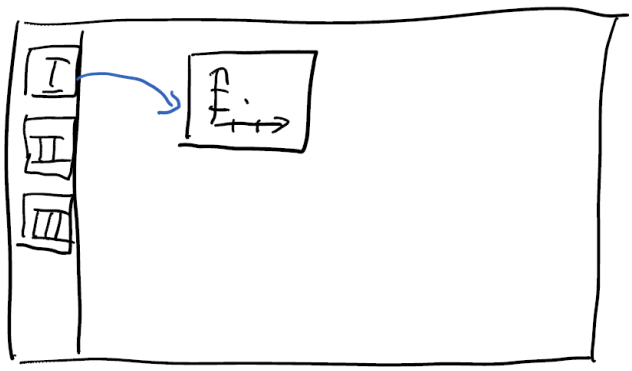


Abbildung 39: Mit einer Geste wird eine Aufgabe aus dem Daten-Explorer auf die Powerwall bewegt.

Damit während der gesamten Analyse das Bild der Aufgabenstellung sichtbar bleibt, dupliziert der Versuchsleiter mit einer Geste diese Ansicht. Nun werden zwei identische Ansichten dargestellt.

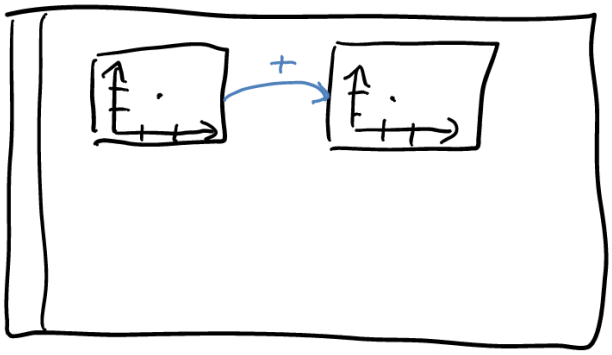


Abbildung 40: Mit einer Geste wird eine Ansicht auf der Powerwall dupliziert.

Auf der neu erstellten Ansicht führt der Versuchsleiter eine Wischbewegung nach links aus. Dadurch ändert sich die Visualisierung in eine HeatMap-Visualisierung.

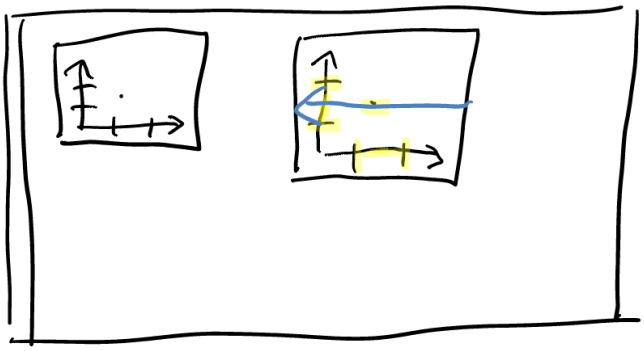


Abbildung 41: Durch eine Wischgeste wird die Visualisierung in einer Ansicht gewechselt.

Anhand der HeatMap-Visualisierung diskutieren die Teilnehmer, welche Bereiche des Bildes für die Probanden interessant gewesen sein müssen und nun näher untersucht werden sollen. Um diese Bereiche zu identifizieren, werden verschiedene Probanden nach und nach durch Gesten ausgewählt. Ist ein Proband ausgewählt, werden dessen Blickpunkte in der HeatMap hervorgehoben. Anschließend werden von den Teilnehmern der Konferenz mehrere AOIs durch entsprechende Gesten definiert.

Die AOIs werden sowohl in der HeatMap-Ansicht, als auch in der Ansicht mit dem Bild der Aufgabenstellung angezeigt.

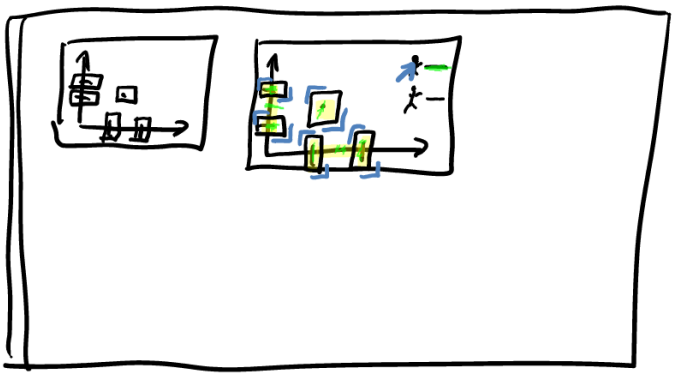


Abbildung 42: In den Ansichten werden die Areas of Interest (AOIs) dargestellt.

Mit einer weiteren Wischgeste nach links über die HeatMap-Ansicht wechselt diese zur Visualisierung eines Gaze-Duration-Diagramms, in dem alle Probanden angezeigt werden.
Der Versuchsleiter arrangiert die Reihenfolge der vertikalen Koordinatenachsen entsprechend seiner These.

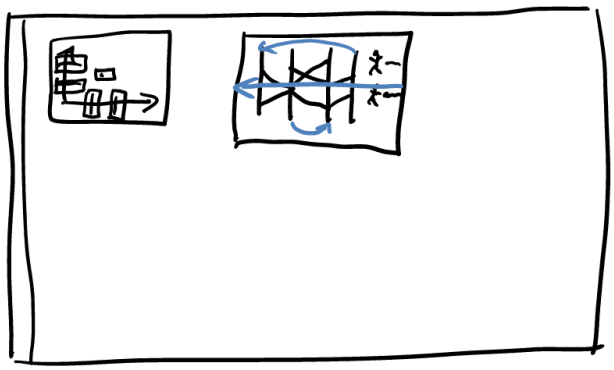


Abbildung 43: Mit Gesten werden die Achsen im Gaze Duration Diagram sortiert.

Auf dem Gaze-Duration-Diagramm fällt ein Proband auf, der offensichtlich ein statistischer Ausreißer ist. Mit einer Geste markiert der Versuchsleiter diesen Probanden und entfernt ihn mit einer weiteren Geste aus dieser Visualisierung.

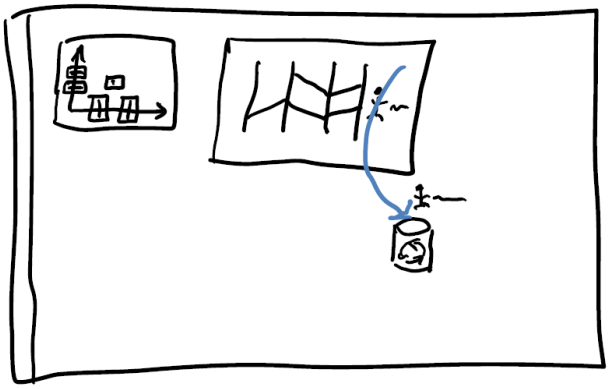


Abbildung 44: Mit einer Verschiebegeste wird ein Proband aus dem Diagramm in den Papierkorb geschoben und gelöscht.

Bei näherer Betrachtung scheinen sich die Probanden auf dem Gaze-Duration-Diagramm in zwei verschiedene Gruppen aufzuteilen. Den ersten Repräsentanten der ersten Gruppe wählt der Versuchsleiter mit einer Geste aus und zieht ihn aus der Visualisierung auf eine freie Fläche der Powerwall. Es erscheint dasselbe Gaze-Duration-Diagramm an dieser stellt. Allerdings enthält es ausschließlich die Darstellung des Probanden, der soeben aus dem ersten Diagramm herausgezogen wurde.
Der Versuchsleiter zieht auf dieselbe Methode alle Probanden, die zur selben Gruppe gehören, in das zweite Diagramm.

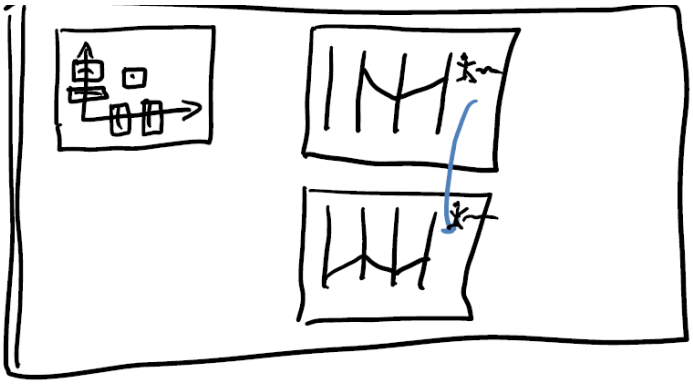


Abbildung 45: Die Probanden werden mit Gesten in zwei Gruppen eingeteilt.

Der Versuchsleiter möchte dieses Ergebnis in Form eines Screenshots festhalten, um es später in einem Artikel zu veröffentlichen. Mit einer Geste erstellt er aus den beiden Gaze-Duration-Diagrammen ein Bild und überträgt es auf sein Notebook.

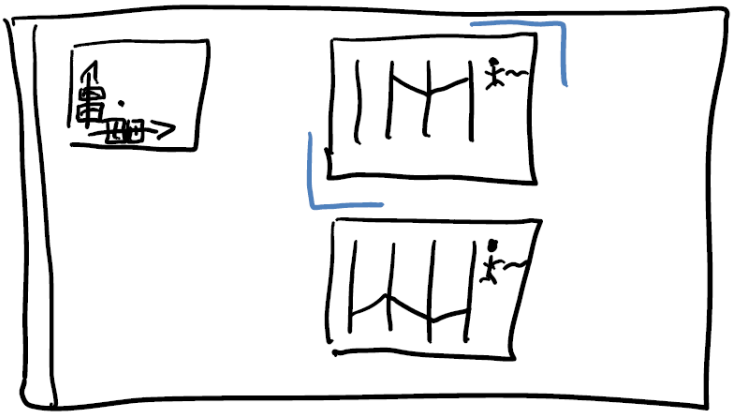


Abbildung 46: Mit einer Geste wird von der Ansicht ein Screenshot erstellt.

Um zu überprüfen, ob sich die beiden Gruppen in einer anderen Aufgabe genauso verhalten, führt der Versuchsleiter auf einem der beiden Diagramme eine Wischgeste nach unten (oder oben) aus. Alle drei Ansichten wechseln die zugrunde liegende Aufgabe.
Dies setzt allerdings voraus, dass die AOIs der ersten Aufgabe auch sinnvoll für die zweite Aufgabe sind. Andernfalls müssen diese zuerst angepasst werden.

Arbeiten in zwei Gruppen

Die Diskussion über die These des Versuchsleiters spaltet die Analysegruppe in zwei Lager mit gegensätzlichen Meinungen. Um eine Meinung genauer zu erörtern, kopiert sich ein Mitarbeiter die aktuelle Ansicht in den privaten Bereich auf seinen Tablet-PC. Anschließend zieht sich dieser Mitarbeiter mit einigen Kollegen zurück. Der Rest des Teams fährt mit der Analyse fort. Nachdem die Kleingruppe ihre separate Arbeit beendet hat, kehren die Kollegen zurück und übertragen ihre Ergebnisse wieder in den öffentlichen Raum an der Powerwall. Das gesamte Analyseteam kann nun die Ergebnisse beider Gruppen vergleichen.

Unterbrechung oder Abschluss der Analyse

Die Zeit, die für die Besprechung veranschlagt wurde, ist vorbei. Um die Analyse zu einem späteren Zeitpunkt fortsetzen zu können, oder die abschließenden Ergebnisse bei Beendigung der Analyse festzuhalten, speichert der Versuchsleiter den Zustand der Interaktionsumgebung.

Nachträgliche Wiederholung der Analyse

Auf einer Fortbildung zur Analyse von Eyetracking-Studien mit der Interaktionsumgebung möchte der Versuchsleiter die Vorgehensweise bei der letzten Analyse vorstellen. Hierzu lädt er den zuvor gespeicherten Zustand der Analyse.
Anschließend navigiert er mit zwei verschiedenen Gesten vorwärts und rückwärts durch die einzelnen Aktionen der gespeicherten Analyse und macht somit einzelne Aktionen rückgängig oder führt sie erneut aus und erklärt dabei dem Plenum das Vorgehen.

4.3.2 Szenario: MegaMol

Es soll gewährleistet sein, dass sowohl die Interaktionsumgebung als auch das Interaktionskonzept auch auf anderen Gebieten als der Informationsvisualisierung angewendet werden kann. Daher wurde dieses zweite Szenario erstellt. Es zeigt die Verwendung einer gestengesteuerten Interaktionsumgebung im Bereich der wissenschaftlichen Visualisierungen.

Mit der Visualisierungssoftware Megamol können Moleküle aus punktbasierte Datensets visualisiert werden (Grottel, 2012). Dieses zweite Szenario beschreibt beispielhaft die Verwendung der Interaktionsumgebung zur Visualisierung und Manipulation von Proteindaten mithilfe der MegaMol-Software. In diesem Szenario soll ein wissenschaftlicher Mitarbeiter seinem Kollegium die neuesten Erkenntnisse seiner Forschungsarbeit mit verschiedenen Proteinen präsentieren.

Das Anwendungsszenario für eine multimodale Interaktionsumgebung mit MegaMol beschreibt die Präsentation neuer Forschungsergebnisse im Bereich der Proteinvisualisierung. Zunächst bereitet ein Mitarbeiter die Moleküldaten für die Präsentation vor, integriert sie kurz vor Beginn der Präsentation in die Interaktionsumgebung und hält schließlich seinen Vortrag. In der anschließenden Diskussion werden Darstellungen der Proteine generiert, die der Mitarbeiter für die Verwendung in Publikationen speichert.

Offline-Vorbereitung

Im Rahmen der Vorbereitungen des Vortrags wählt der wissenschaftliche Mitarbeiter drei Proteine aus, anhand derer er seine Forschungsergebnisse präsentieren möchte. Auf seinem Notebook importiert der Mitarbeiter die Dateien mit den Protein-Daten in das System der multimodalen Interaktionsumgebung.
Für die importierten Daten werden Vorschaubilder im Daten-Explorer des Notebooks angezeigt.

Abbildung 47: Auf einem Notebook werden die Daten für die Präsentation geladen.

Kurz vor Beginn der Präsentation

Der wissenschaftliche Mitarbeiter betritt die multimodale Interaktionsumgebung und tritt mit seinem Notebook an die Powerwall heran. Das Notebook wird von der Interaktionsumgebung erkannt und in Kommunikationsinfrastruktur integriert. Auf der Powerwall erscheint ein Notebook-Symbol. Der wissenschaftliche Mitarbeiter aktiviert mit einer Geste das Notebook-Symbol. Neben dem Notebook-Symbol werden die Vorschaubilder der während der Vorbereitungen in das Notebook importierten Protein-Daten angezeigt. Mit einer weiteren Geste markiert der wissenschaftliche Mitarbeiter alle drei Vorschaubilder und bewegt sie auf eine freie Fläche der Powerwall.

Anschließend werden die Protein-Daten auf alle Geräte der multimodalen Interaktionsumgebung verteilt und in den Daten-Explorern aller Geräte angezeigt.

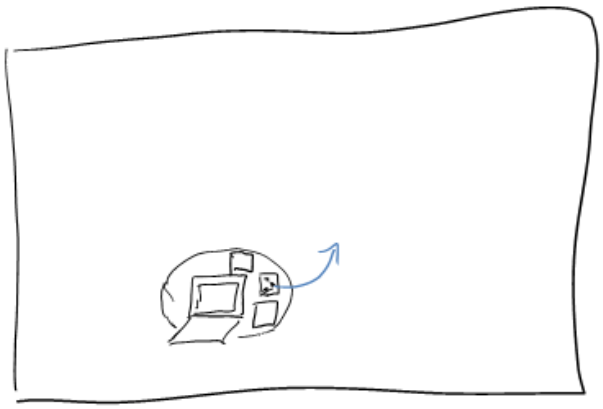


Abbildung 48: Über eine Geste werden die Proteindaten aus dem Notebook in die multimodale Umgebung integriert.

Während der Präsentation

Im Verlauf seines Vortrags möchte der Mitarbeiter nun seine Forschungsergebnisse anhand eines der Proteine den Zuschauern präsentieren. Mit einer Freihandgeste zieht der Mitarbeiter das Vorschaubild des entsprechenden Proteins aus dem Daten-Explorer der Powerwall auf die freie Fläche der Powerwall. Anschließend wird an dieser Stelle eine MegaMol-Ansicht erstellt, in der das Protein dargestellt wird.

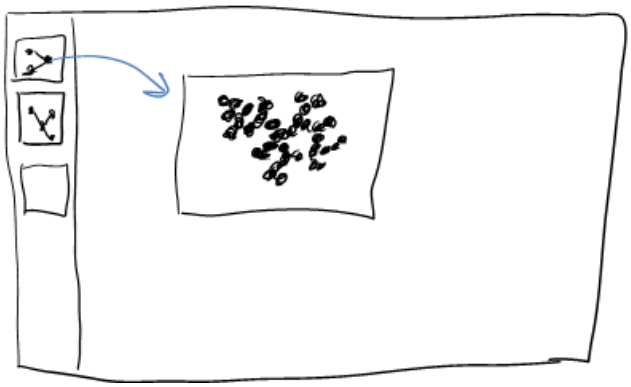


Abbildung 49: Aus dem Daten-Explorer wird eine MegaMol-Ansicht auf der Powerwall geöffnet.

Der wissenschaftliche Mitarbeiter wird während der Präsentation die Darstellung des Proteins manipulieren. Um stets eine Referenzansicht (R) mit der Ausgangsdarstellung des Proteins zur Verfügung zu haben, dupliziert der wissenschaftliche Mitarbeiter die MegaMol-Ansicht des Proteins mit einer Geste. Nun werden zwei identische MegaMol-Ansichten dargestellt (R und A in Abbildung 50).

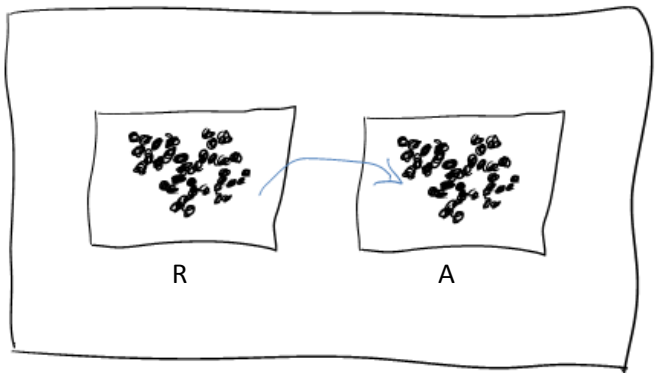


Abbildung 50: Mit einer Geste wird die MegaMol-Ansicht dupliziert.

Damit die Referenzansicht nicht zu viel Platz der Powerwall in Anspruch nimmt, positioniert der wissenschaftliche Mitarbeiter die Referenzansicht in der linken oberen Ecke der Powerwall. Hier kann sie gleichzeitig von allen Kollegen in Plenum gesehen werden. Anschließend vergrößert der Mitarbeiter die zweite MegaMol-Ansicht mit einer weiteren Geste, sodass diese den größten Teil der Powerwall belegt. Nun kann der wissenschaftliche Mitarbeiter seine Erkenntnisse in dieser Ansicht (A) darstellen, während die Referenzansicht (R) oben links noch zu sehen ist (siehe Abbildung 51).

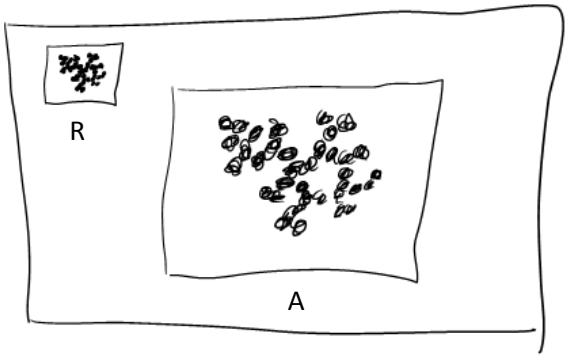


Abbildung 51: Mit Gesten zum Verschieben und Zoomen werden die Ansichten positioniert.

Um bestimmte Strukturen des Proteins hervorzuheben, markiert der wissenschaftliche Mitarbeiter mit einer Auswahlgeste diese Strukturen und färbt sie mit einer weiteren Geste in einer auffälligen Farbe ein. Mithilfe von weiteren Gesten wählt der Mitarbeiter für einige besonders wichtige Strukturen eine andere geometrische Darstellungsform aus.

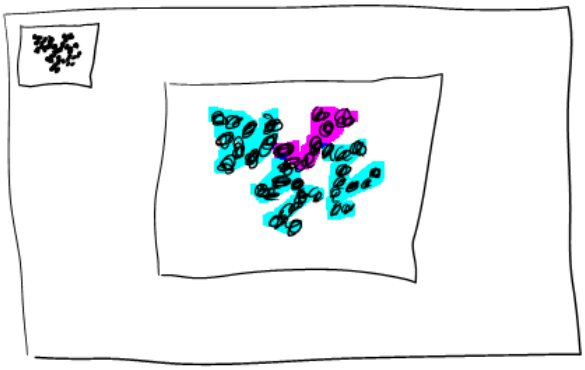


Abbildung 52: Mit verschiedenen Gesten wird die Darstellung des Proteins beeinflusst.

Nachdem der wissenschaftliche Mitarbeiter in der momentanen Ansicht alle Besonderheiten des Proteins präsentiert hat, möchte er weitere Details in einer anderen Darstellung hervorheben. Hierzu ändert der Mitarbeiter die Gesamtdarstellung des Proteins innerhalb der MegaMol-Ansicht mit einer weiteren Geste.

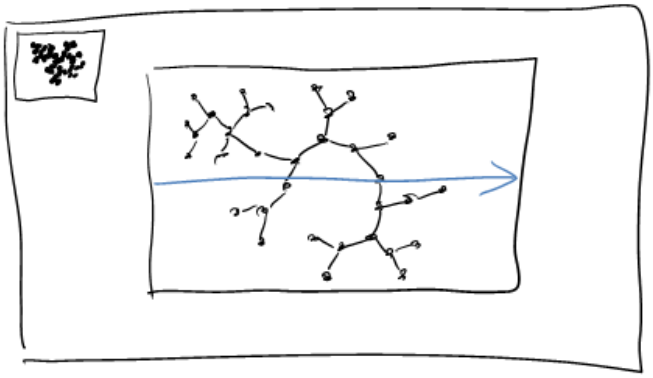


Abbildung 53: Mit einer Geste wird die Darstellung des gesamten Proteins geändert.

Mithilfe weiterer Gesten, die der Mitarbeiter mit einer oder beiden Händen ausführt, manipuliert der wissenschaftliche Mitarbeiter die Position der 3D-Kamera in der MegaMol-Ansicht. Auf diese Weise ändert sich der dargestellte Ausschnitt des Proteins innerhalb der MegaMol-Ansicht. Ebenso führt der Mitarbeiter Zoomgesten aus, um bestimmte Merkmale des Proteins im Detail dem Publikum präsentieren zu können.

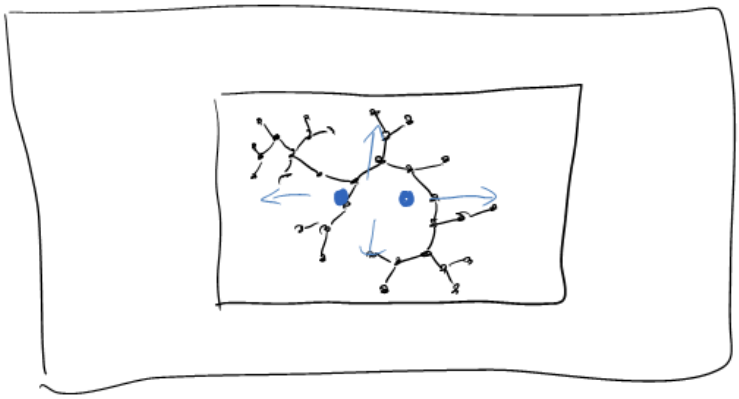


Abbildung 54: Mit Greif- und Zoomgesten wird der Betrachtungswinkel des Proteins beeinflusst.

Während der Diskussion, die sich an den Vortrag des wissenschaftlichen Mitarbeiters anschließt, generiert der Mitarbeiter eine Ansicht des Proteins, die er gerne in einem Paper zu seinen Forschungsergebnissen veröffentlichen möchte. Mit einer Geste markiert der Mitarbeiter einen Bereich der MegaMol-Ansicht und erstellt mit einer weiteren Geste einen Screenshot dieser Darstellung. Den Screenshot speichert der wissenschaftliche Mitarbeiter mithilfe einer weiteren Geste und dem Notebook-Symbol auf der Powerwall auf seinem Notebook. Der Screenshot steht dem Mitarbeiter nach Beenden der Präsentation weiterhin zur Verfügung.

5 Konzept

In diesem Kapitel wird das Konzept zur Gesteninteraktion mit powerwallbasierten Visualisierungen vorgestellt. Zunächst werden grundsätzliche Faktoren des Interaktionskonzepts vorgestellt, die nicht von der Umsetzung der einzelnen Gesten abhängen. Anschließend werden drei Formen der Gesteninteraktion mit Powerwall-Visualisierungen präsentiert und deren Vorteile, Nachteile und technische Schwierigkeiten diskutiert. Zuletzt wird die Einbettung dieses Interaktionskonzept in eine multimodale Umgebung umrissen.

5.1 Grundlegende Ideen

Dieser Abschnitt beschreibt grundlegende Ideen zur Interaktion mit Powerwall-Visualisierungen. Diese Ideen umfassen die Erkennung der Gesten mittels eines Sensors, die Unterteilung des Interaktionsbereiches in zwei Teilbereiche, die Erkennung des Benutzers, das Clutching und NUI-Prinzipien.

5.1.1 Gestenerkennung

Es wird ein System zur Erfassung von Gesten verwendet, das neben dem eigentlichen Sensor auf keine Hilfsmittel wie beispielsweise Marker oder eine 3D-Maus angewiesen ist (siehe Kapitel 2.3). Die Verwendung eines solchen Sensors ermöglicht den direkten Einstieg in ein Interaktionsszenario, ohne Vorbereitungen wie das Anlegen eines mit Markern versehenen Anzugs.

Zusätzlich muss das Gestenerkennungssystem erweiterbar sein. Das bedeutet, dass beispielsweise durch den Einsatz mehrerer gleichartiger Sensoren der Erkennungsbereich für Gesteninteraktionen vergrößert werden kann.

5.1.2 Interaktionsbereiche

Als Anwendung des Visual Information Seeking Mantras von Ben Shneiderman „Overview first, zoom and filter, then details on-demand“ (Shneiderman, User Interface Design, 2002) wird der Erkennungsbereich des Sensors in zwei Interaktionsbereiche mit unterschiedlicher Distanz zum Sensor unterteilt (siehe Abbildung 55):

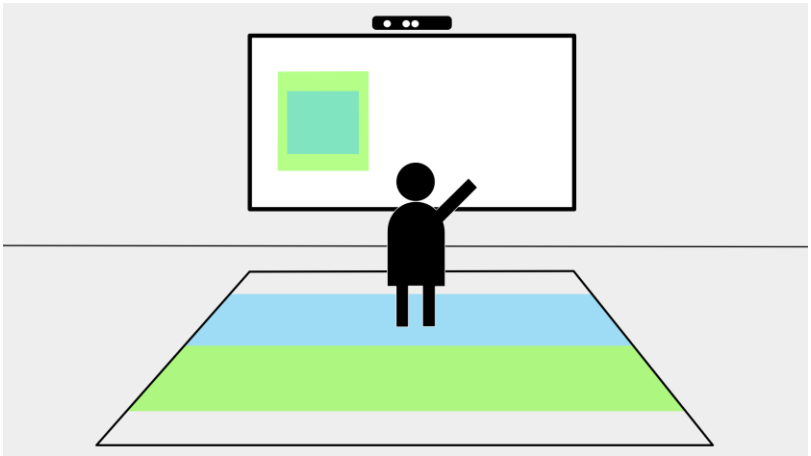


Abbildung 55: Interaktionsbereiche im Erkennungsbereich der Kinect (schwarzes Trapez). Im Fern-Interaktionsbereich (grün) können Übersichtsaufgaben durchgeführt werden, im Nah-Interaktionsbereich (blau) können Daten innerhalb der Ansichten manipuliert werden.

In dem vom Sensor weiter entfernten Bereich (in Abbildung 55 grün markiert) steht der Benutzer so weit von der Powerwall entfernt, dass er einen Überblick über die gesamte Projektionsfläche oder zumindest einen großen Bereich der Projektionsfläche hat. In diesem Fern-Interaktionsbereich hat

der Benutzer die Möglichkeit Übersichtsaufgaben mittels Gesten durchzuführen. Hierzu zählen beispielsweise das Verschieben oder Skalieren von Powerwall-Visualisierungen, das Erstellen neuer Visualisierungen oder das Erstellen von Verbindungen zwischen bestehenden Visualisierungen. Diese und weitere Standardgesten können in allen Anwendungsszenarien durchgeführt werden. Alle Standardgesten für Übersichtsaufgaben werden in Kapitel 5.2 vorgestellt.

Bewegt sich der Benutzer näher auf die Powerwall zu, betritt er den Nah-Interaktionsbereich (in Abbildung 55 blau markiert). Aufgrund der geringeren Distanz zur Powerwall kann der Benutzer nicht mehr die gesamte Projektionsfläche überblicken. Sein Fokus liegt vielmehr auf den Inhalten in dem Teilbereich der Powerwall, auf den er sich zubewegt hat. Nun kann der Benutzer Detailaufgaben innerhalb einer Powerwall-Visualisierung, wie beispielsweise die Manipulation eines Diagramms innerhalb einer Visualisierung, ausführen. Interaktionen im Nah-Interaktionsbereich sind Kontext- und Anwendungsspezifisch. Jedes Anwendungsszenario stellt hier Gesteninteraktionen für die jeweiligen Visualisierungen zur Verfügung. In Kapitel 5.2 werden daher nur exemplarisch Interaktionen und zugehörige Gesten für den Nah-Interaktionsbereich vorgestellt.

5.1.3 Schatteninterface

Um dem Benutzer die Orientierung auf der Powerwall zu erleichtern wird dessen vom Gesteninteraktionssystem erkannter Umriss als Schatten auf der Powerwall dargestellt (vgl. (Shoemaker, Tsukitani, Kitamura, & Booth, 2010)). Dieser Schatten ist halbtransparent, sodass die Powerwall-Visualisierungen dahinter immer erkennbar sind. Da der Schatten alle Bewegungen des Benutzers wie ein Spiegelbild darstellt, weiß der Benutzer immer, welche Punkte er auf der Powerwall mit seinen Händen erreichen und mit welchen Visualisierungen er von seinem jetzigen Standpunkt aus interagieren kann.

Gleichzeitig dient das Schatteninterface als Menü-Schnittstelle. Während der Benutzer mit einer Powerwall-Visualisierung interagiert, erscheinen an bestimmten Punkten innerhalb und um das Schatteninterface Menü-Objekte. Dadurch wird gewährleistet, dass Aktionen, die durch das Menü ausgelöst werden können, in ständiger Reichweite zum Benutzer sind.
Beispielsweise erscheint ein Papierkorb in Hüfthöhe rechts vom Benutzer. Verschiebt der Benutzer eine Powerwall-Visualisierung auf diesen Papierkorb, wird diese von der Powerwall entfernt. Auf der gegenüberliegenden Seite des Schatteninterfaces erscheint eine Festplatte. Verschiebt der Benutzer eine Visualisierung auf diese Festplatte, können die Daten der Visualisierung gespeichert werden.

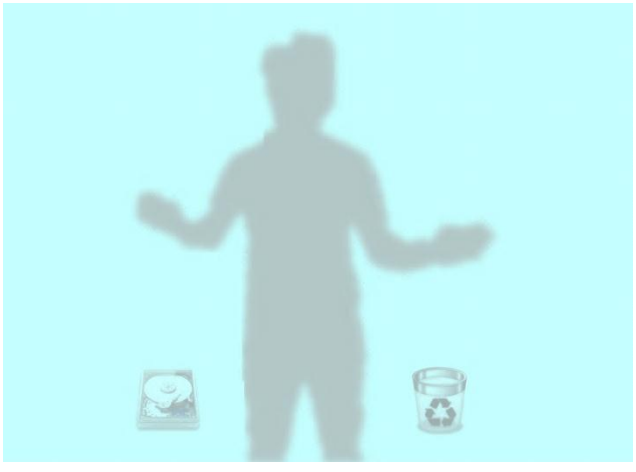


Abbildung 56: Spiegelbildliche Darstellung des Benutzers als Schatten mit Interaktionsobjekten (Festplatte und Papierkorb).

5.1.4 Benutzererkennung

Es kann nicht davon ausgegangen werden, dass sich zu jedem Zeitpunkt ausschließlich ein Benutzer des Powerwall-Interaktionssystems im Erkennungsbereich befindet. Beispielsweise könnten in einem Szenario zur Datenanalyse neben dem Benutzer weitere Personen die Powerwall-Visualisierungen betrachten und mit dem Benutzer über die dargestellten Daten diskutieren.
Daher muss unterschieden werden können, welche Personen als Benutzer Gesteninteraktionen zur Manipulation der Powerwall-Visualisierungen durchführen können und welche anderen Personen keine Interaktionen durchführen können. Für die Erkennung des Benutzers können verschiedene Strategien angewendet werden. Die im Folgenden aufgeführten Strategien sind denkbar: Kennzeichnung des Benutzers durch ein optisches Merkmal, Verfolgung der zuerst erkannten Person, Wahl der Person mit kürzestem Abstand zum Sensor, Aktivierung mittels einer Pose.

- **Optisches Merkmal**
  Diese Strategie ähnelt der Bewegungserkennung durch mehrere Kameras mittels Marker. Die Person, welche mit den Powerwall-Visualisierungen interagieren möchte, wird durch ein eindeutiges Merkmal ausgezeichnet. Dies kann beispielsweise ein rotes T-Shirt oder eine blaue Mütze sein. Ausschließlich diejenige Person, welche das farbige Merkmal trägt, kann mit der Powerwall in Interaktion treten.
  Ein Nachteil dieser Strategie ist, dass das optische Merkmal an eine andere Person abgegeben werden muss, wenn diese die Kontrolle über die Interaktionen übernehmen will.

- **Erste Person**
  Bei dieser Strategie wird diejenige Person zum Benutzer, die als erstes in den Erkennungsbereich des Sensors tritt. Solange sich diese Person innerhalb des Erkennungsbereichs befindet, kann ausschließlich sie als Benutzer mit den Powerwall-Visualisierungen interagieren. Verlässt die erste Person den Erkennungsbereich, wird die Person zum Benutzer, welche als zweites den Erkennungsbereich betreten hat.
  Nachteilig ist hierbei, dass die Kontrolle nur dann übergeben werden kann, wenn derjenige, der die Kontrolle innehat, den Erkennungsbereich verlässt.

- **Kürzester Abstand**
  Diese Strategie ermöglicht die Kontrolle zu übergeben und selbst dabei im Erkennungsbereich des Sensors zu bleiben. Es agiert immer diejenige Person als Benutzer, die den kürzesten Abstand zum Sensor hat.
  Allerdings müssen sich die anderen Personen, die sich im Erkennungsbereich aufhalten, anpassen: Soll eine Interaktion im Fern-Interaktionsbereich durchgeführt werden, darf keine Person näher am Sensor stehen als der Benutzer innerhalb des Fern-Interaktionsbereichs.

- **Posenaktivierung**
  Bei dieser Strategie bekommt immer diejenige Person die Kontrolle über die Powerwall-Visualisierungen übertragen, die zuvor eine bestimmte Pose eingenommen hat. Dabei muss eine Pose gewählt werden, die nicht zufällig oder versehentlich von Personen im Erkennungsbereich des Sensors eingenommen wird. Nachteil dieser Strategie ist die Verdeckung des Benutzers durch andere Personen innerhalb des Erkennungsbereiches. Die Verdeckung kann die Gestenerkennung durch den Sensor erschweren oder gar verhindern.

In einer Interaktionsumgebung, die dieses Konzept umsetzt, ist es möglich zwischen diesen Erkennungsstrategien zu wählen.

5.1.5 Clutching

Befinden sich mehrere Personen im Erkennungsbereich des Kinect Sensors, ist davon auszugehen, dass der Benutzer mithilfe von Gesten den anderen Personen Erklärungen zu den Powerwall-Visualisierungen liefert. Diese Bewegungen und Gesten dürfen nicht als Gesten zur Interaktion mit Powerwall-Visualisierungen interpretiert werden. In Kapitel 2.2 wurde dieses „Live-Mic"-Problem bereits beschrieben. Eine Lösung für dieses Problem ist das sogenannte „Clutching" (Wigdor & Wixon, 2011): Erst nachdem eine bestimmte Geste ausgeführt wurde, kann die eigentliche Interaktion beginnen.

Im Fern-Interaktionsbereich erscheint in Hüfthöhe des Schatteninterfaces eine rote Linie. Diese Linie erstreckt sich über die gesamte Breite der Powerwall. Solange diese Linie rot eingefärbt ist, ist keine Interaktion mit den Powerwall-Visualisierungen möglich. Der Benutzer kann beispielsweise mit Zeigegesten anderen anwesenden Personen Erklärungen zu den Visualisierungen liefern ohne unbeabsichtigt eine Interaktion mit einer der Visualisierungen auszulösen.
Hebt der Benutzer beide Hände über Hüfthöhe hinaus, durchqueren die Hände seines Schattens diese rote Linie. Dabei ändert sich die Farbe der Linie nach grün. Der Benutzer hat die Interaktionserkennung gestartet. Nun kann er mit den Powerwall-Visualisierungen interagieren. Beginnt der Benutzer innerhalb von fünf Sekunden keine Interaktion, wechselt die Farbe der Linie zurück nach rot. Der Benutzer kann dann erst wieder eine Interaktion starten, wenn seine Hände die Linie erneut von unten nach oben durchquert haben.

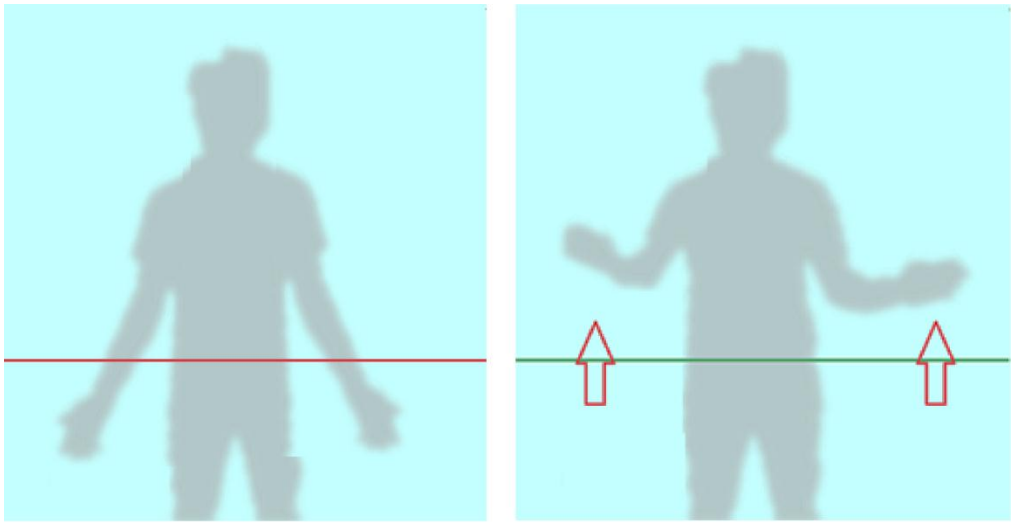


Abbildung 57: Schatteninterface ohne gestartete Interaktionserkennung. Bewegungen des Benutzers werden nicht als Gesteninteraktionen interpretiert (links). Rechts hat der Benutzer beide Hände über die Linie bewegt und kann Interaktionen auslösen.

Im Nah-Interaktionsbereich signalisiert das Schließen der Hand den Start einer Interaktion. Erkennt der Kinect-Sensor eine geöffnete Hand, wird auf der Handfläche des Schatteninterfaces ein farbiger Kreis dargestellt. Die Kreisfläche ist transparent, solange die Hand nicht geschlossen ist. Schließt der Benutzer die Hand, wird die Kreisfläche eingefärbt. Sofern an der Position der Hand des Schatteninterfaces auf der Powerwall eine Visualisierung dargestellt ist, die eine Interaktion im Nah-Interaktionsbereich erlaubt, wird diese Interaktion ausgelöst. Das Öffnen der Hand beendet die Interaktion und der Kreis der Hand wird wieder mit transparenter Füllung dargestellt.

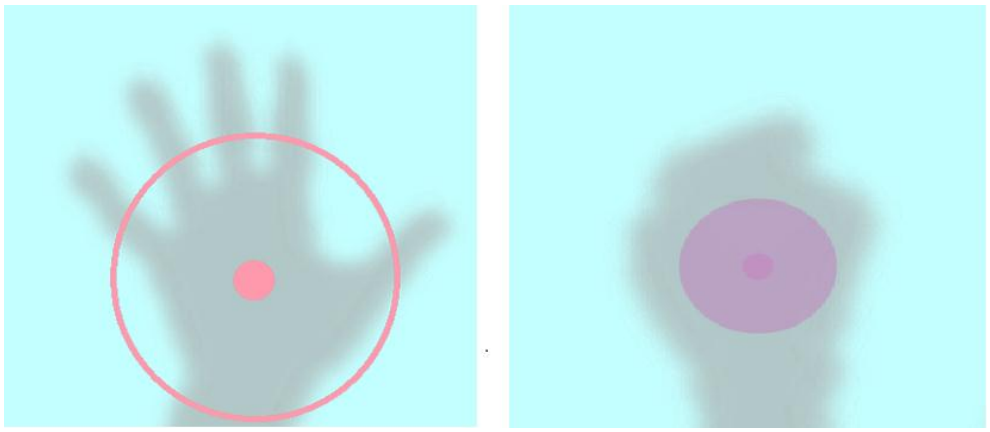


Abbildung 58: Im Nah-Interaktionsbereich erkannte geöffnete Hand (links). Rechts wurde eine Greifgeste erkannt und der Benutzer kann eine Interaktion durchführen.

5.1.6 Natürliche Interaktion

Ziel dieses Lösungskonzeptes ist es ein Natural User Interface für die Interaktion mit Powerwall-Visualisierungen zu erstellen. Wie in Kapitel 2.6 beschrieben sollen NUIs sowohl erfahrenen als auch unerfahrenen Benutzern eine leichte Interaktion mit dem betreffenden System ermöglichen.
Um dieses Ziel zu erreichen sind neben den intuitiven Interaktionen, die in Kapitel 5.2 beschrieben werden, Hilfestellungen für die Interaktion vorgesehen. Hierzu zählen

- das **Schatteninterface**, welches dem Benutzer zeigt, in welchem Bewegungsrahmen er momentan mit Visualisierungen interagieren kann
- **Markierungen innerhalb des Schatteninterfaces**, die beispielsweise die Position der Hände darstellen
- die visuellen Darstellungen der „**Clutches**“, die dem Benutzer zeigen, wann eine Interaktion mit den Visualisierung angestoßen werden kann
- die **Markierungen der erkannten Hände** im Nah-Interaktionsbereich mit dem farblichen Hinweis, ob die Hand als „offen“ oder „geschlossen“ erkannt wurde
- **Animationen** auf den Menü-Items, die innerhalb und um dem Schatteninterface platziert werden

Mit der Erfahrung des Benutzers werden diese Hilfestellungen reduziert. Somit soll sich ein erfahrener Benutzer voll und ganz auf die Interaktionen und deren Ergebnisse konzentrieren können, während ein unerfahrener Benutzer durch die Hilfestellungen an die Interaktion mit Powerwall-Visualisierungen herangeführt wird.
Je nach Erfahrungsstand des Benutzers können die Hilfestellungen halbtransparent dargestellt oder nach und nach weggelassen werden.

5.2 Interaktionskonzept

In diesem Abschnitt werden zunächst die einzelnen Interaktionen, die in den Nah- und Fern-Interaktionsbereichen ausgeführt werden können, beschrieben. Anschließend werden drei Paradigmen vorgestellt, nach denen die einzelnen Interaktionen ausgelöst und beendet werden können. Am Ende dieses Abschnitts werden die Vor- und Nachteile dieser Varianten diskutiert.

5.2.1 Interaktionen

Um zwischen den Auswirkungen der einzelnen Interaktionen im Nah- und Fern-Interaktionsbereich unterscheiden zu können, werden in diesem Abschnitt folgende Begriffe verwendet:

Ansicht: Übergeordnete Darstellungseinheit, in der Daten visualisiert werden können. Entspricht einem Fenster im Desktop-Interaktionsparadigma.

Visualisierung: Bezeichnet die grafischen Darstellungen von Daten innerhalb einer Ansicht. Dies kann beispielsweise ein Diagramm oder ein Bild sein.

Powerwall-Visualisierung: Verbund aus einer Ansicht mit einer oder mehreren Visualisierungen, die auf einer Powerwall dargestellt und mithilfe dieses Interaktionskonzepts manipuliert werden können.

In den Abbildungen zu den jeweiligen Interaktionen werden diese Piktogramme verwendet:

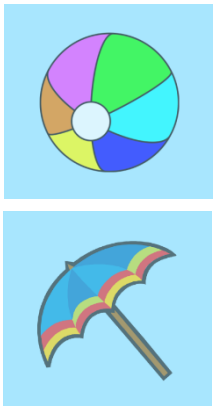
	Positionsmarkierung der Hand eines Benutzers auf der Powerwall. Es wurde noch keine Interaktion ausgelöst oder die Interaktion wurde bereits beendet.
	Positionsmarkierung der Hand eines Benutzers auf der Powerwall, mit der gerade eine Interaktion durchgeführt wird.
	Bewegung einer Hand des Benutzers
	Darstellung zweier verschiedener Beispielvisualisierungen innerhalb jeweils einer quadratisch dargestellten Ansicht.
	Zur „Momentaufnahme“ der Abbildung zeitlich vorgelagerte Position einer Ansicht.

Tabelle 1: Piktogramme zur Darstellung der Interaktionen mit Powerwall-Visualisierungen.

Folgende Interaktionen mit Powerwall-Visualisierungen sind im Fern-Interaktionsbereich möglich: *Verschieben*, *Löschen*, *Skalieren*, *Duplizieren*, *Koppeln*, *Visualisierung ändern* und *Auswahl treffen*. Dabei beziehen sich diese Aktionen ausschließlich auf Ansichten. Dies bedeutet, dass aus dem Fern-Interaktionsbereich keine Manipulationen an der Visualisierung innerhalb einer Ansicht vorgenommen werden können. Die Interaktion *Inhalt manipulieren* kann im Nah-Interaktionsbereich ausgeführt werden.

Ansicht verschieben

Nach dem Auslösen dieser Interaktion durch eine beliebige Hand des Benutzers, kann die Position einer Ansicht durch die Bewegung des Armes auf der Anzeigefläche der Powerwall verändert werden (siehe Abbildung 59).

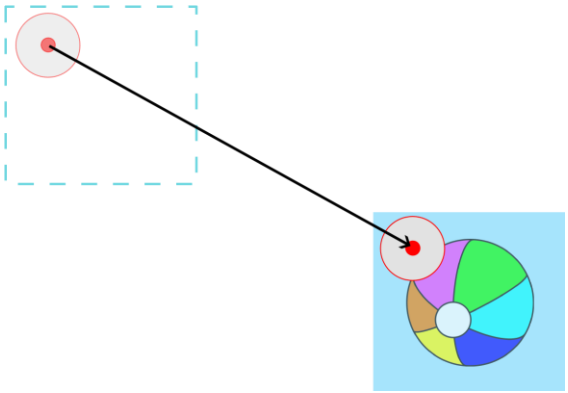


Abbildung 59: Beispiel der Interaktion "Ansicht verschieben" - mit der Bewegung in Pfeilrichtung wird die Ansicht von links oben nach rechts unten verschoben.

Ansicht löschen

Beim Verschieben (siehe „Ansicht verschieben“) wird die Ansicht über das Papierkorb Menü-Item des Schatteninterfaces positioniert (siehe Abbildung 60). Beim Beenden dieser Interaktion wird die Ansicht von der Powerwall gelöscht.

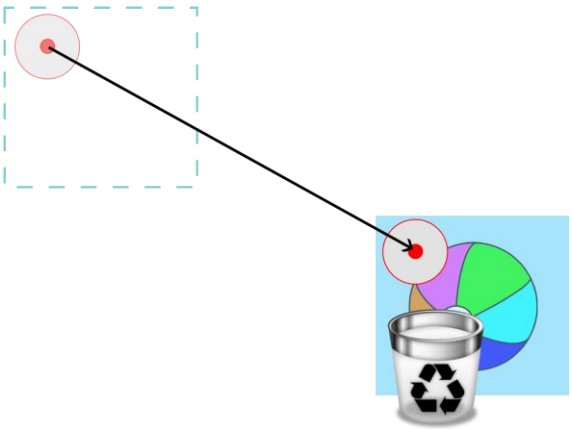


Abbildung 60: Interaktion "Ansicht löschen". Durch das Verschieben der Ansicht in den Papierkorb wird die Ansicht gelöscht.

Ansicht Skalieren

Diese Interaktion wird vom Benutzer ausgelöst, während sich beide Hände innerhalb der zu skalierenden Ansicht befinden (Abbildung 61 links). Anschließend können die Breite und die Höhe der Ansicht durch das Verändern des Abstands beider Hände in horizontaler bzw. vertikaler Richtung beeinflusst werden (Abbildung 61 rechts).

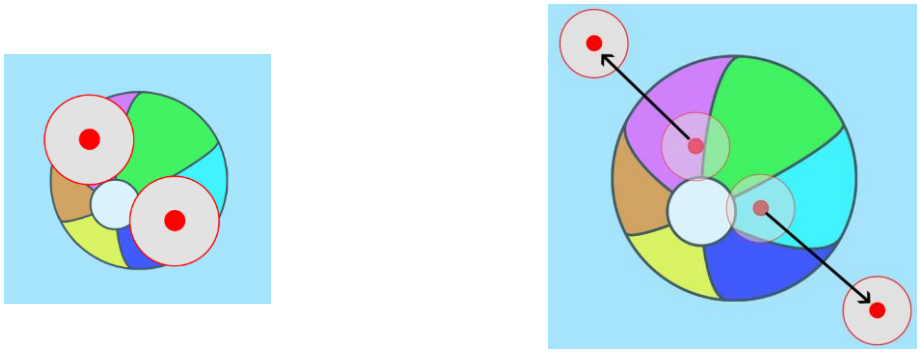


Abbildung 61: Interaktion "Ansicht skalieren". Auswahl der zu skalierenden Ansicht mit beiden Händen (links). Rechts wird die Ansicht durch Auseinanderbewegen der Hände vergrößert.

Ansicht duplizieren

Zum Erstellen neuer Ansichten können bestehende Ansichten dupliziert werden. Um diese Interaktion auszulösen, müssen sich beide Hände des Benutzers innerhalb der zu duplizierenden Ansicht befinden. Im Gegensatz zur Skalierungs-Interaktion müssen die Positionsmarkierungen beider Hände hierbei beim Auslösen der Interaktion dicht beieinander liegen (Abbildung 62 links). Nach dem Auslösen der Interaktion bewegt der Benutzer eine Hand aus der zu duplizierenden Ansicht heraus, während die andere innerhalb dieser Ansicht bleibt, und „reißt“ so eine kopierte Ansicht aus dem Original heraus. Nach Beendigung der Interaktion wird die Ansicht an der Position außerhalb dupliziert (Abbildung 62 rechts).

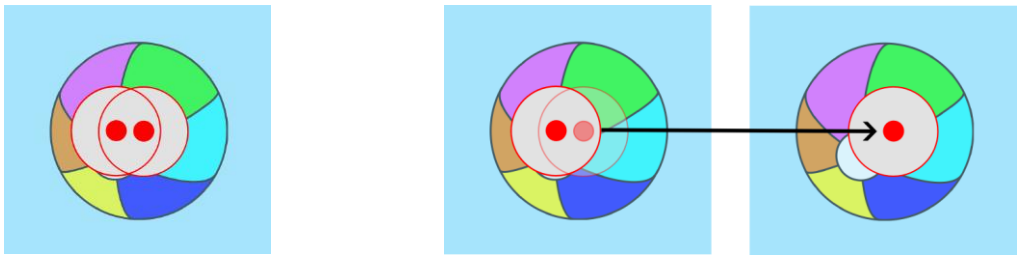


Abbildung 62: Interaktion "Ansicht duplizieren". Zu duplizierende Ansicht wird mit beiden Händen aktiviert (links) und anschließend das Duplikat zur Seite herausgezogen (rechts).

Ansichten koppeln

Ansichten können miteinander verbunden werden, sodass beispielsweise Interaktionen in allen gekoppelten Ansichten gleichzeitig angewendet werden können. Um diese Interaktion auszulösen müssen sich beide Hände des Benutzers über jeweils einer der beiden zu koppelnden Ansichten befinden. Nach dem Auslösen der Interaktion bewegt der Benutzer beide Hände aufeinander zu, bis die Hilfestellungen beider Hände sich auf der Powerwall überschneiden (Abbildung 63 oben) und verbindet mit dieser Bewegung die beiden Ansichten. Nach Beendigung dieser Interaktion wird eine Verbindung zwischen den beiden Ansichten dargestellt (Abbildung 63 unten).

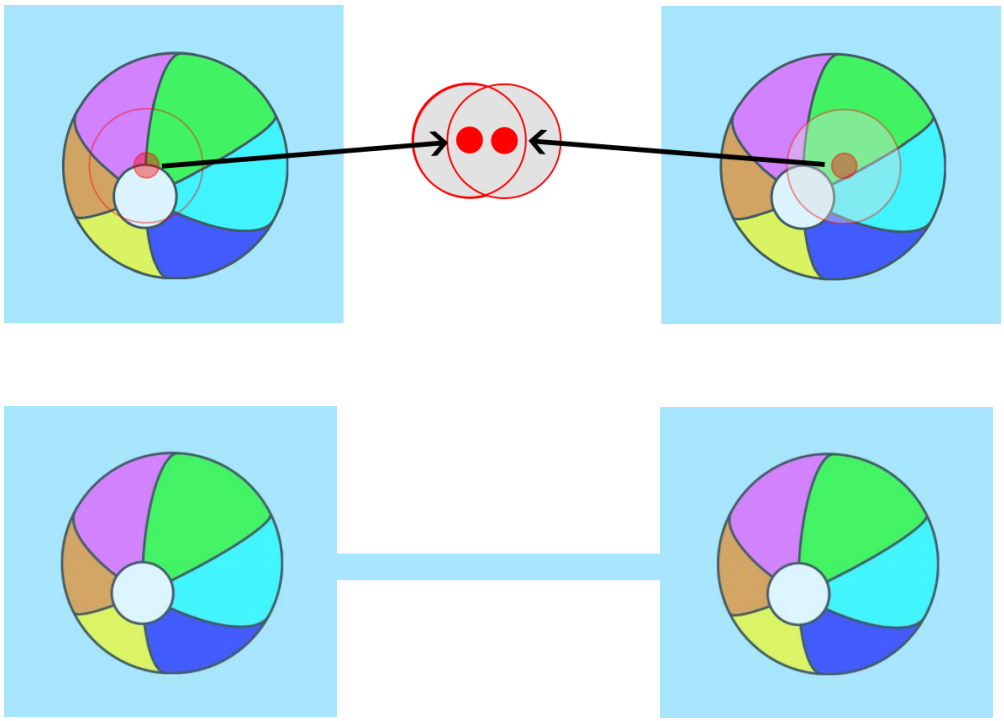


Abbildung 63: Interaktion "Ansichten koppeln". Oben werden die zu koppelnden Ansichten mit je einer Hand ausgewählt und die Hände zusammengeführt. Anschließend sind die beiden Ansichten miteinander verbunden (unten).

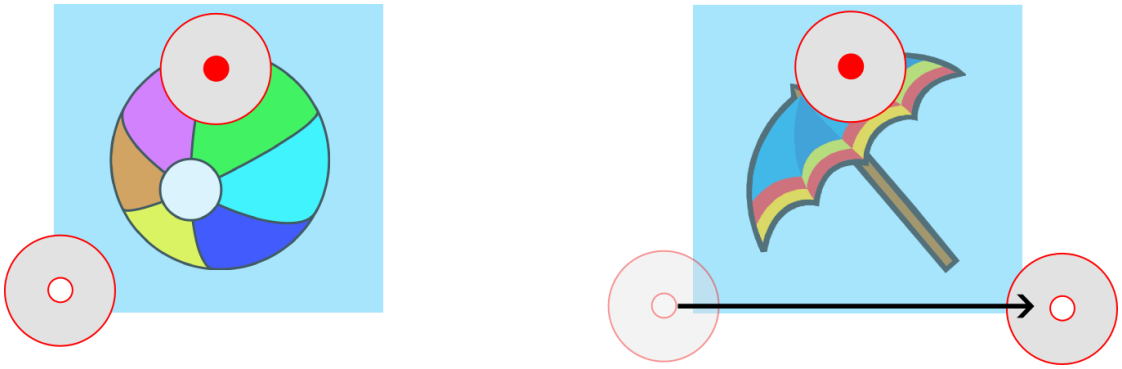


Abbildung 64: Interaktion "Visualisierung ändern". Links wird in der Ansicht eine Ball-Visualisierung dargestellt. Mit einer Wischgeste wird die dargestellte Visualisierung gewechselt (rechts).

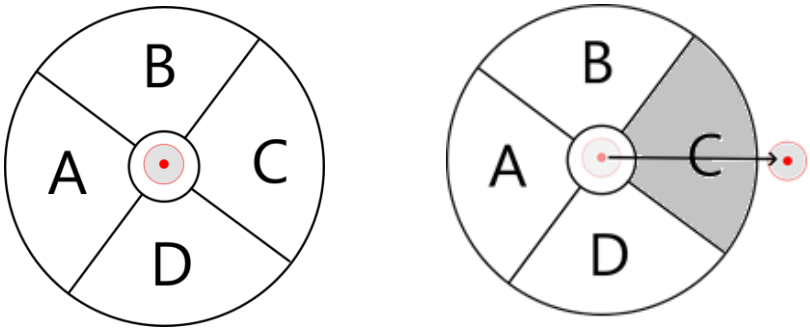


Abbildung 65: Interaktion "Auswahl treffen". In einem Radialmenü werden vier Elemente zur Auswahl dargestellt (links). Durch eine Bewegungsgeste in die entsprechende Richtung wählt der Benutzer ein Element aus (rechts).

Visualisierung ändern

Eine Ansicht kann verschiedene Visualisierungen darstellen. Um zwischen den Visualisierungen zu wechseln wählt der Benutzer mit einer Hand die Ansicht aus, in der die Visualisierung gewechselt werden soll und löst somit die Interaktion aus (siehe Abbildung 64, links). Anschließend durchquert der Benutzer die Ansicht der anderen Hand. Mit einer Animation, die der Bewegungsrichtung der zweiten Hand entspricht, wechselt die dargestellte Visualisierung der Ansicht (siehe Abbildung 64, rechts).

Auswahl treffen

Wenn der Benutzer im Programmverlauf auf der Powerwall eine Auswahl zu treffen hat, wird ihm diese Auswahl in Form eines Radialmenüs dargestellt. Um ein Element des Radialmenüs auszuwählen, bewegt der Benutzer eine Hand in das Zentrum des Radialmenüs (Abbildung 65, links) und durchquert anschließend das Element des Radialmenüs, welches er auswählen möchte (Abbildung 65 rechts).

Inhalt manipulieren

Die Interaktion „Inhalt manipulieren“ ist ein Beispiel für eine Interaktion mit Daten einer Visualisierung innerhalb einer Ansicht. Diese Interaktion wird aus dem Nah-Interaktionsbereich heraus durchgeführt. Die möglichen Interaktionen mit einer Visualisierung hängen von der jeweiligen Visualisierung selbst ab. Beispielweise können aus einem Diagramm, das innerhalb einer Ansicht dargestellt wird, Daten entfernt werden.

Nach dem Auslösen dieser Interaktion auf der Visualisierung innerhalb einer Ansicht kann der Benutzer die Darstellung der Visualisierung manipulieren. Wie in Abbildung 66 dargestellt, kann Beispielsweise ein Teil der Visualisierung (ein Segment des Balls) aus der Visualisierung extrahiert und entweder über das Menü-Item „Papierkorb“ gelöscht oder durch Bewegen in eine andere Visualisierung verschoben werden.

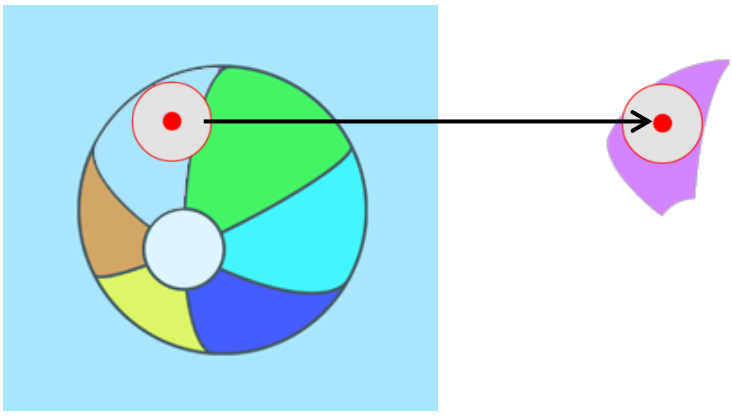


Abbildung 66: Durch eine Greifgeste im Nah-Interaktionsbereich kann der Inhalt einer Ansicht manipuliert werden.

5.2.2 Interaktionsbeginn

In diesem Abschnitt werden drei Verfahren vorgestellt, die zur Erkennung des Auslösens und des Beendens einer Interaktion herangezogen werden können. Die Verfahren sind: „Schließen und Öffnen der Hände“, „Zeitverzögerung“ und eine Kombination der beiden.

1. **Schließen und Öffnen der Hände**
   Wie im Abschnitt „Hand- und Fingertracking“ in Kapitel 2.4.3 beschrieben, ist es möglich mit Gesteninteraktionssystemen die Hände des Benutzers zu erkennen und zu evaluieren, ob die Hände geöffnet oder geschlossen sind. In diesem Verfahren zur Erkennung von Interaktionen werden die Zustände „Hand offen“ und „Hand geschlossen“ sowie der Übergang zwischen diesen beiden Zuständen in Form einer Greifgeste betrachtet:
   Schließt der Benutzer eine Hand, wird an der Stelle, an der die Mitte der Handfläche erkannt wurde, eine Interaktion ausgelöst. Öffnet der Benutzer die Hand wieder, so wird eine laufende Interaktion mit dieser Hand beendet.

2. **Dwell-Time** (Jacob R. , 1990)
   Im Gegensatz zum Verfahren „Schließen und Öffnen der Hände“ ist es bei diesem Verfahren nicht nötig, Hände und Finger zu erkennen. Mithilfe des Gestenerkennungssystems werden die Positionen der Hände des Benutzers ermittelt.
   Möchte der Benutzer eine Interaktion auslösen, positioniert er eine Hand an der Stelle der Powerwall, an der die Interaktion ausgelöst werden soll. Ruht die Hand an dieser Stelle für eine bestimmte Zeitspanne und bewegt sich nicht über eine gewisse Mindestdistanz, wird eine Interaktion an dieser Stelle ausgelöst. Um diese Interaktion wieder zu beenden, lässt der Benutzer die interagierende Hand wieder für eine bestimmte Zeitspanne an einer Position ruhen. Anschließend ist die Interaktion beendet.
   Während eine Hand an einer Position ruht, wird durch eine Animation angezeigt, wann die Interaktion ausgelöst oder wieder beendet wurde.

3. **Kombination**
   In diesem Verfahren werden die beiden Verfahren „Schließen und Öffnen der Hände“ und „Dwell-Time“ kombiniert:
   Im Fern-Interaktionsbereich werden Interaktionen durch das „Zeitverzögerung“-Verfahren ausgelöst und beendet. Im Nahbereich hingegen findet das Verfahren „Schließen und Öffnen der Hände“ Anwendung.

5.3 Einbettung in eine multimodale Umgebung

Für eine geeignete Nutzung scheint es sinnvoll, die Interaktion mit Powerwall-Visualisierungen in eine multimodale Interaktionsumgebung einzubetten. In diesem Kapitel wird zunächst die Gestaltung dieser multimodalen Umgebung durch weitere Interaktionsmodalitäten, anschließend die Vernetzung der Interaktionsmodalitäten und zuletzt das Konzept von privaten und öffentlichen Bereichen vorgestellt.

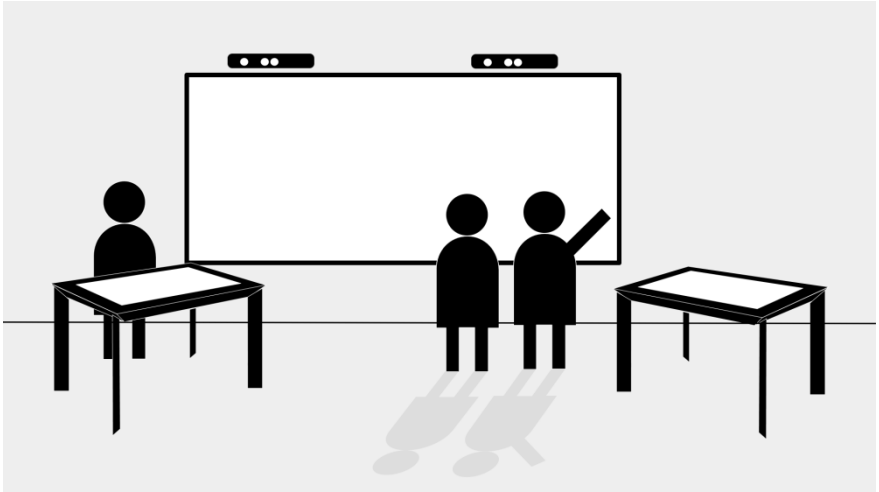


Abbildung 67: Piktogramm einer multimodalen Interaktionsumgebung. In dieser Umgebung ermöglichen Kamera-Sensoren eine Gesteninteraktion mit einer Powerwall. Zusätzlich werden Tabletop-Computer zur Interaktion mit den Daten verwendet.

5.3.1 Weitere Interaktionsmodalitäten

Neben der Powerwall als Zentrum der multimodalen Interaktionsumgebung, die von allen Personen innerhalb der Interaktionsumgebung betrachtet werden kann, kommen Multi-Touch-Tischcomputer zum Einsatz. Zusätzlich können Notebooks, Tablet-PCs oder Smartphones in die multimodale Interaktionsumgebung integriert werden. In Kombination mit diesen Modalitäten entsteht ein System ubiquitärer Interaktionsmöglichkeiten (Weiser, 1991).

Multi-Touch Tabletop-Computer

Ein Multi-Touch-Tischcomputer ermöglicht die Interaktion mit Visualisierungen durch mehrere Benutzer gleichzeitig. Werden ein oder mehrere Tischcomputer in Sichtweite der Powerwall aufgestellt, können Daten gleichzeitig auf der Powerwall betrachtet und auf einem der Tischcomputer bearbeitet werden. Alternativ kann die Powerwall durch einen Multi-Touch-Tischcomputer ferngesteuert werden. Darüber hinaus können Realweltobjekte – sogenannte „Tangible Objects“ – zur Interaktion mit dem Tischcomputer und Visualisierungen verwendet werden.

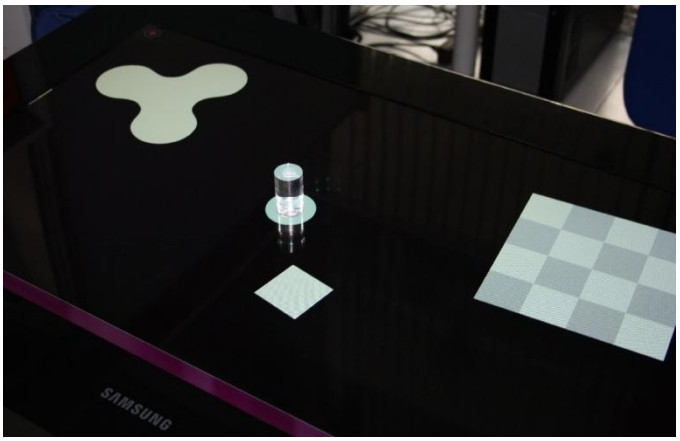


Abbildung 68: Ein Plexiglaszylinder wird als Tangible Objekt zur Interaktion erkannt.

Das Konzept zur Interaktion mit Multi-Touch-Tischcomputer im Rahmen einer multimodalen Interaktionsumgebung wird in (Püttmann, 2012) beschrieben.

Notebooks und Tablets

Die Powerwall in der multimodalen Interaktionsumgebung erlaubt die Betrachtung von Visualisierungen durch mehrere Personen gleichzeitig. Multi-Touch-Tischcomputer ermöglichen die Manipulation von Visualisierungen durch mehrere Benutzer zur selben Zeit. Zusätzlich werden Notebooks und Tablet-PCs in diese Interaktionsumgebung integriert. Notebooks und Tablets sind für die Benutzung durch eine Person ausgelegt. Sie bieten Rückzugsmöglichkeiten für Benutzer aus der Gruppeninteraktion. Einzelne Benutzer können auf einem Notebook oder Tablet-PC Ausschnitte der Visualisierungen auf der Powerwall darstellen und für sich betrachten. Alternativ kann die Powerwall von einem Notebook oder Tablet aus ferngesteuert werden.
Notebooks und Tablets können mit Benutzern der multimodalen Interaktionsumgebung assoziiert werden. Das Schatteninterface von Benutzern, die mit einem Notebook oder Tablet assoziiert wurden, verfügt über einen weiteren Menü-Eintrag, das dieses tragbare Gerät symbolisiert. Über diesen Menü-Eintrag kann der Benutzer beispielsweise Daten oder Visualisierungen auf das Notebook oder Tablet übertragen, indem er die entsprechende Ansicht oder Visualisierung auf dem Menü-Eintrag ablegt. Die übertragenen Daten können auf dem tragbaren Gerät aus der multimodalen Interaktionsumgebung herausgenommen, manipuliert und wieder in die Interaktionsumgebung integriert werden.

5.3.2 Vernetzung

Um eine Verbindung zwischen der Powerwall, den Tischcomputern, Notebooks und Tablet-PCs zu ermöglichen, verfügt die multimodale Interaktionsumgebung über ein Kommunikationsframework. Über dieses Framework werden Daten zur Visualisierung zur Verfügung gestellt und auf jedes Gerät übertragen, das diese Daten darstellt. Ebenso werden alle Interkationen mit Visualisierungen zwischen den Geräten der multimodalen Interaktionsumgebung kommuniziert, sodass sichergestellt ist, dass alle Daten auf allen Geräten synchron dargestellt werden. Gleichzeitig werden alle Interaktionen protokolliert, um Auswertungen zu ermöglichen und durchgeführte Interkationen nachvollziehen zu können. Die Architektur des Kommunikationsframeworks wird in Kapitel 6 beschrieben.

5.3.3 Private und öffentliche Bereiche

Mit der Powerwall und den Multi-Touch-Tischcomputern unterstützt die multimodale Interaktionsumgebung die Arbeit von Gruppen durch das Konzept von privaten und öffentlichen Bereichen (Scott, 2003). Mehrere Personen können gleichzeitig Visualisierungen betrachten und manipulieren. Hierfür steht ein öffentlicher Bereich („public space“) zur Verfügung. Änderungen, die an einer Visualisierung im öffentlichen Bereich vorgenommen werden, werden automatisch auf alle anderen Geräte, die sich in der Interaktionsumgebung befinden, übertragen und dort angewendet. Somit wird gewährleistet, dass die Darstellung der Visualisierungen im öffentlichen Bereich auf allen Geräten synchron ist.

„Private Spaces“ hingegen erlauben Visualisierungen auf einem einzelnen Gerät der multimodalen Interaktionsumgebung zu isolieren, getrennt zu betrachten und unabhängig zu bearbeiten. Hierzu können Visualisierungen aus dem öffentlichen Bereich in den privaten Bereich eines Notebooks, Tablet-PCs oder Tischcomputers übertragen werden. Anschließend werden Manipulationen an diesen Visualisierungen nicht mehr auf die anderen Geräte der Interaktionsumgebung übertragen. Nach der Bearbeitung in einem privaten Bereich können Visualisierungen wieder in den öffentlichen Bereich zurücktransferiert werden.

6 Architektur

In diesem Kapitel werden die Architektur des Kommunikationsframeworks und weiterer Komponenten beschrieben, die als Grundlage für den Softwareprototyp dienen. Zunächst werden Anforderungen an die Architektur eines Kommunikationsframeworks beschrieben und anschließend daraus grundsätzliche Überlegungen für den Softwareentwurf abgeleitet. Daraufhin werden die einzelnen Komponenten aus dem Entwurf, deren Aufbau und Zusammenspiel dokumentiert.

Die Architektur baut dabei auf dem Standard des Client-Server-Prinzips auf (Niemann, 1995). Wo nötig wurden Anpassungen vorgenommen, die später im Kapitel beschrieben werden. Es werden folgende Begriffe in diesem Kapitel verwendet:

Kommunikationsframework: Infrastruktur, über die Daten aus beliebigen Quellen an beliebig viele teilnehmende Geräte und Clients verteilt und Daten ausgetauscht werden. Hierzu zählen die Serverkomponente, die Web Services zum Datenaustausch und die Client-Schnittstellen.

Serverkomponente: Hauptprogramm des Kommunikationsframeworks, das die Kommunikation zwischen allen Komponenten des Frameworks koordiniert.

Datenprovider: Ein Programm, das anderen Komponenten des Kommunikationsframeworks über den Server Daten zur Verfügung stellt.

(Visualisierungs-)Client: Ein Programm, das Daten von Datenprovidern über die Serverkomponente abrufen und Benutzern zur Interaktion darstellt.

(Interaktions-)System: Ein System ist die Kombination aus dem Kommunikationsframework und für den speziellen Fall implementierten Clients. Bestimmte Clients stellen die im System verwendeten Daten zur Verfügung, die über das Kommunikationsframework zu anderen Clients übertragen werden, die diese Daten manipulieren und anzeigen können.

6.1 Anforderungen an die Architektur

In der Aufgabenstellung zu dieser Arbeit (Kapitel 3) wird gefordert, dass mit in der im Rahmen dieser Diplomarbeit erstellten Software verschiedene Visualisierungstypen dargestellt und mit den Visualisierungen interagiert werden können soll. Es sollen Mehrbenutzer-Interaktionen mit wissenschaftlichen und informationswissenschaftlichen Visualisierungen auf einer Powerwall und anderen Geräten möglich sein. Hieraus lassen sich folgende Anforderungen an die Architektur eines solchen Interaktionssystems ableiten: Datenunabhängigkeit, Erweiterbarkeit, Mehrgerätefähigkeit, Plattformunabhängigkeit, erweiterbare Visualisierungen und Datensynchronität. Diese Anforderungen werden im Folgenden genauer betrachtet:

- **Datenunabhängigkeit**:
  Um zu ermöglichen, dass Interaktionssysteme für verschiedene Visualisierungstypen und verschiedene Datengrundlagen erstellt werden können, soll eine spezielle Infrastruktur geschaffen werden. Diese Infrastruktur stellt ein Kommunikationsframework dar, das beliebige Daten verarbeiten kann. Es soll nicht nötig sein, beim Erstellen eines neuen Interaktionssystems diese Infrastruktur anpassen und neu kompilieren zu müssen.

- **Erweiterbarkeit**:
  Verschiedene Datengrundlagen, die von unterschiedlichen Interaktionssystemen verwendet werden, stellen eigene Anforderungen an die Kommunikation zwischen den beteiligten Geräten. Das Kommunikationsframework soll diese Anforderungen ohne weitere Anpassungen erfüllen können. Daher ist eine Erweiterbarkeit der Kommunikationsmöglichkeiten über das Kommunikationsframework erforderlich.

- **Mehrgerätefähigkeit**
  Ein Interaktionssystem darf nicht auf die Verwendung mit einer Powerwall beschränkt sein. In Zusammenarbeit mit der Diplomarbeit „Gestensteuerung für Tabletop-Computer-basierte Visualisierungen“ (Püttmann, 2012) werden Tabletop-Computer ebenfalls in ein Interaktionssystem mit der Powerwall integriert. Auch die Integration von einer beliebigen Anzahl an Notebooks und Tablets ist denkbar. Dies muss bei der Architektur des Kommunikationsframeworks berücksichtigt werden.

- **Plattformunabhängigkeit**
  Wenn verschiedene Gerätetypen in ein Interaktionssystem integriert werden, soll es möglich sein Client-Geräte unterschiedlicher Plattformen zu verwenden. Die Kommunikation innerhalb des Systems soll unabhängig von den Plattformen der verwendeten Geräte sein.

- **Flexible Visualisierungen**
  Ermöglicht das Kommunikationsframework die Verwendung unterschiedlicher Datentypen, muss auch die Darstellung und Interaktion mit den Daten flexibel gestaltet werden. Es soll möglich sein für unterschiedliche Datentypen verschiedene Visualisierungen zu verwenden, die jeweils eigene Interaktionen mit den Daten ermöglichen.

- **Datensynchronität**
  Das Kommunikationsframework muss gewährleisten können, dass alle verbundenen Client-Geräte ständig über dieselben Daten verfügen. Ebenso müssen die Ergebnisse jeder Interaktion mit einer Visualisierung auf alle anderen Geräte übertragen werden können, damit diese ihrerseits die Darstellung der Daten anpassen können.

6.2 Grundsätzliche Entwurfsüberlegungen

Dieser Abschnitt beschreibt, wie die zuvor formulierten Anforderungen an die Architektur umgesetzt werden sollen. Abbildung 69 zeigt eine Entwurfsskizze, deren Bestandteile „Client-Server-Architektur“, „Message Broking“, „WebServices“, „Daten-Provider“ und „Plugins“ im Folgenden beschrieben werden. Gleichzeitig wird ein Bezug zu den zuvor formulierten Anforderungen hergestellt.

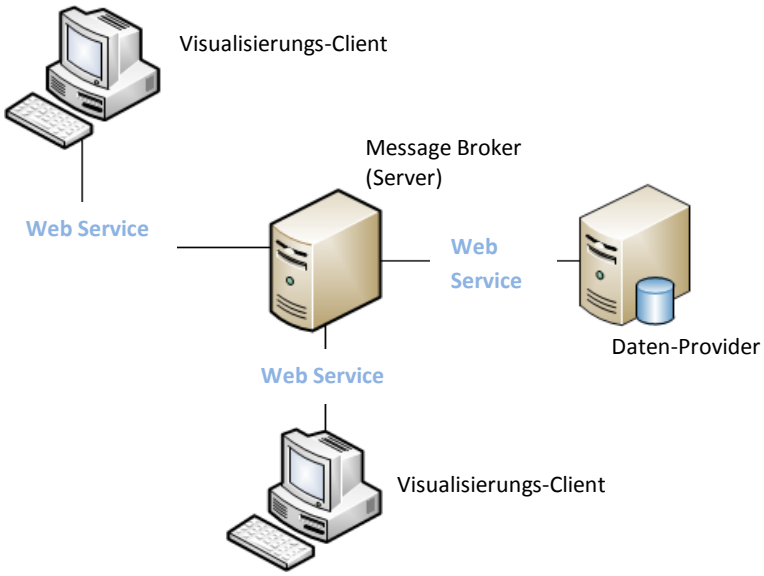


Abbildung 69: Entwurfsskizze für das Kommunikationsframework bestehend aus einem Server (mitte), einem Daten-Provider Client (rechts) und zwei Visualisierungs-Clients (oben und unten). Die Komponenten sind über ein Netzwerk verbunden und kommunizieren über Web Services.

- **Client-Server-Architektur**
  Durch einen Server (Abbildung 69 Mitte) als zentrale Stelle können alle Teilnehmer der Kommunikation im Netzwerk verwaltet werden, ohne dass die Clients voneinander Kenntnis haben müssen. Dabei sind verschiedene Clienttypen mit unterschiedlichen Aufgaben möglich (Mehrgerätefähigkeit).

- **Message Broking**
  Der Server (Abbildung 69 Mitte) fungiert als Verteilerstelle für alle Nachrichten, die innerhalb des Netzwerks ausgetauscht werden sollen. Dies garantiert die Datensynchronität, sodass alle verbundenen Clients alle Nachrichten erhalten.

- **Web Services**
  Alle Nachrichten werden zwischen den einzelnen Geräten im Netzwerk über Web Services ausgetauscht (Foggon, Maharry, Ullman, & Watson, 2003). Durch die Verwendung dieses standardisierten und offenen Prinzips, können Geräte unabhängig vom Betriebssystem in die Kommunikation integriert werden (Plattformunabhängigkeit). Gleichzeitig wird die Möglichkeit der Erweiterbarkeit geschaffen, indem ein allgemeines Nachrichtenformat verwendet wird, das im Bedarfsfall um weitere Nachrichtentypen erweitert werden kann.

- **Visualisierungs-Clients mit Plugins**
  Zur flexiblen Gestaltung der Visualisierungen können Plugins erstellt und auf die verschiedenen Visualisierungs-Clients verteilt werden. So können auch verschiedene Clients unterschiedliche Visualisierungen verwenden. Visualisierungen können durch die Plugins nachträglich erstellt und eingefügt werden, denn sie müssen lediglich diejenigen Datentypen kennen, die sie auch darstellen sollen.

- **Daten-Provider**
  Um eine Unabhängigkeit von den verwendeten Daten zu erreichen, wird eine Komponente entwickelt, die Daten aus einer beliebigen Quelle laden und über das Kommunikationsframework zur Verfügung stellen kann. Über den Server können Clients auf diese Daten zugreifen und sie darstellen.

6.3 Übersicht über die Komponenten

Dieser Abschnitt dient dazu eine Gesamtübersicht über das Kommunikationsframework zur Implementierung einer multimodalen Interaktionsumgebung zu geben. Zunächst werden die einzelnen Komponenten und die Verbindungen zwischen den Komponenten beschrieben. Anschließend werden die wichtigsten Abläufe innerhalb des Kommunikationsframeworks dargestellt. Die Stellen, an denen das Kommunikationsframework erweitert werden muss, um eine Interaktionssystem zu entwickeln, werden besonders hervorgehoben und die Schritte zur Implementierung der Erweiterungen vorgestellt.

Alle Komponenten des Kommunikationsframeworks sind im Namespace *Vis.PowerInteractions* in der Programmiersprache C# implementiert. Abbildung 70 zeigt ein Komponentendiagramm, in dem die Komponenten des Kommunikationsframeworks dargestellt sind. Die Komponenten „Server“, „DataProvider“, „VisualizationClient“ und „Abstraction“ werden im Folgenden kurz vorgestellt:

- Die **Server**-Komponente (Abbildung 70, oben rechts) verwaltet alle Clients, die an der Kommunikation im Framework teilnehmen und beinhaltet den MessageBroker, welcher Nachrichten über das Framework verteilt. Diese Komponente stellt eine Schnittstelle vom Typ IMessageBrokerService bereit. Über diese Schnittstelle können sich Clients und DataProvider am Server anmelden und über ihn Nachrichten verschicken.

- Die Komponente **DataProvider** (Abbildung 70, unten rechts) stellt Daten aus einer Datenquelle dem Server und anderen Clients zur Verfügung. Dafür implementiert sie die Schnittstelle IDataProviderService, über die der Server Daten abrufen und an Clients verteilen kann. Der Zugriff auf die Daten, sowie die Transformation in das Format des Kommunikationsframeworks müssen für jede Datenquelle speziell implementiert werden.

- Die **VisualizationClient**-Komponente (Abbildung 70, unten links) wird von einer Anwendung verwendet, die über das Kommunikationsframework auf Daten zugreift und diese visualisiert. Das Abrufen der Daten und die Anzeige müssen jeweils speziell implementiert werden. Für die Kommunikation mit dem Server werden vom Framework die Klassen Client.Backend, MessageClient und MessageClientService zur Verfügung gestellt. Diese implementieren die Schnittstellen zum Empfangen von Nachrichten aus dem Framework über den Server (Schnittstelle IMessageClientService) und zum Versenden von Nachrichten an den Server und an andere Clients.

- Die Komponente **Abstraction** (Abbildung 70, oben links) stellt Bibliotheken zur Verfügung, die von allen anderen Komponenten verwendet werden. Hierzu zählen beispielsweise Basisklassen zur Implementierung von Visualisierungen oder Klassen zur Erweiterung der Nachrichten, die über das Kommunikationsframework ausgetauscht werden können.

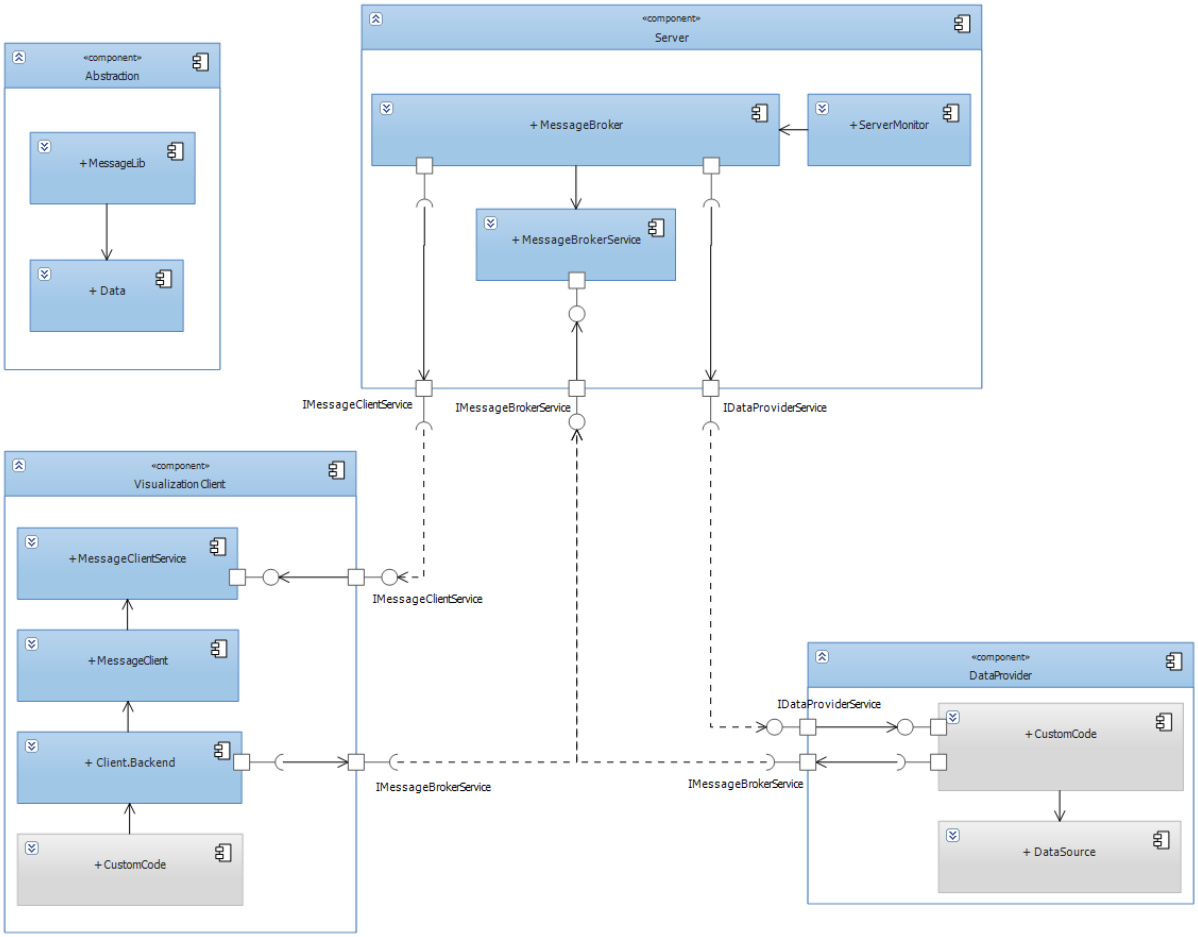


Abbildung 70: Komponentendiagramm des Kommunikationsframeworks. Oben ist die Serverkomponente mit den Schnittstellen zum Nachrichtenaustausch zwischen den Clients dargestellt. Rechts unten die Komponente "DataProvider", mit der Daten zur Visualisierung zur Verfügung gestellt werden können. Links unten die VisualizationClient-Komponente, die Daten zur Interaktion darstellen kann. Die Abstraction-Komponente (links oben) stellt Basisklassen zur Erweiterung des Frameworks dar.
Grau dargestellte Komponenten müssen für jede Anwendung separat implementiert werden.

6.3.1 Server-Komponente

Die Server-Komponente ist im Namespace *Vis.PowerInteractions.Comm* implementiert. Die zentrale Klasse der Server-Komponente ist die Klasse MessageBroker (siehe Abbildung 71). Sie wird im Framework von der Anwendung „ServerMonitor“ instaziiert. Beim Aufruf der Methode **StartService()** startet die Klasse MessageBroker einen Web Service vom Typ MessageBrokerService. Über diesen Service können sich Clients und Daten-Provider registrieren und Nachrichten über das Kommunikationsframework übermitteln. Ein Web Service vom Type MessageBrokerService stellt über die Schnittstelle IMessageBrokerService die folgenden Methoden zur Verfügung:

- **RegisterClient()** : Über diese Methode kann sich ein beliebiger Client am Server anmelden. Der Client muss über einen Web Service mit der Schnittstelle IMessageClientService verfügen. Ab diesem Zeitpunkt können Nachrichten an diesen Client versendet werden.

- **UnregisterClient()** : Über diese Methode kann sich ein Client wieder vom Server abmelden. Anschließend können keine Nachrichten mehr an diesen Client versendet werden.

- **RegisterDataProvider()** : Über diese Methode kann sich ein Daten-Provider am Server anmelden. Der Daten-Provider muss über einen Web Service mit der Schnittstelle IDataProviderService verfügen. Ab diesem Zeitpunkt können Daten von diesem Daten-Provider über den Server abgerufen werden.

- **UnregisterDataProvider()** : Über diese Methode kann sich ein Daten-Provider vom Server abmelden. Anschließend können keine Daten mehr von diesem Daten-Provider abgerufen werden.

- **TransferMessage()** : Über diese Methode kann ein Client eine Nachricht über den Server verschicken. Je nach Nachricht wird diese auf dem Server verarbeitet, an einen bestimmten Client verschickt oder an alle Clients weitergeleitet.
  Informationen über Nachrichten, die mit dieser Methode im Kommunikationsframework übertragen werden können, sind in Kapitel 6.3.4 dokumentiert.

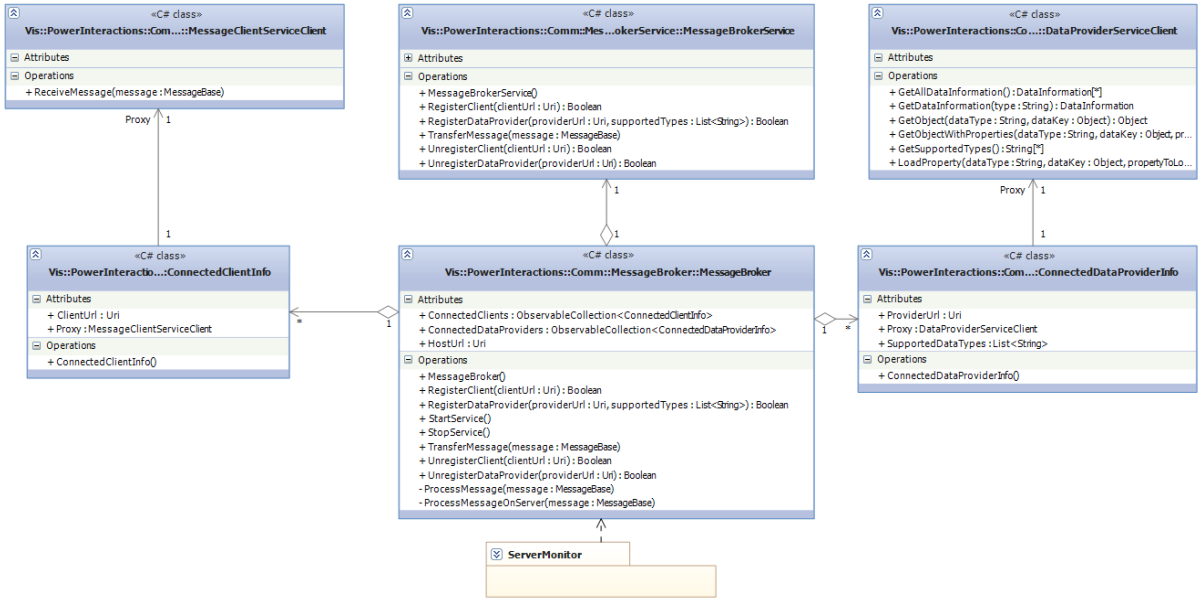


Abbildung 71: Klassenstruktur der Server-Komponente (Abbildung 70 oben rechts). Der Teil "ServerMonitor" kann für jedes Interaktionssystem unterschiedlich implementiert werden.

Registriert sich ein Client am Server, erstellt dieser eine Instanz der Klasse ConnectedClientInfo. Darin wird die URL zur Web Service-Schnittstelle IMessageClientService des Clients gespeichert, die diesen Client eindeutig identifiziert. Zusätzlich wird eine Instanz der Proxyklasse MessageClientServiceClient gespeichert, über die der Server zu diesem Client Nachrichten übertragen kann. Der genaue Ablauf beim Registrieren eines Clients ist im Sequenzdiagramm in Abbildung 77 dargestellt und wird in Kapitel 6.4.1 beschrieben. Meldet sich ein Client vom Server ab, werden diese Informationen wieder gelöscht.

Analog verhält sich der Server bei der Registrierung eines Daten-Providers: in einer Instanz der ConnectedDataProviderInfo werden die URL zum Web Service mit der IDataProviderService-Schnittstelle und ein Proxy gespeichert. Zusätzlich werden Informationen darüber gespeichert, welche Datentypen dieser Daten-Provider zur Verfügung stellt.

Ruft ein Client die Methode **TransferMessage()** auf, bestimmt der Server anhand der Nachricht, an welche Clients diese Nachricht weitergeleitet werden soll und überträgt sie über deren Web Service-

Schnittstelle. Bestimmte Nachrichten müssen auf dem Server verarbeitet werden. Dies sind beispielsweise Nachrichten zum Laden von Daten. Hier sucht der Server den passenden Daten-Provider, welcher die gewünschten Daten zur Verfügung stellt, ruft von diesem die Daten ab und generiert eine Antwortnachricht, die wie beim Aufruf von **TransferMessage()** an die Clients übertragen wird. Die Verarbeitung von Nachrichten auf dem Server wird in Kapitel 6.4.2 detailliert beschrieben.

6.3.2 DataProvider-Komponente

Dieser Abschnitt beschreibt den strukturellen Aufbau der DataProvider-Komponente. Mithilfe der DataProvider-Komponente können anderen Clients Daten zur Verfügung gestellt werden. Alle Klassen der DataProvider-Komponente sind im Namespace *Vis.PowerInteractions.Comm* implementiert und werden in Abbildung 72 dargestellt.

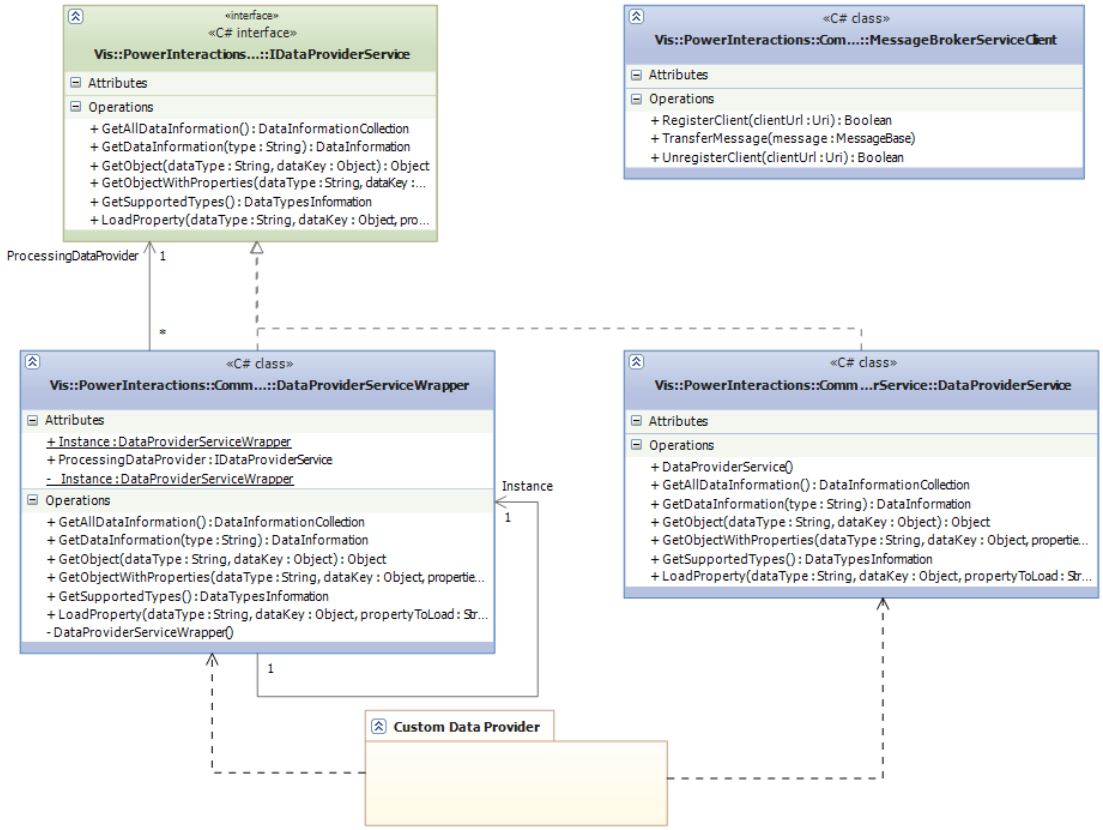


Abbildung 72: Klassenstruktur der DataProvider-Komponente (Abbildung 70 unten rechts). Der Teil "Custom Data Provider" muss für jede Datenquelle speziell implementiert werden.

Das Kommunikationsframework stellt Klassen und Methoden zur Verfügung, mit denen Daten an Clients übertragen werden können. Das Laden der Daten und die Transformation in ein Format, das über das Kommunikationsframework transferiert werden kann, sind abhängig von der verwendeten Datenquelle und müssen daher jeweils speziell implementiert werden. Dies ist in Abbildung 72 durch den Platzhalter „Custom Data Provider“ gekennzeichnet. Quellcode 1 zeigt die Verwendung der Klassen der DataProvider-Komponente, um einen neuen Daten-Provider in einer Microsoft.NET-Anwendung zu erstellen.

Das Interface IDataProviderService legt die Schnittstelle für den Web Service fest, über den die Server-Komponente auf Daten eines Daten-Providers zugreifen kann. Folgende Methoden müssen unterstützt werden:

- **GetSupportedTypes()** : Über diese Methode kann der Server bei Bedarf abrufen, welche Datentypen von diesem Daten-Provider anderen Clients zur Verfügung gestellt werden. Hier kann beispielsweise eine Liste aller Tabellennamen aus einer Datenbank oder eine Liste aller Namen der Datenklassen zurückgegeben werden, die von diesem Daten-Provider zur Verfügung gestellt werden.

- **GetDataInformation()** : Über diese Methode kann der Server Übersichtsinformationen über alle Daten eines Typs abrufen. Die Übersichtsinformationen geben Auskunft darüber, welche Daten in diesem Daten-Provider vorhanden sind. Aus einer Datenbank können hier beispielsweise die Primärschlüssel aller Einträge einer bestimmten Tabelle zurückgegeben werden.

- **GetAllDataInformation()** : Über diese Methode kann der Server Übersichtsinformationen über alle Daten abrufen. Aus einer Datenbank können hier beispielsweise die Primärschlüssel aller Einträge aller Tabellen zurückgegeben werden.

- **GetObject()** : Über diese Methode kann der Server ein bestimmtes Objekt vom Daten-Provider anfordern. Sollten die vom Daten-Provider veröffentlichten Daten Relationen untereinander besitzen, muss hier ein flaches Objekt ohne Relationen zurückgegeben werden. Relationen werden über die Methode **LoadProperty()** abgerufen.

- **GetObjectWithProperties()** : Diese Methode erlaubt dem Server ein Objekt mit Relationen zu laden.

- **LoadProperty()** : Über diese Methode kann der Server die Eigenschaft eines Objekts aus der Datenquelle nachladen, die selbst wieder ein Objekt des Daten-Providers ist und mit dem ersten Objekt in einer Relation steht.

Erstellen eines Daten-Providers in Microsoft.NET

Um einen eigenen Daten-Provider für das Kommunikationsframework zu generieren, muss eine Klasse erstellt werden, welche die Schnittstelle IDataProviderService implementiert. Innerhalb dieser Klasse muss der Zugriff auf die Datenquelle und die Transformation der Daten in das Format des Kommunikationsframeworks erfolgen. Die Implementierung ist abhängig von der jeweiligen Datenquelle.

In einer ausführbaren Anwendung (beispielweise einer WPF-Anwendung, Konsolenanwendung oder einem Windows-Dienst) muss ein Service Host vom Typ *System.ServiceModel*.ServiceHost gestartet werden, um einen Web Service veröffentlichen zu können. Als Diensttyp muss die Klasse *Vis.PowerInteractions.Comm*.DataProviderService angegeben werden. Anschließend muss die Eigenschaft **ProcessingProvider** der statischen Klasse *Vis.PowerInteractions.Comm.DataProviderService*.DataProviderServiceWrapper gesetzt werden. Dazu wird eine Instanz der Klasse verwendet, die zuvor erstellt wurde und die Schnittstelle IDataProviderService implementiert. Anschließend kann der Dienst gestartet und über die Methode **RegisterDataProvider()** bei der Server-Komponente registriert werden.

```
using System;
using System.ServiceModel;
using Vis.PowerInteractions.Comm.DataProviderService;
using Vis.PowerInteractions.Comm.MessageBrokerProxy.Proxy;

namespace DataProviderSample
{
    public class CustomDataProvider : IDataProviderService
    { /* Inhalt dieser Klasse ist von der lokalen Datenquelle abhängig. */ }

    /// <summary>
    /// Konsolenanwendung, die einen Daten-Provider implementiert.
    /// </summary>
    class Program
    {
        static void Main(string[] args)
        {
            // Instanz des Datenproviders, die Daten aus der lokalen Datenquelle
            // laden soll.
            var dataProvider = new CustomDataProvider();

            // Liste der Datentypen, die über diesen Daten-Provider
            // verfügbar sind. Hier: Personen und Adressen.
            var supportedTypes = new string[] { "Person", "Adresse" };


            // Web Service für einen Daten-Provider erstellen.
            var host = new ServiceHost(typeof(DataProviderService));
            // festlegen, dass die Instanz des Datenproviders die Aufrufe
            // des Web Service bearbeiten soll.
            DataProviderServiceWrapper.Instance.ProcessingDataProvider
                    = dataProvider;
            // Web Service starten
            host.Open();


            // Verbindung zur Server-Komponente herstellen.
            var serverProxy = new MessageBrokerServiceClient();
            // Diesen Daten-Provider beim Server registrieren.
            serverProxy.RegisterDataProvider(host.Description.Endpoints[0].Address.Uri,
                                             supportedTypes);

            // Programm erst durch Benutzer beenden lassen.
            Console.ReadKey();

            // zum Beenden des Programms Web Service schließen.
            host.Close();
        }
    }
}
```

Quellcode 1: Implementierung eines Daten-Providers in einer Konsolenanwendung.

6.3.3 VisualizationClient-Komponente

In diesem Abschnitt wird der strukturelle Aufbau der VisualizationClient-Komponente beschrieben. Ebenfalls wird die Plugin-Struktur vorgestellt, mit der weitere Visualisierungen beispielsweise zur Darstellung von Daten auf der Powerwall erstellt werden können. Die Klassen der VisualizationClient-Komponente sind im Namespace *Vis.PowerInteractions.Client* erstellt. Klassen zur Kommunikation mit dem Framework befinden sich im Namespace *Vis.PowerInteractions.Comm*.
Abbildung 73 stellt die Klassen der VisualizationClient-Komponente dar.

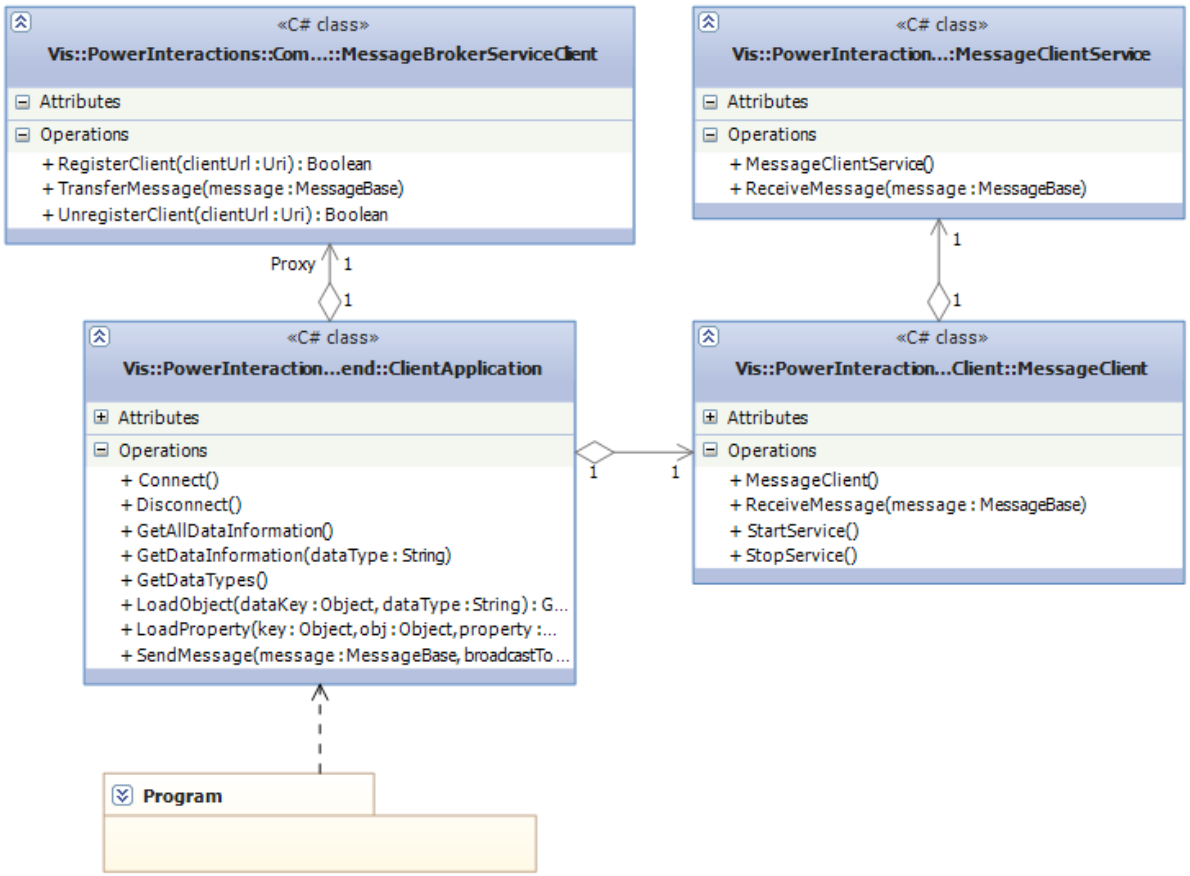


Abbildung 73: Klassenstruktur der VisualizationClient-Komponente (Abbildung 70 unten links). Der Teil "Program" visualisiert Daten und ermöglicht die Interaktion. Für jeden Gerätetyp in der Interaktionsumgebung muss ein VisualizationClient und das dazugehörige Programm implementiert werden.

Der Platzhalter „Program“ in Abbildung 73 repräsentiert eine Anwendung, die einem Benutzer Daten visualisieren und zur Interaktion bereitstellen kann. Da die Anforderungen an diese Anwendung sowohl von dem Gerät, auf dem sie verwendet werden soll, den Interaktionen, die sie dem Benutzer bieten soll, als auch von den anzuzeigenden Daten bestimmt werden, muss die Anwendung für das jeweilige Anwendungsszenario speziell implementiert werden. Die Klassen der VisualizationClient-Komponente stellen einer solchen Anwendungen Methoden zur Verfügung, mit denen Daten aus dem Kommunikationsframework abgerufen werden und Nachrichten zu anderen Clientanwendungen übertragen werden können.

Über die Klasse ClientApplication erhält eine Anwendung, die Daten für Benutzer visualisieren und zur Interaktion bereitstellen soll, Zugriff auf die Server-Komponente und Zugang zu den Daten der beim Server registrierten Daten-Provider. Die Klasse ClientApplication stellt hierfür folgende Methoden zur Verfügung:

- **Connect()** : Mit dieser Methode kann eine Verbindung zur Server-Komponente hergestellt werden. Anschließend kann die Anwendung Nachrichten vom Server empfangen.
- **Disconnect()** : Mit dieser Methode kann eine bestehende Verbindung zur Server-Komponente wieder getrennt werden. Anschließend können weder Nachrichten an den Server gesendet noch vom Server empfangen werden.

- **GetAllDataInformation()** : Über diese Methode werden Übersichtsinformationen von allen Daten, die über Daten-Provider im Kommunikationsframework zur Verfügung gestellt werden, abgerufen.
- **GetDataInformation()** : Mit dieser Methode werden Übersichtsinformationen von allen Daten eines bestimmten Typs abgerufen.
- **GetDataTypes()** : Diese Methode ruft bei allen im Kommunikationsframework verfügbaren Daten-Providern ab, welche Datentypen sie zur Verfügung stellen.
- **LoadObject()** : Mit dieser Methode kann ein bestimmtes Objekt über das Kommunikationsframework abgerufen werden.
- **LoadProperty()** : Mit dieser Methode können Eigenschaften eines Datenobjekts aus der Datenbank nachträglich abgerufen werden.
- **SendMessage()** : Mit dieser Methode können Nachrichten an andere Clients, die mit dem Kommunikationsframework verbunden sind, oder an die Server-Komponente übermittelt werden.

Aufrufe, die an die Server-Komponente gerichtet sind, leitet die Klasse ClientApplication über eine Instanz der Proxyklasse MessageBrokerServiceClient an den Server weiter. Gleichzeitig erstellt sie über die Klasse MessageClient einen Web Service vom Type MessageClientService, über den Nachrichten aus dem Kommunikationsframework empfangen werden können. Empfangene Nachrichten werden über Events an die Anwendung weitergeleitet, die eine Verbindung zum Server aufgebaut hat.

Verwendung von Visualisierungs-Plugins

Visualisierungen, die Benutzern Daten aus dem Kommunikationsframework darstellen und zur Interaktion zur Verfügung stellen können, sollten als Plugins implementiert werden. Dies ermöglicht die Verwendung derselben Visualisierungen auf mehreren Gerätetypen und erleichtert das Hinzufügen neuer Visualisierungen.

Hierfür wurde eine Klassenstruktur entwickelt, mit der Visualisierungs-Plugins erstellt und in Anwendungen der VisualizationClient-Komponente verwendet werden können. Diese Plugin-Struktur befindet sich im Namespace *Vis.PowerInteractions.Client.Visualizations* und wird in Abbildung 74 dargestellt.

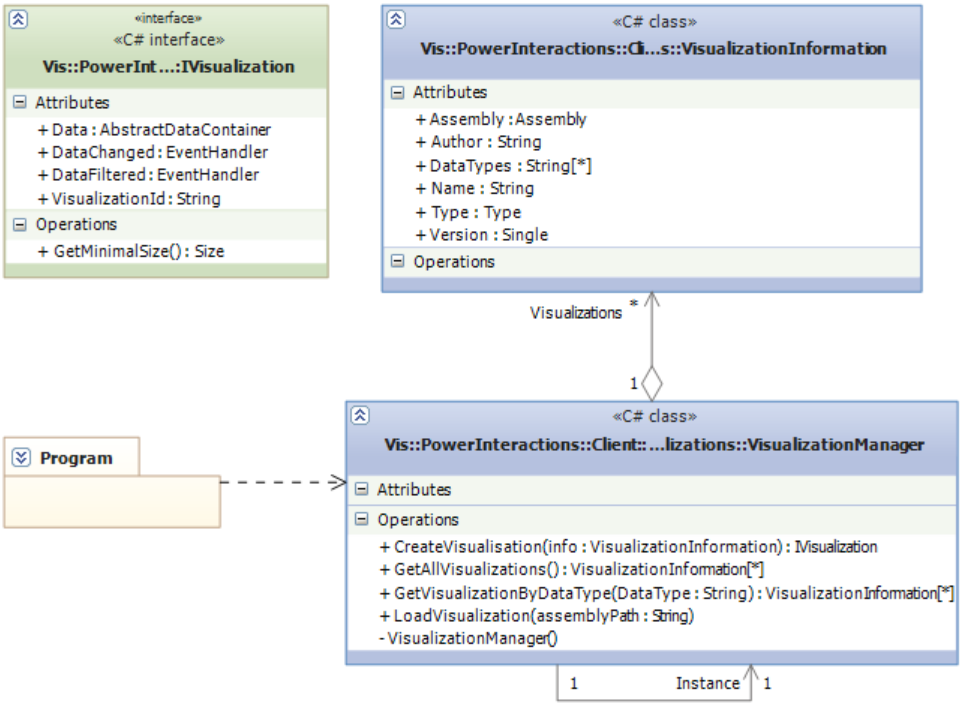


Abbildung 74: Plugin-Struktur für Visualisierungen, die in einem VisualizationClient verwendet werden können. Der Platzhalter „Program“ steht für die Anwendung der VisualizationClient-Komponente, welche die Visualisierungs-Plugins verwenden soll.

Oberflächenelemente, die von Anwendungen in der VisualizationClient-Komponente verwendet werden sollen, müssen die Schnittstelle IVisualization implementieren.

Um die Plugins verwenden zu können, muss in der Anwendungskonfigurationsdatei App.config (Beispiel siehe Quellcode 2) des Programms ein Abschnitt <VisualizationAssemblies> angelegt und der passende ConfigurationHandler angegeben werden.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <!-- Visualization Plugins -->
    <section name="VisualizationAssemblies"
type="Vis.PowerInteractions.Client.Visualizations.VisualizationAssemblySectionHandler,
Vis.PowerInteractions.Client.Visualizations" />
  </configSections>

  <VisualizationAssemblies>
    <Path>C:\MyPlugins\</Path>
  </VisualizationAssemblies>
</configuration>
```

Quellcode 2: XML-Markup in einer Anwendungskonfigurationsdatei, mit der Visualisierungs-Plugins für die Anwendung aktiviert werden.

Innerhalb des XML Elements <VisualizationAssemblies> kann eine Liste von Verzeichnissen angegeben werden, die vom der Klasse VisualizationManager nach Visualisierungs-Plugins durchsucht werden sollen. In jedem angegebenen Verzeichnis muss sich eine XML-Datei mit dem Namen „visConfig.xml“ befinden. Diese Datei spezifiziert ein Visualisierungs-Plugin, auf welche Datentypen die Visualisierung angewendet werden kann und die Bibliothek, in der das Plugin-Control zu finden ist. Quellcode 3 zeigt beispielhaft eine solche XML-Datei.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <Author>Nico Ploner</Author>
  <Path>Vis.PowerInteractions.Samples.Mp3Library.Vis.dll</Path>
  <Visualizations>
    <Visualization name="Mp3Vis"
                   version="1.0"
                   class="Vis.PowerInteractions.Samples.Mp3Library.Vis.Mp3Vis">
      <Type>Vis.PowerInteractions.Samples.Mp3Library.Model.Song</Type>
    </Visualization>
  </Visualizations>
</root>
```

Quellcode 3: Beispiel einer XML-Konfiguration eines Visualisierungs-Plugins.

In einer visConfig.xml-Datei legt der Inhalt des <Path>-Elements den relativen Pfad zur Bibliothek fest, in der das Visualisierungs-Plugin implementiert ist. Basis zu diesem relativen Pfad ist das Verzeichnis der visConfig.xml-Datei.

Zu jedem Visualisierungs-Plugin in dieser Bibliothek muss angegeben werden, welche Klasse innerhalb der Bibliothek dieses Plugin implementiert. Zusätzlich wird die Information benötigt, welche Datentypen durch dieses Plugin visualisiert werden können.
Im Beispiel in Quellcode 3 verweist die visConfig.xml auf eine Bibliothek, die im selben Verzeichnis liegt. Darin ist in der Klasse *Vis.PowerInteractions.Samples.Mp3Library.Vis*.Mp3Vis ein Visualisierungs-Plugin implementiert, das Daten vom Typ *Vis.PowerInteractions.Samples.Mp3Library.Model*.Song visualisiert.

Erstellen eines Visualisierungs-Clients in Microsoft.NET

Visualisierungs-Clients können beispielsweise eine WPF- oder Surface-Anwendung sein. Als ein Prototyp wurde ein Visualisierungs-Client für das Kommunikationsframework erstellt, der eine Musikbibliothek auf der Powerwall darstellen kann.

Im Hauptfenster der Anwendung wird mithilfe der ClientApplication-Klasse aus der Komponente VisualizationClient eine Verbindung zur Serverkomponente hergestellt. Dazu wird die Methode **Connect()** aufgerufen. Um auf Nachrichten reagieren zu können, die vom Server empfangen werden, werden die nötigen Events der ClientApplication-Klasse abonniert. Anschließend wird beispielhaft ein Song aus der Musikbibliothek vom Server abgerufen und mithilfe eines Visualisierungs-Plugins auf der Powerwall angezeigt. Schließlich wird allen anderen Clients mitgeteilt, dass dieser Song geladen wurde. Quellcode 4 zeigt beispielhaft die Implementierung eines Visualisierungs-Clients in einer WPF-Anwendung.

Zur Verwendung der Visualisierungs-Plugins wurden eine Anwendungskonfigurationsdatei wie in Quellcode 2 und eine visConfig.xml wie in Quellcode 3 beschrieben verwendet.

```
using System;
using System.Linq;
using System.Windows;
using Vis.PowerInteractions.Client.Backend;
using Vis.PowerInteractions.Client.Visualizations;

namespace MusicLibrary
{
    /// <summary>
    /// Hauptfenster des Musikbibliothek für den Tag der Wissenschaft 2012
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            this.Loaded += new RoutedEventHandler(MainWindow_Loaded);
        }

        /// <summary>
        /// ClientApplication, über die eine Verbindung zum Server
        /// möglich ist.
        /// </summary>
        private ClientApplication clientApp;

        /// <summary>
        /// Wird ausgeführt, wenn das Hauptfenster geladen wurde.
        /// </summary>
        void MainWindow_Loaded(object sender, RoutedEventArgs e)
        {
            // Instanz der ClientApplication aus der VisualizationClient-Komponente
            clientApp = App.Current as ClientApplication;

            // Nötige Events für Servernachrichten abonnieren
            clientApp.ObjectLoadedEvent += Application_ObjectLoaded;
            /*
             * ebenso für andere Events verfahren
             */

            // Verbindung zum Server herstellen
            clientApp.Connect();

            // Exemplarisch einen Song mit der ID 1 abrufen
            clientApp.LoadObject(1, "Song");
        }

        /// <summary>
        /// Wird ausgeführt, wenn ein Objekt vom Daten-Provider geladen wurde.
        /// </summary>
        void Application_ObjectLoaded(object sender, ObjectLoadedEventArgs e)
        {
            // überprüfen, welches Objekt geladen wurde
            if (e.Object.GetType().ToString() == "Song")
            {
                // ein Song wurde geladen
                // > diesen in einer Plugin-Visualisierung für Songs anzeigen
                var song = e.Object;
```

```
                // die erste Visualisierungsmöglichkeit für Songs verwenden
                var visInfo = VisualizationManager.Instance
                        .GetVisualizationByDataType("Song").First();
                // eine Visualisierung davon erstellen und den Song darstellen
                var vis = VisualizationManager.Instance
                        .CreateVisualization(visInfo);
                vis.Data = song;

                // Visualisierung im Fenster anzeigen
                this.AddChild(vis);

                // anderen Clients mitteilen, dass der Song geladen wurde
                clientApp.BroadcastNewData(1, song);
            }
        }
    }
}
```

Quellcode 4: Beispielcode zum Erstellen eines Visualisierungs-Clients und der Verwendung von Plugins zur Visualisierung.

6.3.4 Abstraction-Komponente

Die Komponente „Abstraction“ implementiert Basisklassen und Basisfunktionalität für den Daten- und Nachrichtentransfer über das Kommunikationsframework. Sie ermöglicht die Datenunabhängigkeit sowie die Erweiterbarkeit und wird von allen Komponenten des Frameworks verwendet. Ebenso wird die Funktionalität zum Filtern von Daten innerhalb einer Visualisierung hier zur Verfügung gestellt. Die Klassen der Abstraction-Komponente sind in den *Namespaces Vis.PowerInteractions.Data* und *Vis.PowerInteractions.Comm.MessageLib* implementiert.

Zu den Basisklassen des Datentransfers gehören die abstrakte Klasse AbstractDataContainer sowie ihre Realisierungen DataContainer und CollectionDataContainer und die Klasse Filter (siehe Abbildung 75). Für den Nachrichtentransfer ist die abstrakte Klasse MessageBase von zentraler Bedeutung (siehe Abbildung 76 oben). Die Abstraction-Komponente stellt bereits zahlreiche Nachrichten zur Verfügung.

Daten und Filterung

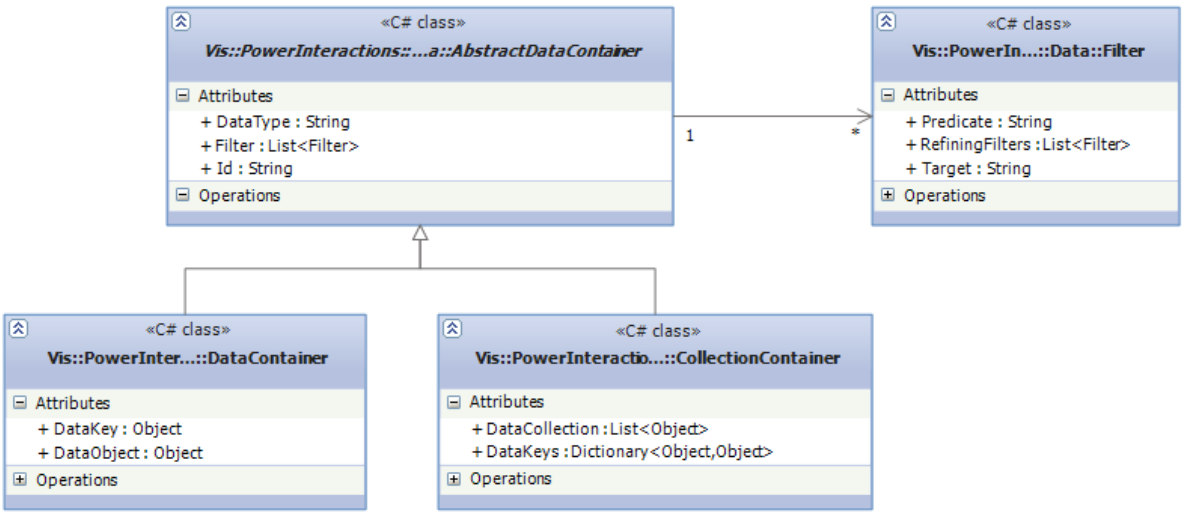


Abbildung 75: Klassenstruktur zum Datentransfer im Kommunikationsframework und zur Datenfilterung.

Zum Austausch und Synchronisierung von Daten zwischen den Clients werden Datencontainer verwendet. Die Klassen DataContainer und CollectionContainer beinhalten Objekte, die von einem Daten-Provider geladen und an alle Clients verteilt wurden. Alle Kopien haben dieselbe eindeutige Id, sodass die Daten eines Containers zwischen mehreren Clients synchronisiert werden können. In der Eigenschaft **Filter** können Filterobjekte gespeichert werden, welche die Menge der angezeigten Daten beeinflussen können.

Datencontainer können an Visualisierungs-Plugins übergeben werden, sodass diese die Daten des Containers darstellen und Interaktionen ermöglichen können.

Nachrichtentransfer

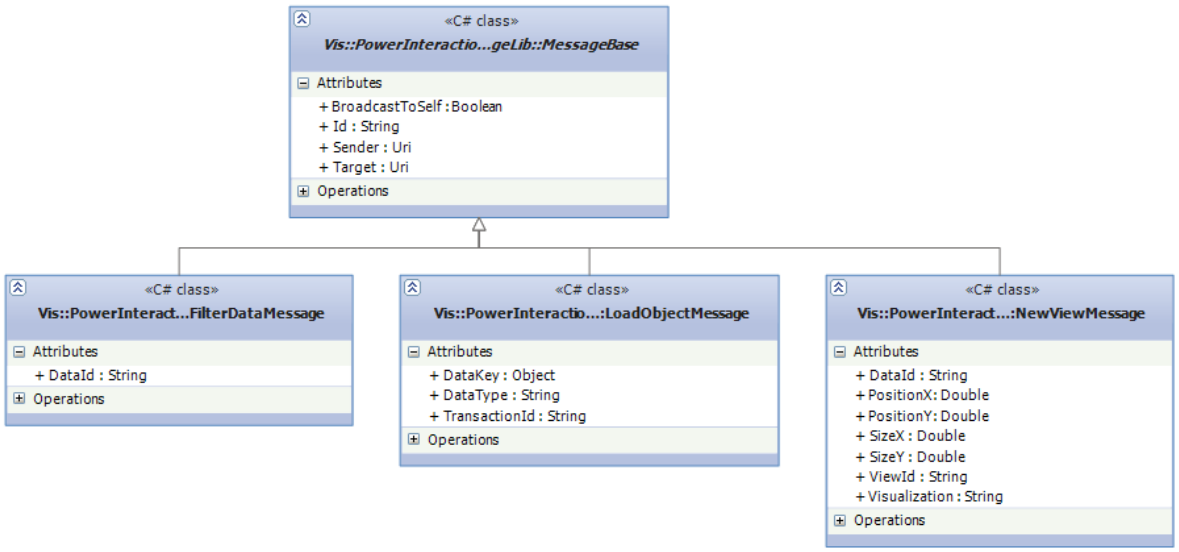


Abbildung 76: Klassenstruktur der Nachrichten, die durch das Kommunikationsframework übertragen werden können. Beispielhaft ist je eine Nachrichtenklasse zum Filtern (FilterDataMessage), zum Datentransfer (LoadObjectMessage) und zur Dateninteraktion (NewViewMessage) dargestellt.

Zur Nachrichtenübermittlung zwischen Clients bietet die Abstraction-Komponente die Basisklasse MessageBase an. Alle Nachrichten, die über das Kommunikationsframework übermittelt werden sollen, müssen von dieser Basisklasse erben (siehe Abbildung 76).

Beim Übermitteln einer Nachricht müssen folgende Eigenschaften der Basisklasse MessageBase gesetzt sein:

- **BroadcastToSelf** : True, wenn diese Nachricht vom Server auch an den Absender der Nachricht übermittelt werden soll, andernfalls False.
- **Target** : Adressat dieser Nachricht. Ist kein Uri angegeben, wird die Nachricht an alle verbundenen Clients im Kommunikationsframework gesendet. Andernfalls erhält nur der Client mit dem angegeben Uri diese Nachricht.
- **Sender**: Absender dieser Nachricht. Diese Eigenschaft muss die URL zum Web Service des VisualizationClient beinhalten.

Je nach zu übermittelnder Nachricht müssen die zusätzlichen Eigenschaften der Realisierungen der MessageBase-Klasse ausgefüllt werden.

Die Abstraction-Komponente stellt bereits eine Vielzahl an Nachrichten bereit, die über das Kommunikationsframework transferiert werden können. Diese Nachrichten lassen sich in drei Gruppen unterteilen:

- Nachrichten zur Übermittlung von Informationen über Visualisierungen
- Nachrichten zum Laden und Übertragen von Daten
- Nachrichten zum Filtern von Daten

In Abbildung 76 dargestellt ist für jede Gruppe ein Beispiel dargestellt. Alle Nachrichten werden im Folgenden vorgestellt.

Nachrichten zur Übermittlung von Informationen über Visualisierungen
Diese Nachrichten teilen anderen Clients Änderungen an Visualisierten Daten mit. Folgende Nachrichten sind in der Abstraction-Komponente implementiert:

- ChangeDataMessage: Teilt den Clients mit, dass in einer Ansicht nun andere Daten dargestellt werden sollen.
- ChangeVisualizationMessasge: Teilt den Clients mit, dass in einer Ansicht eine andere Visualisierung verwendet werden soll.
- ConfigVisualizationMessage: Überträgt Konfigurationsparameter einer Ansicht an alle Clients
- CoupleDataMessage: Teilt den Clients mit, dass die Daten zweier Ansichten gekoppelt werden sollen.
- CoupleViewMessage: Teilt den Clients mit, dass zwei Ansichten gekoppelt werden sollen.
- DecoupleDataMessage: Teilt den Clients mit, dass die Daten zweier Ansichten entkoppelt werden sollen.
- DecoupleViewMessage: Teilt den Clients mit, dass zwei Ansichten wieder entkoppelt werden sollen.
- MoveViewMessage: Teilt den Clients mit, dass eine Ansicht an eine andere Position im Darstellungsbereich verschoben wurde.
- NewViewMessage (siehe Abbildung 76, rechts): Teilt den Clients mit, dass eine neue Ansicht für Daten erstellt wurde
- RemoveViewMessage: Teilt den Clients mit, dass eine bestimmte Ansicht entfernt wurde.
- ResizeViewMessage: Teilt den Clients mit, dass eine Ansicht vergrößert oder verkleinert wurde.
- ScrollVisualizationMessage: Teilt den Clients mit, dass innerhalb einer Ansicht gescrollt wurde.
- ZoomVisualizationMessagse: Teilt den Clients mit, dass innerhalb einer Ansicht gezoomt wurde.

Nachrichten zum Laden und Übertragen von Daten
Diese Nachrichten geben Auskunft über das Laden von Daten und Übersichtsinformationen aus Daten-Providern. Folgende Nachrichten sind in der Abstraction-Komponente implementiert:

- AllDataInformationMessage: Antwortnachricht vom Server an einen Client, der Übersichtsinformationen über alle Daten abgerufen hat.
- CollectionPropertyLoadedMessage: Antwortnachricht vom Server an einen Client, der eine Listeneigenschaft eines Objekts abgerufen hat.
- DataInformationMessage: Antwortnachricht vom Server an einen Client, der Übersichtsdaten über einen bestimmten Datentyp abgerufen hat.
- DataTypesInformationMessage: Antwortnachricht vom Server an einen Client, der abgerufen hat, welche Datentypen im Kommunikationsframework verfügbar sind.
- GetAllDataInformationMessage: Nachricht von einem Client an den Server zum Abrufen der Übersichtsdaten aller Datentypen, die im Kommunikationsframework verfügbar sind.
- GetDataInformationMessage: Nachricht von einem Client an den Server, zum Abrufen der Übersichtsdaten eines Datentyps, der im Kommunikationsframework verfügbar ist.
- GetDataTypesMessage: Nachricht an den Server zum Abrufen aller Datentypen, die im Kommunikationsframework verfügbar sind.
- LoadCollectionPropertyMessasge: Nachricht an den Server, um eine Listeneigenschaft eines Objekts von einem Daten-Provider abzurufen.
- LoadObjectMessage: Nachricht an den Server, um ein Objekt von einem Daten-Provider abzurufen.
- LoadPropertyMessage: Nachricht an den Server, um eine Eigenschaft eines Objekts von einem Daten-Provider abzurufen.
- NewDataMessage: Teilt allen Clients mit, dass neue Daten von einem Daten-Provider abgerufen wurden.
- ObjectLoadedMessage: Antwortnachricht vom Server an einen Client, der ein Objekt von einem Daten-Provider abgerufen hat.
- ObjectOutdatetMessage: Teilt allen Clients mit, dass ein bestimmtes Objekt geändert wurde und erneut vom Daten-Provider abgerufen werden muss.
- PropertyLoadedMessage: Antwortnachricht vom Server an einen Client, der eine Eigenschaft eines Objekts abgerufen hat.
- RemoveDataMessage: Teilt allen Clients mit, dass ein bestimmtes Datum gelöscht werden soll.
- SaveObjectMessage: Nachricht an den Server, Änderungen an einem Objekt im Daten-Provider zu speichern.

Nachrichten zum Filtern von Daten
Diese Nachrichten teilen Clients mit, nach welchen Kriterien Daten gefiltert und so von der Anzeige in Visualisierungen ausgenommen werden sollen.

- FilterDataMessage: Teilt den Clients mit, nach welchen Kriterien Daten innerhalb einer Ansicht gefiltert werden sollen.

6.4 Abläufe

Dieser Abschnitt beschreibt die wichtigsten Abläufe innerhalb des Kommunikationsframeworks detailliert. Es wird dargestellt, wie ein Client beim Server registriert wird, wie Nachrichten über das Kommunikationsframework übertragen werden und Datenobjekte aus einem Daten-Provider geladen werden.

6.4.1 Client registrieren

Bevor Visualisierungs-Clients oder Daten-Provider miteinander kommunizieren können, müssen sie bei der Server-Komponente des Kommunikationsframeworks registriert werden. Abbildung 77 stellt den Ablauf des Registrierens eines Clients beim Server dar.

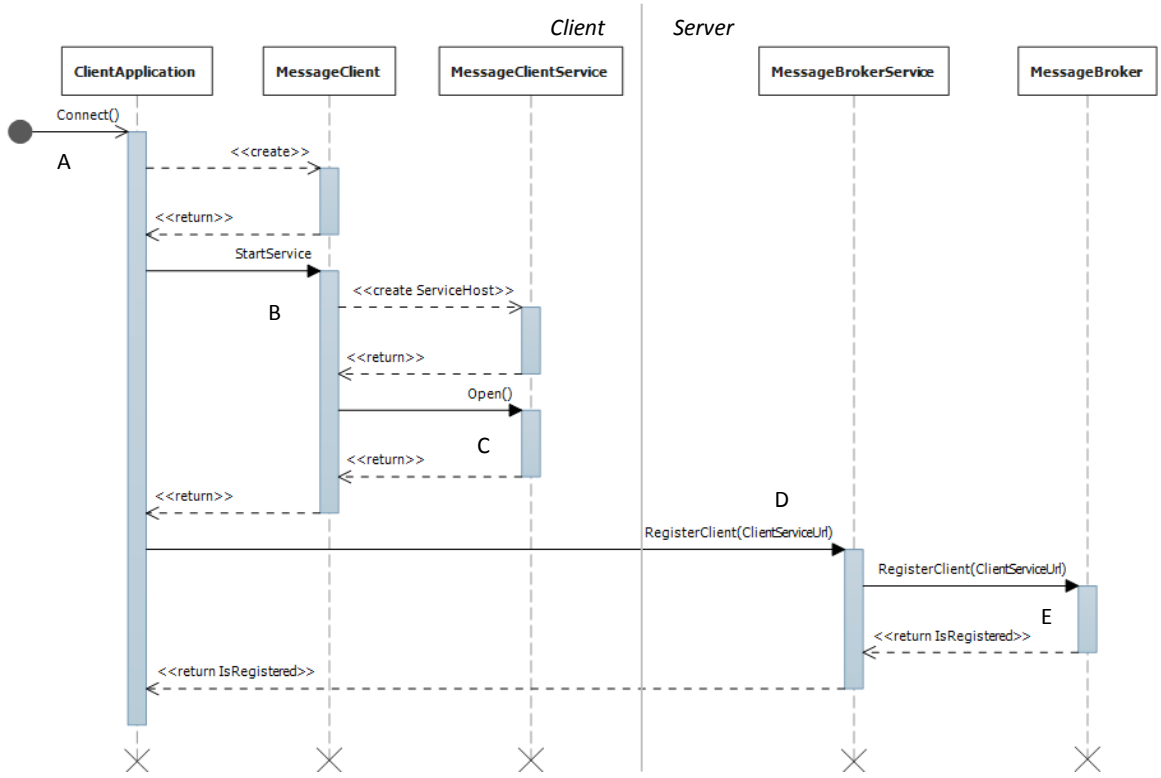


Abbildung 77: Sequenzdiagramm zum Ablauf beim Registrieren eines VisualizationClients beim Server. Über die Klasse ClientApplication und den WebService des VisualizationClients wird der Client beim MessageBroker des Servers registriert.

Soll ein Visualisierungs-Client bei der Server-Komponente registriert werden, kann dies entweder direkt über die vom Server angebotene Web Service-Schnittstelle geschehen oder über die Klasse ClientApplication der VisualizationClient-Komponente.

Wird auf einer Instanz der Klasse ClientApplication die **Connect()**-Methode aufgerufen (A in Abbildung 77), so wird zunächst eine Instanz des Klasse MessageClient erstellt, welche den Web Service des Visualisierungs-Clients verwaltet. Über diesen Web Service kann der Server Nachrichten

nach der Registrierung an diesen Client übermitteln (vgl D in Abbildung 78). Anschließend wird über die Methode **StartService()** (B in Abbildung 77) in der Klasse MessageClient ein .NET ServiceHost erstellt, welcher den Web Service mit der Schnittstelle IMessageClientServie und der Methode **Open()** startet (C in Abbildung 77). Mit der URL dieses Web Services als Parameter wird nun die Methode **RegisterClient()** auf dem Web Service des Servers aufgerufen (D in Abbildung 77). Der Web Service des Servers teilt der Klasse MessageBroker mit, dass ein neuer Client registriert werden soll. Der MessageBroker speichert die URL zu diesem Client, um Nachrichten an ihn übermitteln zu können (E in Abbildung 77). Der boolesche Rückgabewert der Methode „RegisterClient()“ des MessageBrokers gibt an, ob der Client beim MessageBroker registriert ist.

6.4.2 Nachricht übertragen

Über das Kommunikationsframework können Clients untereinander Nachrichten austauschen und Nachrichten an den Server übermitteln. Dies kann direkt über die von den einzelnen Komponenten bereitgestellten Web Services erfolgen. Das Framework bietet zur Implementierung von Client-Anwendungen Klassen und Methoden, welche die Kommunikation vereinfachen.

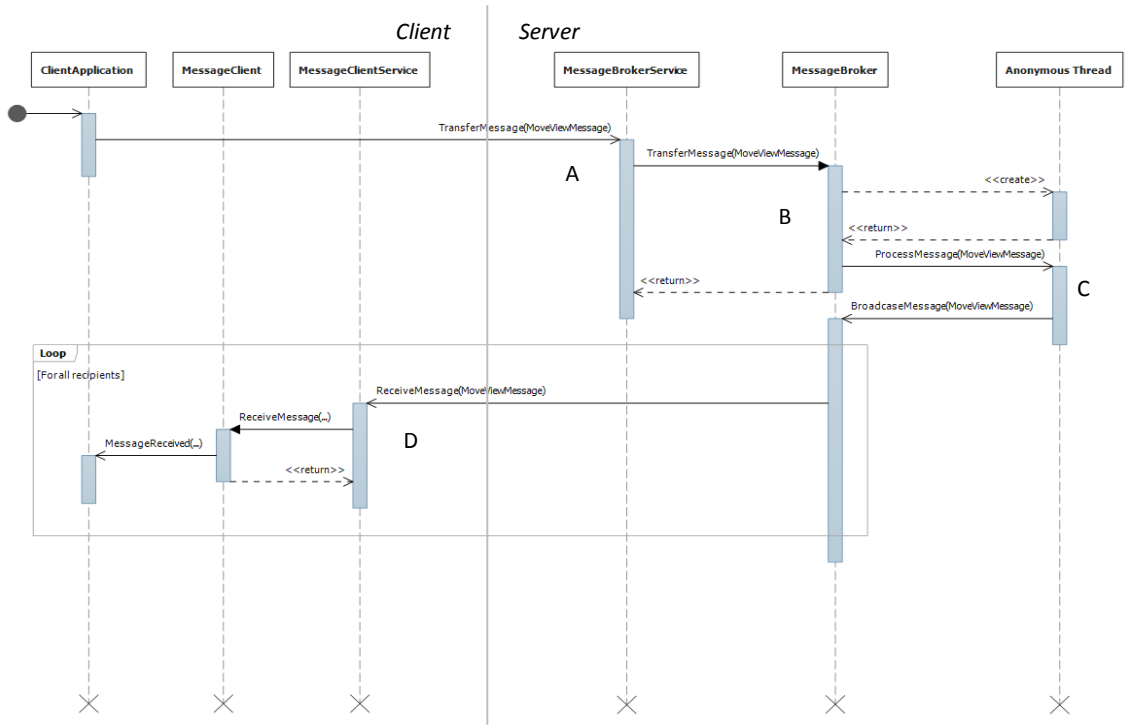


Abbildung 78: Sequenzdiagramm zum Ablauf beim Versenden einer Nachricht an alle Clients in der Interaktionsumgebung. Über die Klasse ClientApplication und den WebService des VisualizationClients wird die Nachricht zum MessageBroker des Servers übertragen, der die Nachricht an alle verbundenen Clients weiterleitet.

Wird die Methode **TransferMessage()** der Klasse ClientApplication aufgerufen, wird die zu übermittelnde Nachricht an die Serverkomponente weitergeleitet. Dazu wird auf dem Web Service des Servers die Methode **TransferMessage()** aufgerufen (A in Abbildung 78). Damit dieser Aufruf die Clientanwendung nicht blockiert, ist der Aufruf asynchron. Der Web Service des Servers übergibt die Nachricht an den MessageBroker, der für die Verteilung von Nachrichten zuständig ist (B in Abbildung 78). In einem separaten Thread verarbeitet der MessageBroker die Nachricht (C in Abbildung 78). Je nach Inhalt der Nachricht (vergleiche Abbildung 79) verteilt der MessageBroker die Nachricht in einer Schleife an alle Empfänger der Nachricht durch den Aufruf der Methode ReceiveMessage() jedes Empfänger-Clients (D in Abbildung 78) oder verarbeitet die Nachricht selbst.

Nachrichtenverarbeitung auf dem Server

Die Serverkomponente verfügt über unterschiedliche Möglichkeiten eine Nachricht empfangene Nachricht zu verarbeiten: Die Nachricht kann an einen bestimmten Empfänger gerichtet sein, an alle Clients verteilt werden oder auf dem Server selbst verarbeitet werden. Abbildung 79 stellt dar anhand welcher Kriterien die Verarbeitung einer Nachricht auf dem Server verläuft.

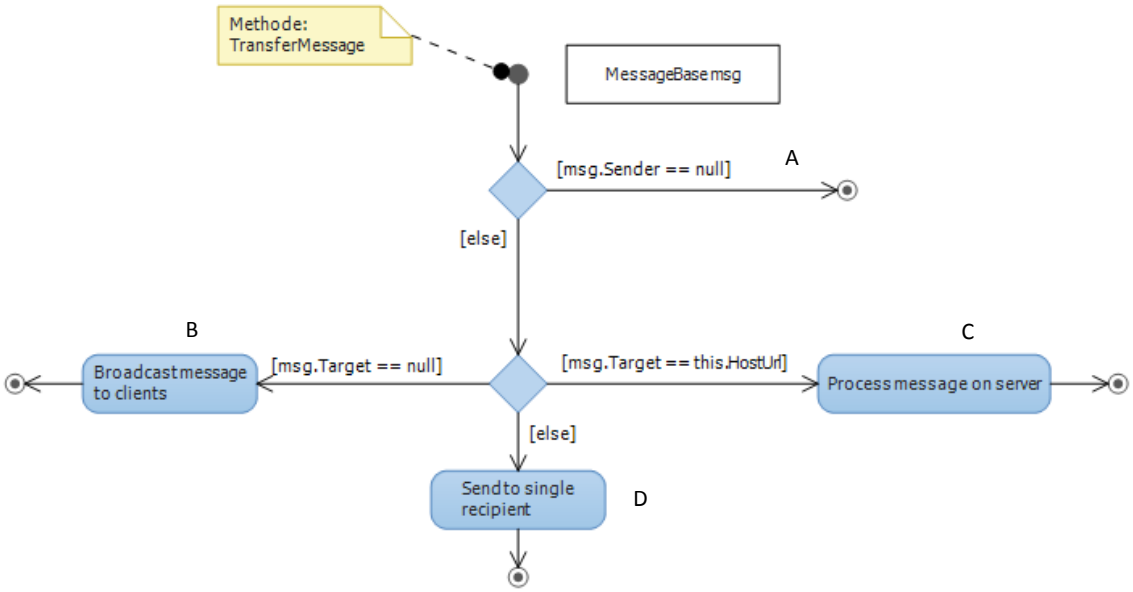


Abbildung 79: Aktivitätsdiagramm zur Verarbeitung einer Nachricht im MessageBroker der Server-Komponente. Es wird entschieden, ob die Nachricht auf dem Server verarbeitet oder an welche Clients die Nachricht weitergeleitet wird.

Wird in der Klasse MessageBroker die Methode **TransferMessage()** aufgerufen, prüft die Server-Komponente zunächst, ob die Nachricht über einen Absender verfügt. Ist die Eigenschaft **Sender** der Nachricht nicht gesetzt, wird die Nachricht ignoriert und nicht weiter verarbeitet (A in Abbildung 79). Ist ein Absender der Nachricht vorhanden, entscheidet die Eigenschaft **Target** über die weitere Verarbeitung der Nachricht:

- Es ist kein Empfänger in der Eigenschaft **Target** der Nachricht gesetzt:
  Diese Nachricht wird an alle verbundenen Clients weitergeleitet (B in Abbildung 79).
- Der Wert der Eigenschaft **Target** entspricht der URL des Servers:
  Diese Nachricht wird auf dem Server verarbeitet. Dies sind beispielsweise Nachrichten zum Abrufen von Daten aus einem Daten-Provider (C in Abbildung 79).
- Der Wert der Eigenschaft **Target** entspricht der URL eines verbundenen Clients:
  Diese Nachricht wird nur an diesen einen Client weitergeleitet (D in Abbildung 79).

Nachdem bestimmt wurde, an welche Clients die Nachricht übertragen werden soll, wird für jeden Client die folgende Logik ausgeführt: Über die Web Service-Schnittstelle MessageClientService des jeweiligen Clients wird die Nachricht an den Client übermittelt. Der Web Service benachrichtigt die ClientApplication über ein Event über die erhaltene Nachricht. Diese kann nun weiterverarbeitet oder an die eigentliche Client-Anwendung weitergegeben werden.

Nachrichten, die auf dem Server verarbeitet werden, generieren zunächst keine Antwortnachrichten. Je nach Implementierung auf dem Server können neue Nachrichten generiert werden, die anschließend nach demselben Schema verarbeitet werden. Dies sind beispielsweise Nachrichten zum Laden von Datenobjekten.

6.4.3 Datenobjekt laden

Das Laden eines Datenobjekts erfolgt durch den Austausch zweier Nachrichten über das Kommunikationsframework: Die Client-Komponente überträgt eine Nachricht vom Typ LoadObjectMessage an den Server. In dieser Nachricht sind folgende Informationen enthalten:

LoadObjectMessage	
Target = Server-URL	*Gibt an, dass diese Nachricht vom Server zu verarbeiten ist*
Sender = Client-URL	*Gibt an, an wen die Antwortnachricht zu versenden ist*
DataType	*Gibt an, von welchem Datentyp das zu ladende Objekt ist*
DataKey	*Identifiziert das zu ladendende Datenobjekt*
TransactionId	*Ermöglicht die Zuordnung einer Antwortnachricht*

Die Antwortnachricht des Servers hat den folgenden Inhalt:

ObjectLoadedMessage	
Target = Client-URL	*Gibt den Client an, an den die Nachricht übertragen wird*
Sender = Server-URL	
Object	*Das geladene Objekt*
TransactionId	*Zuordnungsmöglichkeit für den Client zur Anfragenachricht*

Abbildung 80 beschreibt den Ablauf des Nachrichtenaustauschs beim Laden eines Objekts von einem Daten-Provider.

Beim Aufruf der Methode **LoadObject()** der Klasse ClientApplication (A in Abbildung 80) wird eine Nachricht vom Typ LoadObjectMessage erstellt. In dieser Nachricht werden die Informationen über das zu ladende Datenobjekt an den Server durch den Aufruf der Methode **TransferMessage()** übertragen (B in Abbildung 80). Da die Eigenschaft **Target** dieser Nachricht die URL des Servers beinhaltet, wird die Nachricht nicht an andere Clients weitergeleitet, sondern vom Server selbst verarbeitet. Um zu verhindern, dass die Verarbeitung der Nachricht den Server für weitere Nachrichten blockiert, wird für die Verarbeitung ein eigener Prozess gestartet (C in Abbildung 80). Innerhalb dieses Prozesses wird anhand der **DataType**-Eigenschaft der Nachricht der erste registrierte Daten-Provider gesucht, der Daten dieses Typs zur Verfügung stellt. Von diesem Daten-Provider wird über dessen Web Service das Objekt mit dem gesuchten **DataKey** in einem synchronen Aufruf der Methode **LoadObject()** angefordert (D in Abbildung 80). Anschließend wird das abgerufene Objekt in einer Antwortnachricht vom Typ ObjectLoadedMessage in der Eigenschaft **Object** gespeichert. Die **Target**-Eigenschaft wird auf die URL des Clients gesetzt, der die Nachricht vom Typ LoadObjectMessage an den Server übertragen hat. Die Eigenschaft **TransactionId** wird auf denselben Wert wie in der LoadObjectMessage gesetzt. Somit kann der Client diese Antwortnachricht seinem vorherigen Aufruf zuordnen. Dies ist nötig, da der gesamte Vorgang für den Client asynchron abläuft.

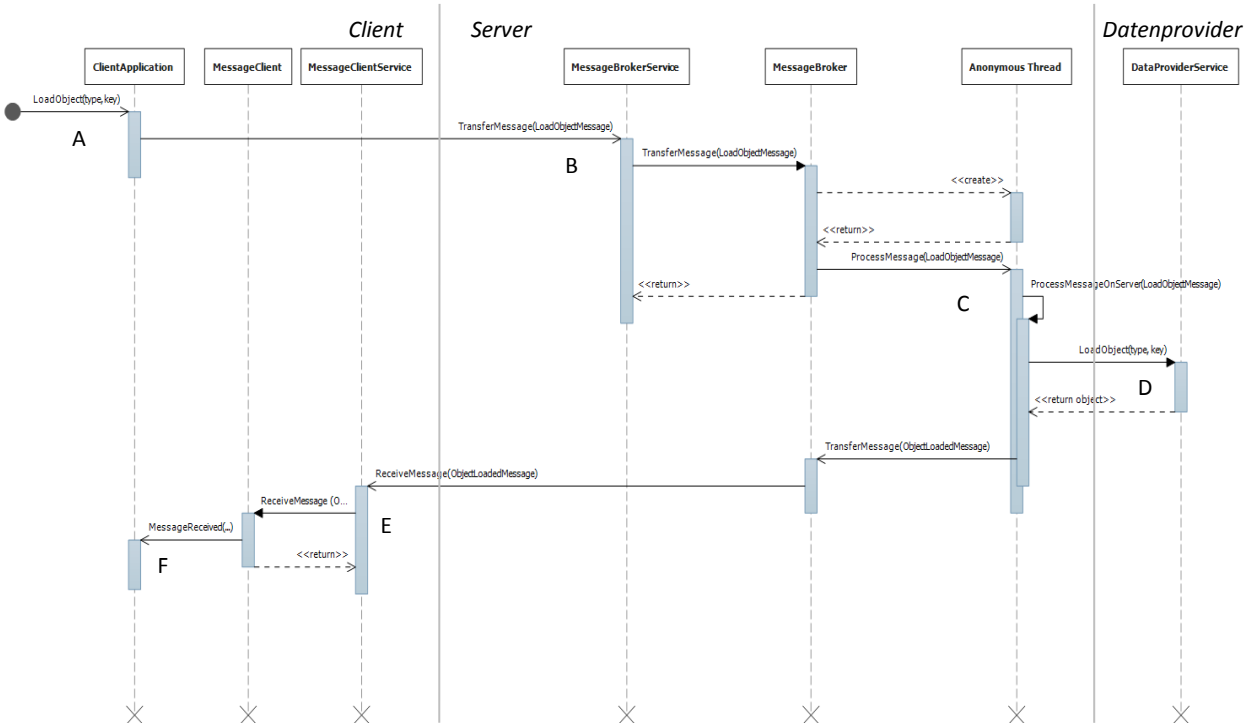


Abbildung 80: Sequenzdiagramm zum Laden eines Datenobjekts von einem Daten-Provider. Der VisualizationClient sendet eine Nachricht an den MessageBroker des Servers, der beim Daten-Provider das geforderte Objekt abruft und über eine Antwortnachricht an den VisualizationClient zurücküberträgt

Über den Web Service MessageClientService des aufrufenden Clients wird die Antwortnachricht an diesen zurückübertragen (E in Abbildung 80). Über das Event MessageReceived erhält die Klasse ClientApplication die Antwortnachricht (F in Abbildung 80) und kann diese verarbeiten und an die Implementierung des Visualisierungs-Clients weitergeben.

7 Prototyp

Dieses Kapitel beischreibt Aufbau und Implementierung des Prototyps zur Interaktion mit Powerwall-Visualisierungen, der im Rahmen dieser Diplomarbeit erstellt wurde. Der Prototyp implementiert eine Analyseumgebung für Eyetracking-Daten und basiert dabei strukturell auf der in Kapitel 6 vorgestellten Architektur. Der Interaktion mit der Powerwall, die in diesem Prototyp Anwendung findet, liegt das Interaktionskonzept aus Kapitel 5 zugrunde.

Im Folgenden werden die einzelnen Komponenten des Prototyps anhand der Architektur aus Kapitel 6 dargestellt: Zunächst werden der technische Aufbau und die im Prototyp verwendeten Komponenten beschrieben. Anschließend werden die Bestandteile der Benutzeroberfläche, sowie die Implementierung des Daten-Providers, des Visualisierungs-Clients und der Gestenerkennung vorgestellt. Abschließend wird die Erweiterung des Prototyps zu einer multimodalen Interaktionsumgebung beschrieben.

7.1 Technischer Aufbau

Der Prototyp wurde für die Powerwall im VR-Labor des Informatikgebäudes der Universität Stuttgart konzipiert und dort aufgebaut. Das VR-Labor bietet im Gegensatz zum Hörsaal V38.02 genügend Platz zur Gesteninteraktion vor der Powerwall. Ebenso kann die Powerwall im VR-Labor direkt von einem Computer angesteuert werden. Für die Powerwall im VISUS-Gebäude ist eine Schnittstelle zum Rechnercluster des Steuerrechners nötig. Es wurde zunächst eine Hälfte der Powerwall im VR-Labor zur Erstellung des Prototyps verwendet. Die verwendete Anzeigefläche der Powerwall ist 2,55 m breit und 1,87 m hoch und besitzt eine Auflösung von 1920 x 1200 Pixeln.

Zur Gestenerkennung wird ein Microsoft Kinect-Sensor eingesetzt. Die Verwendung eines Kinect-Sensors wurde gewählt, da die Kinect eine Interaktion ohne den Einsatz von Markern ermöglicht. Die Personen- und Skeletterkennung ist bereits in die Hardware des Sensors integriert.
Der Kinect-Sensor ist mittig auf einer Höhe von 1,60m über dem Boden angebracht. Da die Kinect somit in die Anzeigefläche der Powerwall hineinragt, ist die Aufhängung, mit der die Kinect am Rahmen der Powerwall befestigt ist, aus durchsichtigem Plexiglas gefertigt (siehe Abbildung 81, links). Die Nah- und Fern-Interaktionsbereiche sind zur Orientierung des Benutzers auf dem Fußboden vor der Powerwall markiert. Der Nah-Interaktionsbereich beginnt ab einem Abstand von 90 cm zur Powerwall. Ab 120 cm beginnt der Fern-Interaktionsbereich (siehe Abbildung 81, rechts).

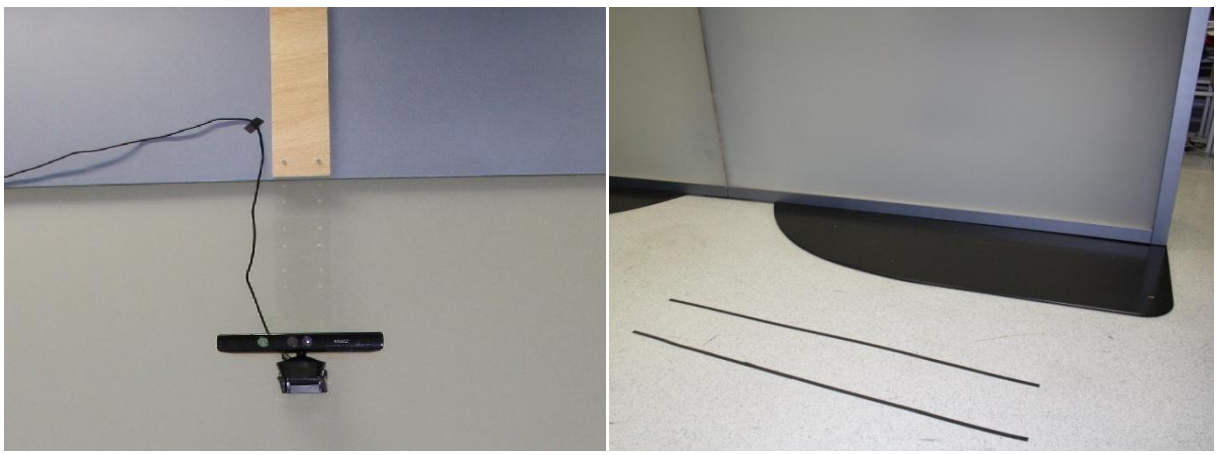


Abbildung 81: Mit einer Plexiglasplatte ist der Kinect-Sensor auf einer Höhe von 1,60 m über dem Boden befestigt (links). Die Interaktionsbereiche vor der Powerwall sind durch Klebebandstreifen am Fußboden markiert (rechts). Ab 90 cm Abstand zur Powerwall beginnt der Nah-Interaktionsbereich, ab 120 cm der Fern-Interaktionsbereich.

Die Powerwall und der Kinect-Sensor sind mit einem Computer verbunden, auf dem eine Instanz des Visualisierungs-Clients für die Powerwall installiert ist, wie er in Kapitel 7.3 beschrieben wird. Der Computer steuert die Visualisierungen auf der Powerwall und verarbeitet die Gesteninteraktionen des Benutzers. Gleichzeitig ist der Computer mit einem Netzwerk verbunden, über das die verschiedenen Komponenten der multimodalen Interaktionsumgebung mithilfe des Kommunikationsframeworks kommunizieren.

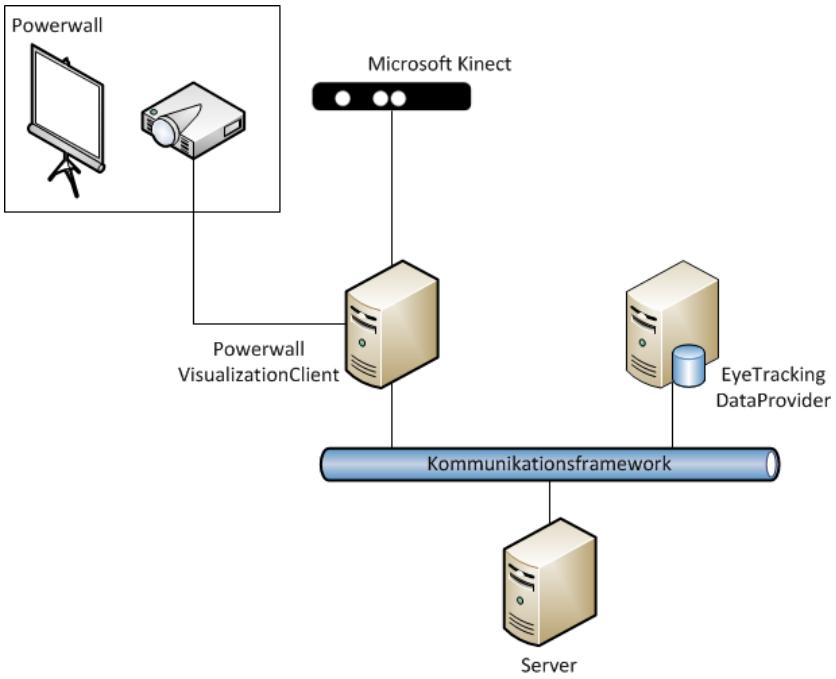


Abbildung 82: Struktureller Aufbau des Prototyps zur Visualisierung und Interaktion mit Eyetracking-Daten. Über das Kommunikationsframework kommunizieren die Serverkomponente, der Eyetracking-Daten-Provider und der Visualisierungs-Client. An den Computer, auf dem der Visualisierungs-Client installiert ist, sind die Powerwall und ein Microsoft Kinect-Sensor angeschlossen.

7.2 Benutzungsoberfläche

Die Benutzungsoberfläche für die Darstellung von Powerwall-Visualisierungen und die Interaktion über die Gestenerkennung wurde in einer WPF-Anwendung realisiert. Diese Anwendung unterstützt das Analyseszenario für Eyetracking-Daten, wie es in Kapitel 4.3.1 vorgestellt wird.
Zunächst werden die implementierten Interkationen und das Schatteninterface mit den Menü-Items vorgestellt. Anschließend werden die Visualisierungen für Eyetracking-Daten beschrieben.

Interaktionen

Folgende Interaktionen aus dem Konzept aus Kapitel 5.2.1 sind implementiert: *Ansicht verschieben*, *Ansicht skalieren*, *Ansicht löschen*, *Ansicht duplizieren*, *Ansichten koppeln*, *Visualisierung wechseln* und *Inhalt manipulieren*.

Zur Erkennung, wann eine Interaktion begonnen werden soll, wird die kombinierte Variante aus Dwell-Time und Greifgesten verwendet (siehe Kapitel 5.2.2). Im Fern-Interaktionsbereich kann eine Interaktion begonnen werden, nachdem der Benutzer eine Hand für eine bestimmte Zeit über dem Objekt ruhen lässt, mit dem er interagieren möchte. Im Nah-Interaktionsbereich werden durch die Metapher des „Zugreifens“ Objekte zur Interaktion ausgewählt.

Die Anwendung nimmt die gesamte Anzeigefläche der Powerwall ein. Am linken Bildrand wird der Daten-Explorer angezeigt. Der Daten-Explorer stellt die verfügbaren Daten aus dem Eyetracking-Datenmodell durch Vorschaubilder dar. Aus dem Daten-Explorer können die Vorschaubilder mit der Geste „Ansicht verschieben“ auf die Interaktionsfläche der Powerwall verschoben und anschließend weiter manipuliert werden.

Schatteninterface

Zur Interaktion mit den Visualisierungen auf der Powerwall ist ein Schatteninterface aus Kapitel 5.1.3 implementiert. Das Schatteninterface stellt das Abbild des Benutzers dar, wie es im dem Tiefenbild des Kinect-Sensors erkannt wird. Der Schatten wird halbtransparent dargestellt, sodass die darunter liegenden Powerwall-Visualisierungen immer sichtbar sind.

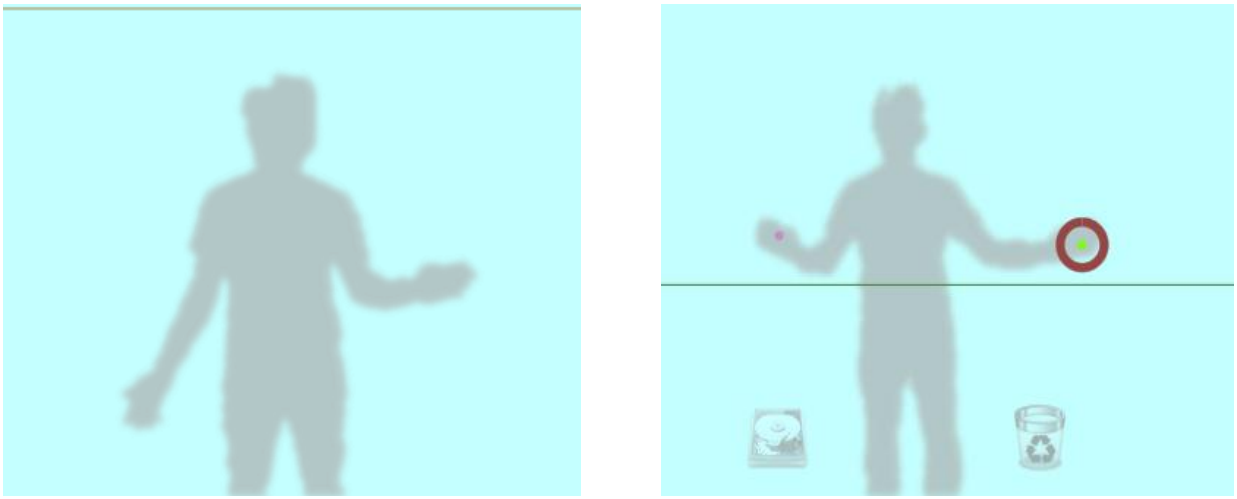


Abbildung 83: Schatteninterface für den Prototyp zur Analyse von Eyetracking-Studien. Links ohne Interaktionsn, rechts mit erkannter Interaktion und Hilfestellungen.

Der Prototyp des Schatteninterfaces verfügt lediglich über ein Menü-Objekt mit einem Papierkorb-Symbol. Das Papierkorb-Symbol erscheint rechts vom Schatten des Benutzers auf Hüfthöhe, sobald der Benutzer eine Interaktion beginnt. Bewegt der Benutzer eine Ansicht in die Nähe des Papierkorbs, wird er mit einem grünen Leuchten als aktiv markiert. Eine Ansicht wird von der Powerwall entfernt, sobald der Benutzer diese Ansicht mit der Interaktion „Ansicht verschieben“ auf dem aktivierten Papierkorb ablegt.

Abbildung 84: Papierkorb-Symbol des Schatteninterfaces. Links inaktiv, rechts aktiv. Wird eine Ansicht auf dem Papierkorb abgelegt, wird sie gelöscht.

Visualisierungen

Für die Darstellung der EyeTracking-Daten (siehe Kapitel 7.3.1) sind drei Visualisierungen als Plugins implementiert: Visualisierung des Stimulus als Bild, sowie Darstellung der Fixationen auf einem Stimulus als HeatMap oder Gaze-Duration-Sequence-Diagramm.

- **Stimulus-Visualisierung**
  In dieser Visualisierung wird ein Stimulus als Grafik dargestellt. Es sind keine Interaktionen innerhalb dieser Visualisierung möglich.

- **Heatmap-Visualisierung**
  Diese Visualisierung stellt einen Stimulus in einer Heatmap dar. Im Hintergrund der Visualisierung wird der Stimulus als Grafik dargestellt. Darüber werden Hitzefelder gelegt. Stellen, an denen viele Fixationen durch Probanden gemacht wurden, werden rot dargestellt. Bereiche mit wenigen Fixationen werden grün dargestellt. Dazwischen wird über orange und gelb abgestuft. Innerhalb dieser Visualisierung sind keine Interaktionen möglich.

- **Gaze-Duration-Sequence-Diagramm**
  In dieser Visualisierung wird ein Stimulus als ein Gaze-Duration-Sequence-Diagramm dargestellt. Die vertikalen Achsen können sortiert werden. Es werden zunächst die Daten aller Probanden angezeigt. Auf der rechten Seite wird eine Liste aller Probanden angezeigt. Aus dieser Liste können Probanden entfernt werden, indem sie auf das Papierkorb-Menü-Item des Schatteninterfaces gezogen werden. Wird ein Proband aus der Liste entfernt, werden ebenfalls dessen Daten aus dem Gaze-Duration-Sequence-Diagramm entfernt.

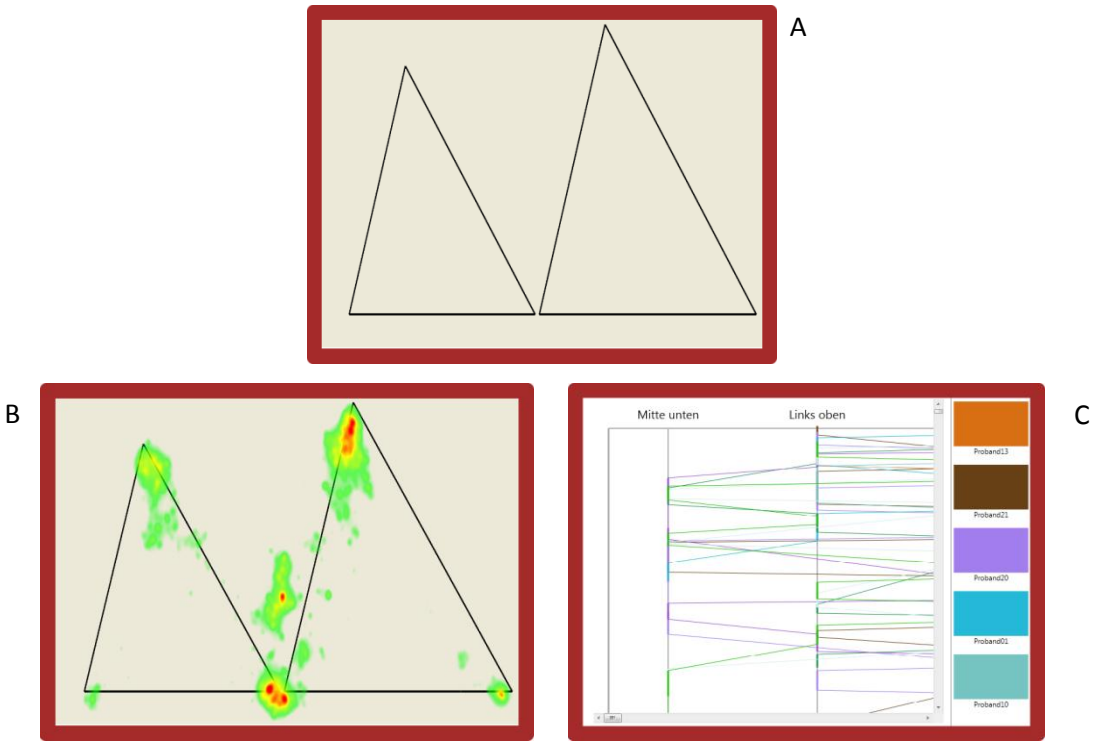


Abbildung 85: Beispielvisualisierungen für Eyetracking-Daten. Bilddarstellung des Stimulus (A), Darstellung der Fixationen auf einem Stimulus in einer Heat-Map (B) und einem Gaze-Duration-Sequence-Diagramm (C).

7.3 Implementierung

Dieses Kapitel stellt die Implementierung des Prototyps vor. Zuerst wird die Umsetzung des Daten-Providers beschrieben, der Eyetracking-Daten aus einer SQLServer-Datenbank über das Kommunikationsframework zur Verfügung stellt. Hierzu wird ein Datenmodell für Eyetracking-Daten entwickelt. Anschließend wird der Visualisierungs-Client vorgestellt, der Daten aus dem Eyetracking-Datenmodell auf der Powerwall visualisiert, sowie die Realisierung der Gestenerkennung beschrieben. Zum Abschluss wird eine Möglichkeit zur Erweiterung des Prototyps zu einer multimodalen Interaktionsumgebung dokumentiert.

7.3.1 Daten-Provider

Dieser Abschnitt beschreibt die Datenkomponente des Prototyps. Die Datenkomponente realisiert die Verwendung von Eyetracking-Daten in der multimodalen Interaktionsumgebung. Zunächst wird das Datenmodell kurz vorgestellt, anschließend wird die Einbettung des Datenmodells in die Architektur der Interaktionsumgebung beschrieben.

Eyetracking-Datenmodell

Eine detaillierte Beschreibung zur Eyetracking-Methode liefert die Diplomarbeit „Eye-Tracking-basiertes Analysekonzept zur Evaluation von Visualisierungen“ (Blascheck, 2012). Die Eyetracking-Daten werden von einem Tobii-Eyetracking-Gerät (Tobii Technology AB, 2011) generiert und mithilfe einer Import-Funktion in das Eyetracking-Datenmodell importiert.

In Abbildung 86 ist das Schema des Eyetracking-Datenmodells dargestellt. Kästen in der Abbildung entsprechen einer Objektklasse in C# und gleichzeitig einer Tabelle in einer SQLServer-Datenbank. Pfeile zwischen zwei Kästen beschreiben eine 1:N-Beziehung zwischen den Objektklassen bzw. den Tabellen in der SQLServer-Datenbank. Bezeichnungen innerhalb eines Kastens entsprechen Eigenschaften der Objektklassen bzw. Spalten in der entsprechenden Tabelle der SQLServer-Datenbank. Eigenschaften, die mit einem Schlüsselsymbol gekennzeichnet sind, werden in der SQLServer-Datenbank als Primärschlüssel verwendet.

Eyetracking-Daten werden in Projekten organisiert. Hierzu stehen die beiden Klassen SuperProject und Project zur Verfügung. Ein Superprojekt kann aus mehreren Projekten zusammengesetzt sein. Projekte bestehen wiederum aus mehreren Szenarien, welche in der Klasse Scanario gespeichert werden. Zu einem Szenario gehören Stimuli, also Bilder oder Grafiken, die im Rahmen eines Eyetracking-Projekts untersucht werden sollen. Diese Medien werden mit Metainformationen wie der Position und Größe der Grafik in der Klasse Stimulus gespeichert. Innerhalb eines Stimulus können bestimmte Bereiche für die Untersuchung von besonderer Bedeutung sein. Diese Bereiche werden in der Klasse AreaOfInterest zum jeweiligen Stimulus gespeichert.

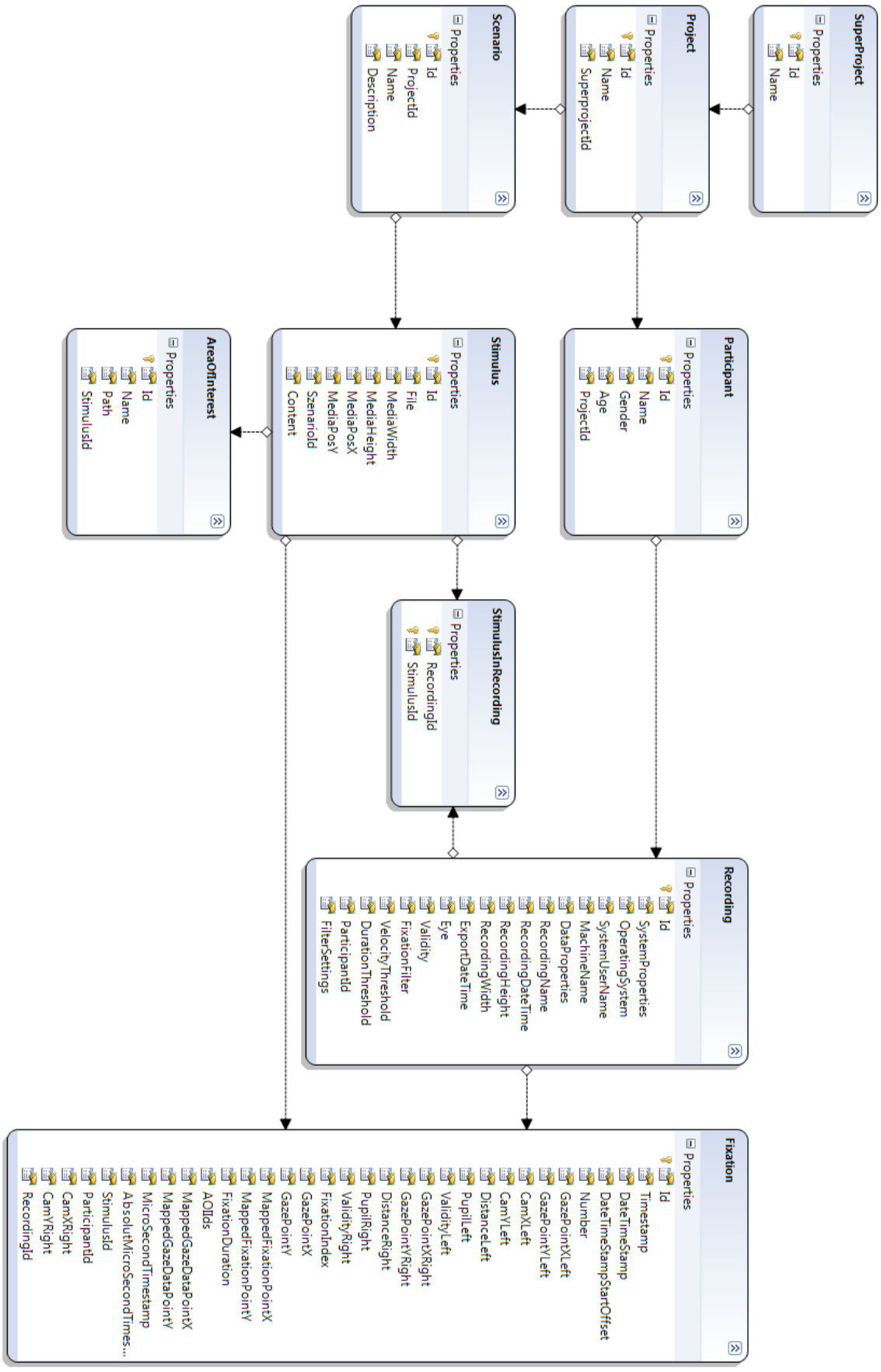


Abbildung 86: Eyetracking Datenmodell für den Daten-Provider des Prototyps. Die Pfeile zwischen den Klassen stellen 1:N-Beziehungen dar.

An einem Eyetracking-Projekt nehmen beliebig viele Probanden Teil. Informationen über die Probanden werden in der Klasse Participant gespeichert und dem jeweiligen Projekt zugeordnet. Wird die Untersuchung der Stimuli eines Szenarios mit einem Probanden durchgeführt, werden Informationen zu dieser Untersuchung in einer Instanz der Recording-Klasse gespeichert. Da Probanden ein Szenario während einer Untersuchung mehrfach durchlaufen können, wird über die Klasse StimulusInRecording eine N:M-Beziehung zwischen den Klassen Stimulus und Recording realisiert. Die Klasse Recording enthält zusätzlich Informationen über die Hardwarekonfiguration, mit der die Untersuchung durchgeführt wurde.

Kerninformationen für das Eyetracking bilden die Fixationen. Fixationen beschreiben Positionsangaben von Blickpunkten eines Probanden auf einem Stimulus, sowie die Verweildauer der Blickpunkte auf einer Stelle. Fixationen werden in der Fixation-Klasse im Datenmodell gespeichert. Jede Fixation wird dem Stimulus zugeordnet, auf dem die Fixation stattgefunden hat. Ebenso wird jede Fixation der jeweiligen Untersuchung über die Recording-Klasse zugeordnet.

Das Eyetracking-Datenmodell ist im Namespace *Vis.PowerInteractions.EyeTracking.Model* implementiert.

Eyetracking-DataProvider

Dieser Abschnitt beschreibt den Teil des Prototyps, der eine DataProvider-Komponente implementiert, wie sie in Kapitel 6.3.2 vorgestellt wurde. Diese DataProvider-Komponente stellt über das Kommunikationsframework Daten aus dem Eyetracking-Datenmodell für die multimodale Interaktionsumgebung des Prototyps zur Verfügung. Der Eyetracking-DataProvider wurde im Namespace *Vis.PowerInteractions.EyeTracking.DataProvider* implementiert und besteht aus zwei Klassen: der Klasse EyeTrackingDataProvider und der Klasse Program (vergleiche Abbildung 87).

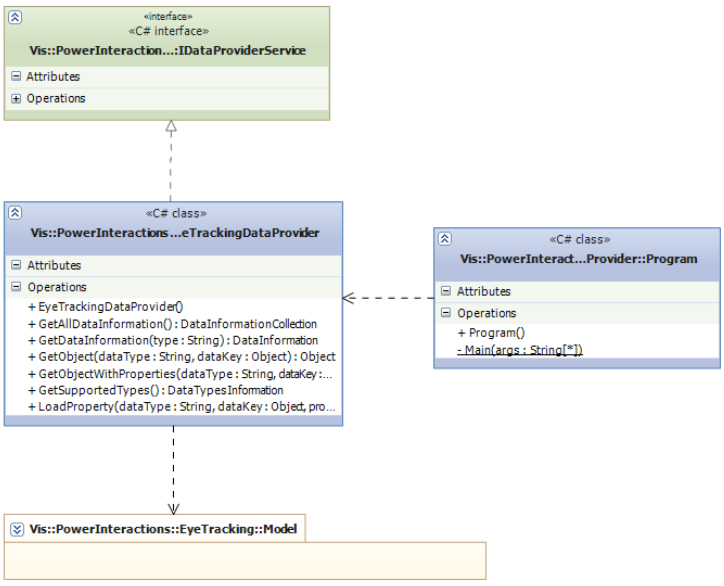


Abbildung 87: Klassendiagramm des Daten-Providers für Eyetracking-Daten. Im Programm wird ein Service erstellt, der über die Schnittstelle IDataProviderService Daten aus dem Eyetracking-Datenmodell zur Verfügung stellt.

Der Eyetracking-DataProvider greift auf eine SQLServer-Datenbank zu, deren Schema dem Eyetracking-Datenmodell (siehe Kapitel 7.3.1) entsprechen muss. Die Verbindung zur Datenbank

erfolgt über das Microsoft ADO.NET EntityFramework (Microsoft, 2012) und kann in der Anwendungskonfigurationsdatei eingestellt werden.

```
<add name="EyeTrackingEntities"
     connectionString="metadata=res://*/EyeTracking.csdl|
                               res://*/EyeTracking.ssdl|
                               res://*/EyeTracking.msl;
                       provider=System.Data.SqlClient;
                       provider connection string=&quot;
                               data source=(local)\SQLEXPRESS;
                               initial catalog=EyeTracking;
                               integrated security=True;
                               multipleactiveresultsets=True;
                               App=EntityFramework&quot;"
      providerName="System.Data.EntityClient"/>
```

Quellcode 5: Verbindungszeichenfolge in der Anwendungskonfigurationsdatei für das ADO.NET EntityFramework zum Zugriff auf die Eyetracking-Daten in einem SQL Server.

Die Klasse EyeTrackingDataProvider (siehe Abbildung 87) implementiert die Schnittstelle IDataProviderService und veröffentlicht über diese Schnittstelle die Daten für das Kommunikationsframework der multimodalen Interaktionsumgebung. Im Folgenden wird die Implementierung der Methoden der IDataProviderServide-Schnittstelle erläutert.

- **GetSupportedTypes()**
  Wird auf dem Eyetracking-DataProvider die Methode **GetSupportedTypes()** aufgerufen, wird eine Auflistung der Namen aller Klassen des Eyetracking-Datenmodells zurückgegeben. Eine Ausnahme bildet die Klasse StimulusInRecording, da diese Klasse lediglich eine N:M-Relation darstellt.
  Es werden voll qualifizierende Namen zurückgegeben. Zum Beispiel: „*Vis.PowerInteractions.EyeTracking.Model*.Project“.

- **GetDataInformation(string)**
  Diese Methode gibt Übersichtsinformationen über alle gespeicherten Objekte einer Klasse aus dem Eyetracking-Datenmodell zurück. Beim Aufruf dieser Methode wird als Parameter der voll qualifizierende Name einer Datenklasse aus dem Eyetracking-Datenmodell erwartet, andernfalls wird ein Fehler zurückgegeben.
  Als Rückgabewert dieser Methode wird ein Objekt, das eine Auflistung der Primärschlüssel aller gespeicherter Objekte der geforderten Datenklasse sowie den voll qualifizierenden Namen der Datenklasse zurückgibt.

- **GetAllDataInformation()**
  Diese Methode gibt Übersichtsinformationen über alle gespeicherten Objekte aller Klassen aus dem Eyetracking-Datenmodell zurück.
  Für jeden Klassennamen aus dem Rückgabewert der **GetSupportedTypes()**-Methode wird die Methode **GetDataInformation()** mit diesem Klassennamen aufgerufen. Die Übersichtsinformationen aller Klassennamen werden als Liste von der Methode **GetAllDataInformation()** zurückgegeben.

- **GetObject(**string, object**)**
  Über diese Methode kann ein Objekt aus der Datenbank abgerufen werden. Als erster

Parameter wird der voll qualifizierende Name einer Datenklasse aus dem Eyetracking-Datenmodell erwartet. Als zweiter Parameter wird der Primärschlüssel eines Objekts dieser Datenklasse erwartet, das in der SQLServer-Datenbank gespeichert ist. Sofern ein Objekt der angegebenen Datenklasse mit dem angegebenen Schlüssel existiert, wird dieses Objekt aus der Datenbank abgerufen und zurückgegeben. Es werden dabei keine Navigationseigenschaften geladen.
Wird beispielsweise ein Objekt vom Typ *Vis.PowerInteractions.EyeTracking.Model*.Project abgerufen, sind die Eigenschaften **Scenarios** und **Participants** leer, auch wenn zu diesem Projekt Szenarien und Probanden gespeichert sind. Dieses Vorgehen verringert das Datenvolumen, das über das Kommunikationsframework übertragen werden muss. Navigationseigenschaften müssen über die Methode **LoadProperty()** nachträglich abgerufen werden.

- **GetObjectWithProperties(**string, object, string[]**)**
  Mit dieser Methode kann ein Objekt aus der Datenbank abgerufen werden. Im Gegensatz zur Methode **GetObject()** kann als dritter Parameter eine Liste von Navigationseigenschaften angegeben werden, die ebenfalls abgerufen und in das geladene Objekt integriert werden. Wird zum Beispiel ein Objekt vom Typ *Vis.PowerInteractions.EyeTracking.Model*.Project abgerufen und im dritten Parameter wird „Scenarios“ übergeben, sind die Eigenschaft **Scenarios** im geladenen Projekt alle Szenarien enthalten, die zu diesem Projekt in der Datenbank gespeichert wurden. Die Eigenschaft **Participants** ist jedoch leer.

- **LoadProperty(**string, object, string**)**
  Diese Methode liefert eine Auflistung von Objekten zurück, die in einer Navigationseigenschaft eines anderen Objekts in der SQLServer-Datenbank gespeichert sind. Als erster Parameter wird der voll qualifizierende Name einer Datenklasse aus dem Eyetracking-Datenmodell erwartet. Als zweiter Parameter wird der Primärschlüssel eines Objekts dieser Datenklasse erwartet, das in der SQLServer-Datenbank gespeichert ist. Der dritte Parameter muss den Namen einer Navigationseigenschaft der Datenklasse aus dem Eyetracking-Datenmodell enthalten.
  Zunächst wird das Objekt der angegebenen Datenklasse mit dem angegebenen Primärschlüssel aus der Datenbank abgerufen. Anschließend wird der Wert der angegebenen Navigationseigenschaft aus dem Objekt herausgelöst und an den Aufrufer zurückgegeben.

Die Klasse Program (siehe Abbildung 87) wird in ein lauffähiges Programm kompiliert. In diesem Programm wird ein Web Service erstellt, über den die Serverkomponente des Kommunikationsframeworks auf die Methoden des Eyetracking-DataProviders zugreifen und Übersichtsinformationen und Objekte abrufen kann.
Nachdem dieser Web Service erstellt wurde, wird die Verbindung zur Serverkomponente aufgebaut. Die URL zur Serverkomponente kann in der Anwendungskonfigurationsdatei angegeben werden (siehe Quellcode 6):

```
<client>
      <endpoint
            address="http://localhost:11111/PowerInteractions/MessageBrokerService/"
            binding="wsHttpBinding"
            bindingConfiguration="..."
            behaviorConfiguration="..."
            contract="Proxy.IMessageBrokerService"
            name="WSHttpBinding_IMessageBrokerService">
</client>
```

Quellcode 6: Verbindung zur Serverkomponente in der Anwendungskonfigurationsdatei des Eyetracking-DataProviders.

7.3.2 Visualisierungs-Client

Für den Powerwall-Prototyp wurde ein VisualizationClient implementiert, wie er in Kapitel 6.3.3 beschrieben ist. In der WPF-Anwendung (siehe Kapitel 7.2) wird die Klasse ClientApplication aus dem ClientBackend verwendet, welche die Verbindung zur Server-Komponente und der Datenquelle (siehe Kapitel 7.3.1) über das Kommunikationsframework herstellt.

Die Visualisierungen Stimulus, HeatMap und Gaze-Duration-Sequence-Diagramm sind als Visualisierungs-Plugins realisiert. Die Plugins befinden sich in der Bibliothek *Vis.PowerInteractions.EyeTracking.PowerWall.Vis*. Diese Bibliothek liegt neben der erforderlichen visConfig.xml in einem Unterverzeichnis zur Anwendung. In der Anwendungskonfigurationsdatei wird diese Bibliothek referenziert und bei Programmstart nach Visualisierungen durchsucht.

```
<!-- List of all libraries that contain visualizations for this program -->
<VisualizationAssemblies>
  <Path>Plugins\Vis.PowerInteractions.EyeTracking.PowerWall.Vis\</Path>
</VisualizationAssemblies>
```

Quellcode 7: XML-Markup für die Visualisierungs-Plugins des Eyetracking-Prototyps. Die Visualisierungen werden beim Programmstart geladen.

Programmstart

Beim Start der Anwendung wird zunächst über die Methode **Connect()** des ClientBackends eine Verbindung zum Server des Kommunikationsframeworks aufgebaut und der Powerwall Visualisierungs-Client registriert (Abschnitt A in Abbildung 88). Anschließend werden von der Datenquelle die Übersichtsinformationen zu allen gespeicherten Stimuli abgerufen. Hierzu wird die Methode **GetDataInformation()** verwendet. Über die Serverkomponente werden die Übersichtsdaten vom Eyetracking-Daten-Provider angefordert und an den Powerwall Visualisierungsclient zurück übertragen (Abschnitt B in Abbildung 88). Dann werden die Bilder zu allen Stimuli von der Datenquelle über die Methode **GetObject()** geladen (Abschnitt C in Abbildung 88). Diese Bilder werden als Vorschau im Daten-Explorer angezeigt.
Sobald die Daten eines Stimulus geladen wurden, wird für diesen Stimulus eine neue Ansicht erstellt und mit den Daten des Stimulus verbunden. Dieser Vorgang wird über die Methoden **BroadcastNewView()** und **BroadcastNewData()** an alle verbundenen Geräte der multimodalen Interaktionsumgebung übermittelt (Abschnitt D in Abbildung 88).

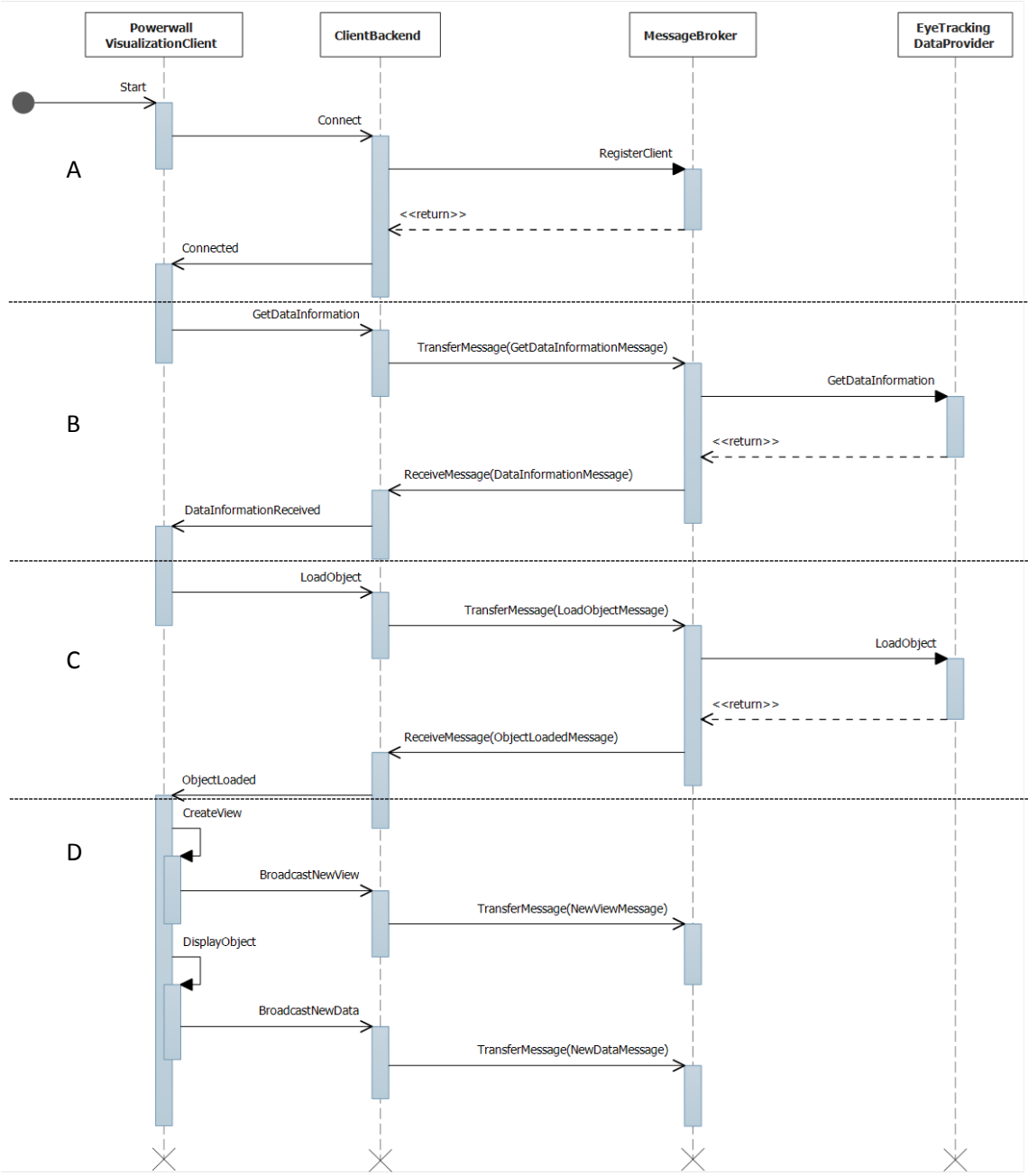


Abbildung 88: Programmablauf beim Start des Powerwall-Prototyps zur Analyse von Eyetracking-Studien. Zuerst wird eine Verbindung zum Server hergestellt (A), anschließend werden die Übersichtsinformationen über alle Stimuli vom Daten-Provider abgerufen (B) und schließlich die Stimuli abgerufen und auf der Powerwall angezeigt (C).

Lädt der Benutzer einen Stimulus aus dem Daten-Explorer, werden alle Daten über das Kommunikationsframework abgerufen, die zum Anzeigen der Visualisierungen benötigt werden. Dies sind:

- alle Fixationen, die zu diesem Stimulus gehören
- alle Recordings, in denen Fixationen zu diesem Stimulus erstellt wurde
- alle Probanden, die in Recordings diesen Stimulus betrachtet haben

Anschließend werden die Visualisierungen im Hintergrund vorbereitet, sodass sie in der Ansicht des Stimulus auf der Powerwall dargestellt werden können.

Interaktionen mit Powerwall-Visualisierungen

Alle Aktionen, die der Benutzer an der Powerwall ausführt, werden als Nachricht über das Kommunikationsframework an den Server übertragen. Dies erlaubt mehrere Instanzen des VisualizationClients auf verschiedenen Computern und Powerwalls zu starten und zu synchronisieren. Folgende Nachrichten werden verwendet:

- NewViewMessage
  Wird verwendet, nachdem eine neue Ansicht erstellt wurde, Beispielsweise nachdem der Benutzer eine Vorschau aus dem Daten-Explorer geladen hat. Diese Nachricht lässt alle verbundenen VisualizationClients dieselbe neue Ansicht erstellen.

- MoveViewMessage
  Wird verwendet, nachdem eine Ansicht auf der Powerwall verschoben wurde. Diese Nachricht teilt allen verbundenen VisualizationClients die neue Position einer Ansicht im auf der Powerwall mit, die vom Benutzer verschoben wurde.

- ResizeViewMessage
  Wird verwendet, nachdem die Größe einer Ansicht geändert wurde. Diese Nachricht teilt allen verbundenen VisualizationClients die neue Größe einer Ansicht mit, die vom Benutzer auf der Powerwall vergrößert oder verkleinert wurde.

- DuplicateViewMessage
  Wird verwendet, nachdem der Benutzer eine Nachricht dupliziert hat. Diese Nachricht teilt allen verbundenen VisualizationClients mit, welche Ansicht zu duplizieren ist.

- CoupleViewMessage
  Wird verwendet, nachdem der Benutzer zwei Ansichten miteinander verbunden hat. Diese Nachricht teilt allen verbundenen VisualizationClients mit, welche Ansichten zu verbinden sind.

- RemoveViewMessage
  Wird verwendet, nachdem der Benutzer eine Ansicht gelöscht hat. Diese Nachricht teilt allen verbundenen VisualizationClients mit, welche Ansicht zu entfernen ist.

- ChangeVisualizationMessage
  Wird verwendet, nachdem der Benutzer innerhalb einer Ansicht eine andere Visualisierung ausgewählt hat. Diese Nachricht teilt allen verbundenen VisualizationClients mit, in welcher Ansicht welche Visualisierung nun darzustellen ist.

- NewDataMessage
  Wird verwendet, nachdem innerhalb einer Ansicht Daten erstmals geladen wurden. Dies passiert beispielsweise nachdem beim Programmstart alle Stimuli aus der Datenquelle abgerufen und Ansichten zugeordnet wurden. Diese Nachricht teilt allen verbundenen VisualizationClients diejenigen Daten mit, die einer bestimmten Ansicht zugeordnet werden sollen.

- FilterDataMessage
  Wird verwendet, nachdem der Benutzer innerhalb einer Visualisierung die Menge der

anzuzeigenden Daten geändert hat. Dies passiert beispielsweise, nachdem der Benutzer aus einem Gaze-Duration-Sequence-Diagramm einen Probanden entfernt hat, dessen Fixationen nun nicht mehr im Diagramm angezeigt werden sollen. Diese Nachricht teilt allen verbundenen VisualizationClients mit, in welcher Ansicht eine Filterung vorgenommen werden soll.

7.3.3 Gestenerkennung

Dieser Abschnitt beschreibt, wie die Erkennung von Gesten mit dem Kinect-Sensor zur Interaktion mit Powerwall-Visualisierungen implementiert wurde. Zunächst wird die Unterscheidung zwischen Nah- und Ferninteraktion und anschließend die Erkennung von Interaktionen im Nah- und Fernbereich vorgestellt.

Unterscheidung Nah- und Fern-Interaktionsbereich

Zur Interpretation der von der Kinect erkannten Bewegungen des Benutzers und der Erkennung von Gesten wurde ein Zustandsautomat implementiert. Basierend auf den Skelettdaten und den Daten der Komponente zur Handerkennung werden die Übergänge zwischen den einzelnen Zuständen des Automaten ausgelöst. Werden bestimmte Zustände erreicht, die einer Geste entsprechen, werden die nötigen Events zur Interaktion mit der Benutzungsoberfläche ausgelöst. Zur logischen Trennung von Nah- und Fern-Interaktionsbereich wurde der Zustandsautomat in zwei Teilautomaten aufgeteilt. Abbildung 89 stellt den Zusammenhang zwischen den beiden Teilautomaten dar. Bewegt sich der Benutzer in einem größeren Abstand als 1,20 m zur Powerwall, ist der Zustand *Fern-Interaktion* aktiv und auf Hüfthöhe des Benutzers wird eine Hilfslinie zur Interaktion dargestellt. Bewegt sich der Benutzer auf die Powerwall zu und unterschreitet die Distanz von 1,20 m, wechselt der Automat in den Zustand *Nah-Interaktion*. Als Abstand des Benutzers zur Powerwall wird der Tiefenwert des Mittelpunkts zwischen den beiden Schultern des Benutzers verwendet. Dieser Wert wird in den Skelettdaten, die vom Kinect-Sensor ermittelt werden, mitgeliefert.

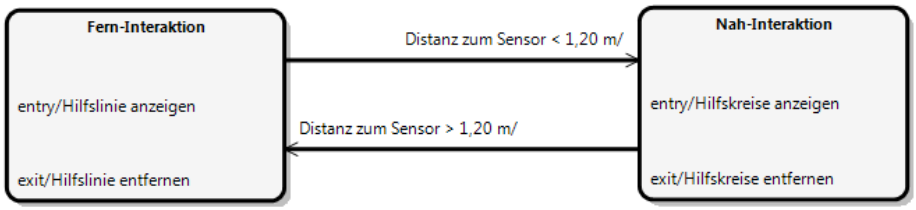


Abbildung 89: Zustandsautomat zur Unterscheidung zwischen Nah- und Ferninteraktion. Je nach Distanz zum Sensor wechselt der Zustandsautomat zwischen den Interaktionsbereichen *Fern-Interaktion* und *Nah-Interaktion*. In jedem Zustand werden entsprechende Hilfestellungen zur Interaktion angezeigt.

Abbildung 90 beschreibt den Teil des Automaten für die Fern-Interaktion und Abbildung 91 stellt den Teil für die Nah-Interaktion dar.

Interaktion im Fernbereich

Der Zustandsautomat zur Ferninteraktion (Abbildung 90) verfügt über die Zustände *Keine Interaktion*, *Clutching*, *Verschieben*, *Ansicht wechseln*, *Koppeln*, *Skalieren* und *Duplizieren*. Der Automat beginnt im Zustand *Keine Interaktion*. Während sich der Automat in diesem Zustand befindet, ist keine Interaktion mit Objekten auf der Powerwall möglich. Wird durch eine Abfolge von Transitionen ein Endpunkt erreicht, wechselt der Automat wieder zurück in den Zustand *Keine Interaktion*.

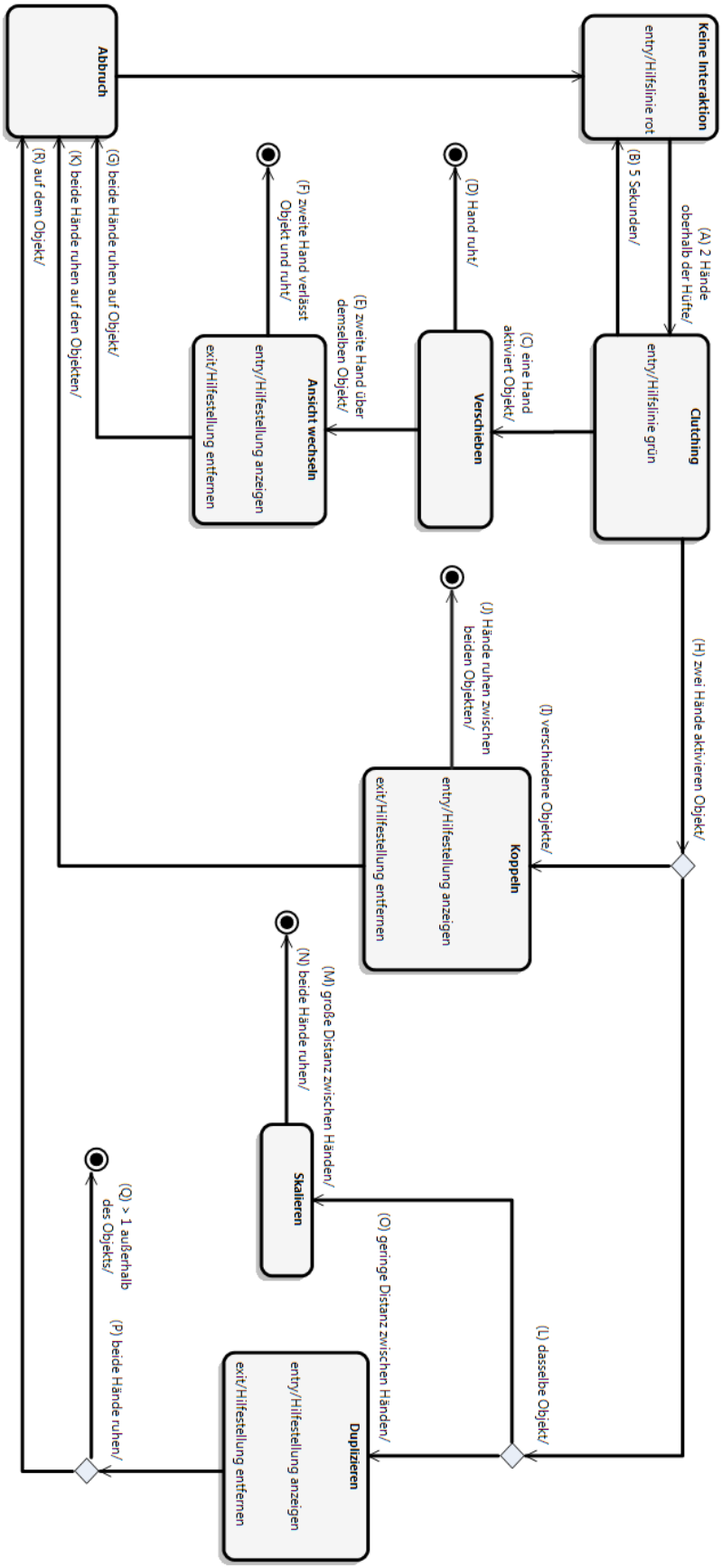


Abbildung 90: Zustandsautomat zur Erkennung von Gesteninteraktionen mit Powerwall-Visualisierungen im Fern-Interaktionsbereich.

Interaktionsbeginn
Hebt der Benutzer im Zustand *Keine Interaktion* beide Hände über die angezeigte Hilfslinie in Hüfthöhe (Transition A) wechselt der Automat in den Zustand *Clutching* und versucht Gesteninteraktionen zu erkennen. Startet der Benutzer keine Interaktion innerhalb von fünf Sekunden, wechselt der Automat zurück in den Ausgangszustand *Keine Interaktion* (Transition B).

Ansicht verschieben
Im Zustand *Clutching* erwartet der Zustandsautomat Interaktionen des Benutzers mit Objekten auf der Powerwall. Aktiviert der Benutzer mit einer Hand ein Objekt (Transition C), wechselt der Automat in den Zustand *Verschieben* und der Benutzer kann das aktivierte Objekt durch Bewegungen mit der Hand auf der Powerwall positionieren. Ruht die Hand über eine kurze Zeit an einer Position, wird die Interaktion beendet (Transition D).

Ansicht wechseln
Bewegt der Benutzer während des Verschiebens die zweite Hand über das dasselbe Objekt, wechselt der Zustandsautomat in den Zustand *Ansicht wechseln* (Transition E). Beim Aktivieren des Zustandes *Ansicht wechseln* werden die Vorschau-Hilfestellungen für die möglichen Ansichten eingeblendet und beim Verlassen wieder ausgeblendet. Lässt der Benutzer beide Hände für kurze Zeit innerhalb des Objekts ruhen, wird die Interaktion abgebrochen (Transition G). Verlässt der Benutzer mit einer Hand das Objekt, wird die Ansicht im Objekt gewechselt und die Interaktion beendet (Transition F).

Ansichten koppeln
Aktiviert der Benutzer mit beiden Händen Objekte, während sich der Automat im Zustand *Clutching* befindet (Transition H) und jede Hand aktiviert dabei ein anderes Objekt (Transition I), wechselt der Automat in den Zustand *Koppeln*. Dabei werden die Hilfestellungen zum Koppeln zweier Ansichten eingeblendet. Beim Verlassen des Zustands *Koppeln* werden die Hilfestellungen wieder entfernt. Bewegt der Benutzer beide Hände aufeinander zu und lässt sie in kurzem Abstand voneinander zwischen den aktivierten Objekten ruhen (Transition J), wird die Interaktion beendet. Ruhen beide Hände auf den jeweils aktivierten Objekten, wird die Interaktion abgebrochen (Transition K).

Ansicht skalieren
Führt der Benutzer beide Hände in dasselbe Objekt auf der Powerwall und aktiviert dieses (Transitionen H + L) entscheidet die Distanz beider Hände innerhalb des Objekts, welche Interaktion gestartet wird: Befinden sich die beiden Hände an gegenüberliegenden Ecken des Objekts (Transition M) wechselt der Automat in den Zustand *Skalieren* und der Benutzer kann durch Bewegen der Hände die Größe des Objekts anpassen. Lässt der Benutzer im Zustand *Skalieren* beide Hände kurzzeitig Ruhen, wird die Interaktion beendet (Transition N).

Ansicht duplizieren
Ist die Distanz beider Hände auf dem aktivierten Objekt klein, wechselt der Automat vom Zustand *Clutching* in den Zustand *Duplizieren* (Transition H + L + O). Beim Erreichen dieses Zustands werden die Hilfestellungen zum Duplizieren des Objekts angezeigt und beim Verlassen des Zustands wieder entfernt. Lässt der Benutzer im Zustand *Duplizieren* beide Hände innerhalb des aktivierten Objekts ruhen (Transition P + R), wird die Interaktion abgebrochen. Bewegt der Benutzer eine Hand aus dem Objekt heraus und lässt sie dort für kurze Zeit ruhen, wird die Interaktion abgeschlossen und das Objekt an der Stelle der Hand, die sich nach außen bewegt hat, dupliziert (Transition P + Q).

Interaktion im Nahbereich

Im Zustandsautomat für die Nah-Interaktion mit der Powerwall gibt es die Zustände *Keine Hand erkannt* und *Hand erkannt*. Wurde im Nah-Interaktionsbereich durch die Kinect eine Hand erkannt, so wird als Hilfestellung der Umriss eines Kreises an dieser Position dargestellt. Durch die Fingererkennung kann ermittelt werden, wann der Benutzer die Hand zu einer Faust schließt. Wird eine geschlossene Hand erkannt, so wird dieser Kreis farbig ausgefüllt.

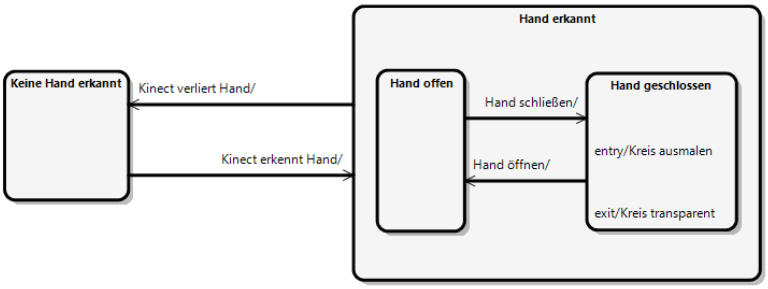


Abbildung 91: Zustandsautomat zur Erkennung von Gesteninteraktionen mit Powerwall-Visualisierungen im Nah-Interaktionsbereich.

7.4 Multimodale Interaktionsumgebung

Da der Prototyp mit der in Kapitel 6 vorgestellten Architektur implementiert ist, kann dieser Prototyp um weitere Geräte zur Interaktion mit Powerwall-Visualisierungen erweitert werden. Beispielweise können Tabletop-Computer wie der Microsoft Surface mit der Powerwall zu einer multimodalen Interaktionsumgebung kombiniert werden. Hierzu muss für den Tabletop-Computer ein spezieller Visualisierungs-Client erstellt werden. Eventuell können dabei die Visualisierungs-Plugins, die für die Darstellung der Eyetracking-Daten auf der Powerwall erstellt wurden, ebenfalls verwendet werden.

Über das Kommunikationsframework können die Instanzen der verschiedenen Visualisierungs-Clients untereinander austauschen und so beispielsweise ihre Darstellungen synchronisieren. Abbildung 92 zeigt den strukturellen Aufbau einer multimodalen Interaktionsumgebung mit einer Powerwall und einem Microsoft Surface.

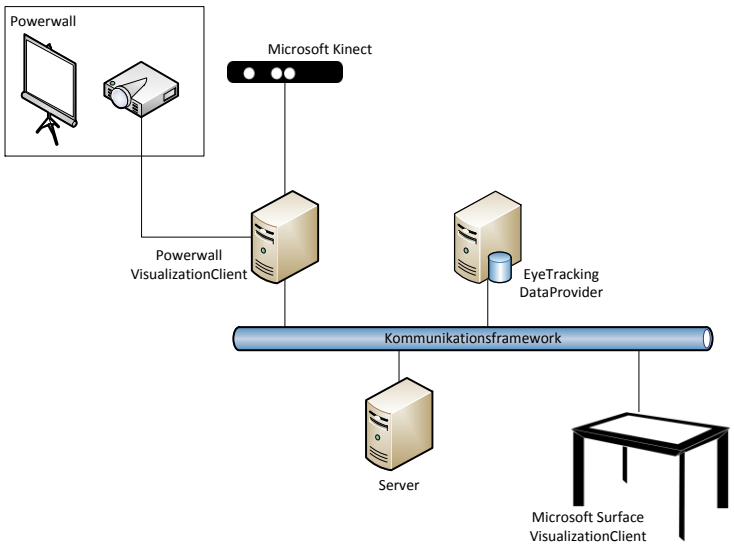


Abbildung 92: Struktureller Aufbau der multimodalen Interaktionsumgebung mit Powerwall-Interaktion und Tabletop-Computer. Über das Kommunikationsframework können sich Visualisierungs-Clients auf Tabletop-Computern mit dem Visualisierungs-Client der Powerwall austauschen.

8 Benutzerstudie

Im Rahmen dieser Diplomarbeit wurde eine Benutzerstudie zur Evaluation des Konzepts zur Interaktion mit Powerwall-Visualisierungen entwickelt und eine Pilotstudie durchgeführt. Dieses Kapitel beschreibt die Motivation zu dieser Studie und deren Aufbau. Anschließend werden die Durchführung und die Ergebnisse der Pilotstudie vorgestellt. Zuletzt werden die Ergebnisse der Pilotstudie diskutiert und Optimierungen für die Hauptstudie vorgeschlagen.

8.1 Motivation

Im Rahmen dieser Diplomarbeit wurde ein System zur Gesteninteraktion mit Powerwall-Visualisierungen erstellt. Durch eine Benutzerstudie soll herausgefunden werden, welche Gesten sich zur Interaktion mit Powerwall-Visualisierungen eignen. Diese Gesten sollen im Prototyp umgesetzt und später in einer weiteren Studie evaluiert werden.

Auf dem Gebiet der Gesteninteraktion mit berührungsempfindlichen Displays sind bereits zahlreiche Studien zu finden: Es wurden Studien zur Bewertung von speziellen Interaktionssystemen mit Touch-Eingaben (Soro, Iacolina, Scateni, & Uras, 2011) und Studien zum Vergleich mit den klassischen Eingabegeräten Tastatur und Maus durchgeführt (Matejka, Grossmann, Lo, & Fitzmaurice, 2009). Ebenfalls sind Studien entwickelt worden, in denen mithilfe von Probanden natürliche Gesten zur Durchführung von Standardaufgaben wie das Auswählen, Bewegen, Skalieren oder Löschen von Objekten gefunden werden sollten. ((Mauney, 2012) und (Wobbrock, Morris, & Wilson, 2009)).

Für die Freihandgesteninteraktion mit großen hochauflösenden Displays sind ebenfalls bereits Studien durchgeführt worden. In (Cabral, Morimoto, & Zuffo, 2005) wurden beispielsweise die Geschwindigkeit und Genauigkeit von Gesteninteraktionen und Mauseingaben verglichen. Spezielle Interaktionssysteme wie beispielsweise ein System zur Dokumentenverwaltung mit Gesten-Interface (Deller, et al., 2008) wurden ebenfalls in Studien bewertet.
Allerdings wurde noch keine Studie durchgeführt, die Gesten zur Interaktion mit Visualisierungen an hochauflösenden Displays findet und bewertet. Die Frage, welche Gesten Menschen bei der Interaktion mit Powerwalls als intuitiv empfinden und ohne darüber nachzudenken durchführen würden, ist noch unbeantwortet. Diese Studie soll intuitive Freihandgesten zur Durchführung grundlegender Aufgaben bei der Interaktion mit Visualisierungen auf großflächigen, hochauflösenden Displays identifizieren und nach ihrer Eignung zur Interaktion mit Powerwall-Visualisierungen bewerten.

Um die Gesteninteraktion mit Powerwall-basierten Visualisierungen geeignet evaluieren zu können, wird im Rahmen dieser Diplomarbeit eine Pilotstudie mit vier Probanden durchgeführt. Auf Basis dieser initialen Studie können qualitative Aussagen über intuitive Gesten zur Interaktion mit Powerwall-Visualisierungen getroffen werden. Darüber hinaus können weitere Forschungsideen auf dem Gebiet der Freihandgesteninteraktion mit großen, hochauflösenden Displays abgeleitet und Optimierungen des Studiendesigns für eine Hauptstudie nach Abschluss dieser Arbeit gefunden werden.

8.2 Methode

Die Studie wurde als ein within-subjects-Design mit zwei Abschnitten entworfen und als Pilotstudie mit vier Teilnehmern im VR-Labor des Instituts für Visualisierung und Interaktive Systeme der Universität Stuttgart durchgeführt. Während der Durchführung waren keine anderen Personen außer dem Probanden und dem Studienleiter anwesend.

Die Teilnehmer sollten im ersten Abschnitt zu Interaktionen mit Objekten auf der Powerwall Gesten finden, die den Teilnehmern selbst intuitiv erschienen. Eine Interaktion, zu der die Teilnehmer eine Geste vorschlagen sollten, war beispielsweise „Objekt auf Powerwall verschieben“ oder „Objekt vergrößern“. Zudem sollten die Teilnehmer die Schwierigkeit der von ihnen vorgeschlagenen Gesten einschätzen.

Im zweiten Studienabschnitt sollten die Probanden vordefinierte Gesten zur Interaktionen mit Powerwall-Objekten ausführen und bewerten. Hierfür waren beispielhaft Gesten für die Interaktionen aus im ersten Abschnitt implementiert worden. Nach der Durchführung einer Geste sollten die Teilnehmer die jeweilige Geste anhand eines Fragebogens bewerten.

8.2.1 Teilnehmer

An dieser Pilotstudie nahmen insgesamt vier Probanden teil. Drei der Probanden waren weiblich, einer männlich. Die Probanden waren im Alter von 17 bis 54 Jahren (Durchschnitt 29,75, Median 24). Als Vorkenntnisse in der Gesteninteraktion gaben drei der Probanden an eine Spielekonsole mit Gesten-Controller zu besitzen, die sie im Schnitt 1,75 Stunden pro Woche benutzen. Ebenfalls drei der vier Probanden gaben an, ein Touch-Gerät, Tablet oder touchfähiges Smartphone zu besitzen. Die durchschnittliche Nutzungsdauer beträgt neun Stunden pro Woche.

	Geschlecht	Alter (Jahre)	Spiele-konsolen	Nutzungsdauer / Woche (h)	Touch-Geräte	Nutzungsdauer / Woche (h)
Proband01	W	24	Ja	0	Ja	6
Proband02	W	54	Nein	0	Nein	0
Proband03	M	24	Ja	1	Ja	14
Proband04	W	17	Ja	6	Ja	16

Tabelle 2: Teilnehmer an der Qualitativen Pilotstudie zur Gestenfindung und Gestenbewertung. Neben demografischen Angaben wurde die Verwendung von Gesten erkennenden Spielekonsolen und Touch-Geräten erhoben.

8.2.2 Studienaufbau

Die Studie wurde im VR-Labor des Instituts für Visualisierung und Interaktive Systeme durchgeführt. Die Powerwall im VR-Labor verfügt über eine Gesamtauflösung von 3840 * 1080 Pixeln und eine Einrichtung zur Darstellung von dreidimensionalen Projektionen. Dazu wird die Powerwall von drei HD-Beamern von hinten beleuchtet. Für die Durchführung der Studie wurden die Aufgaben zur Interaktion mit Powerwall-Visualisierungen mithilfe eines Beamers auf einer Hälfte der Powerwall dargestellt.

Zur Darstellung der Powerwall-Visualisierungen wurde ein DELL XPS Notebook mit einem Intel Core i7 Prozessor, 8 GB RAM, 160GB SSD-Festplatte, HD-Auflösung und HDMI-Anschluss verwendet. Auf diesem Notebook wurde ein Prototyp ausgeführt, der das Interaktionskonzept implementiert und so die Interaktion mit einfachen Powerwall-Visualisierungen ermöglicht. Der Prototyp ist in Kapitel 7 beschrieben. Die Gestenerkennung erfolgte mit einem Microsoft Kinect for Windows Sensor, der in einer Höhe von 1,60 m über dem Boden vom Rahmen der Powerwall herabgehängt und mit dem Notebook verbunden wurde. Um die Sicht auf die Powerwall so wenig wie möglich einzuschränken, wurde der Sensor an einer durchsichtigen Plexiglasplatte befestigt (siehe Abbildung 93).

Zur Unterstützung der Probanden im zweiten Studienabschnitt wurden die im Interaktionskonzept vorgestellten Interaktionsbereiche (siehe Kapitel 5) mit Klebeband auf dem Boden markiert. Der Abstand der Klebebandstreifen betrug 90 cm und 120 cm zur Projektionsfläche der Powerwall.

Zwischen den beiden Markierungen befand sich der Nah-Interaktionsbereich und ab 120 cm Abstand zur Powerwall begann der Fern-Interaktionsbereich (siehe Abbildung 93).

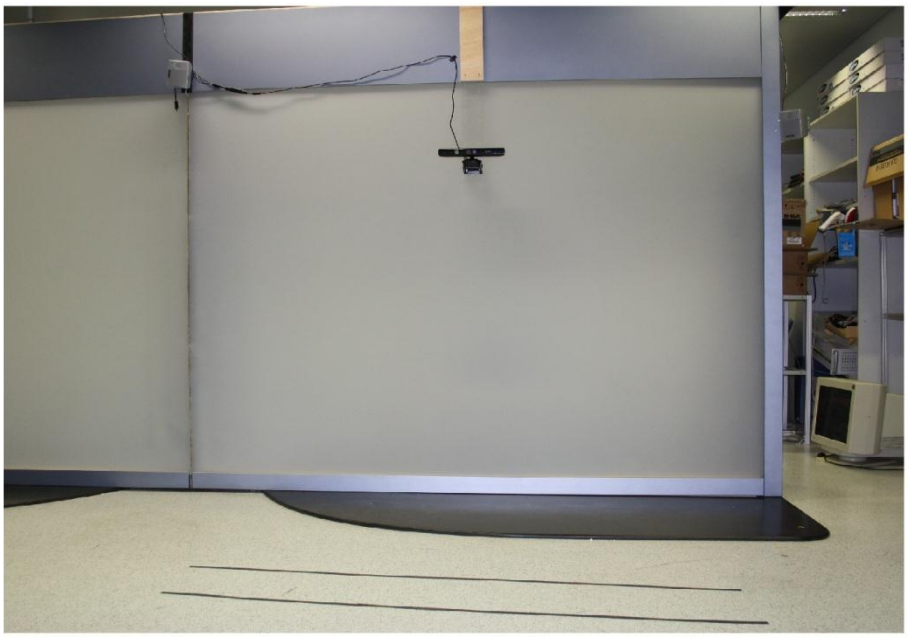


Abbildung 93: Versuchsaufbau der Benutzerstudie zur Findung und Bewertung von Gesteninteraktionen.

Im ersten Studienabschnitt wurden die Teilnehmer bei der Durchführung der Aufgaben gefilmt. Die Aufnahmen wurden mit einer Canon IXUS 800 IS Digitalkamera angefertigt. Während der Aufnahmen waren die Teilnehmer dazu aufgefordert nach der Methode des „lauten Denkens“ (van Someren, Barnard, & Sandberg, 1994) alle Gedanken und Tätigkeiten zu kommentieren und so Erfolge, Probleme oder Unstimmigkeiten mitzuteilen.

8.2.3 Aufgaben und Hypothesen

Die Studie war in zwei Abschnitte aufgeteilt. Im ersten Teil wurden den Teilnehmern Aufgaben gestellt, in denen sie Gesten zur Durchführung bestimmter Interaktionen mit abstrakten Powerwall-Visualisierungen entwickeln sollten. Im zweiten Studienabschnitt sollten die Teilnehmer die Gesteninteraktionen bewerten, die in der prototypischen Implementierung des Interaktionskonzepts dieser Diplomarbeit umgesetzt wurden (siehe Kapitel 5). Zu jedem dieser Abschnitte wurde eine Hypothese formuliert, die im Folgenden beschrieben werden.

Abschnitt 1: Gestenfindung

Im ersten Teil der Studie sollten die Teilnehmer selbst Gesten zur Interaktion mit Powerwall-Visualisierungen entwickeln. Insgesamt wurden sieben Aufgaben gestellt, in denen zu einer Interaktion eine oder mehrere Gesten gefunden werden sollten:

Aufgabe 1.1	Objekt verschieben
Aufgabe 1.2	Objekt löschen
Aufgabe 1.3	Objekt skalieren
Aufgabe 1.4	Objekt duplizieren
Aufgabe 1.5	Objekte koppeln
Aufgabe 1.6	Ansicht in einem Objekt wechseln
Aufgabe 1.7	Objektinhalt verändern

Tabelle 3: Aufgaben im ersten Studienabschnitt. Zu diesen sieben Interaktionen sollten die Teilnehmer Gesteninteraktionen entwickeln.

Zu jeder dieser Aufgaben sollten die Teilnehmer angeben, wie kompliziert ihrer Vorstellung nach eine solche Geste durchzuführen ist und wie ermüdend die Durchführung einer solchen Geste auf Dauer sein würde.

Zusätzlich sollten die Teilnehmer eine Ordnung in den sieben Gesteninteraktionen (siehe Tabelle 3) finden und die Gesten nach der Eigenschaft „schwierig“ sortieren.

Mithilfe dieser Aufgaben aus dem ersten Studienabschnitt sollte folgende **Hypothese 1** überprüft werden:

> **Es gibt intuitive Gesten für bestimmte Freihand-Interaktionen mit Powerwall-Visualisierungen, die von den meisten Anwendern vorgeschlagen werden.**

Abschnitt 2: Beurteilung der Gesten im Prototyp

Im zweiten Studienabschnitt sollten die Teilnehmer die Gesten bewerten, die im Prototyp (siehe Kapitel 7) beispielhaft implementiert wurden. Diese Gesten realisieren ebenfalls die sieben Interaktionen, die in Tabelle 3 aufgelistet sind. Ebenfalls sollten die Konzepte „Interaktionsbereiche“, „Schatteninterface“ und „Hilfestellungen“ des Prototyps bewertet werden.

Zur Bewertung der einzelnen Gesten und Konzepte wurden jeweils folgende Kriterien herangezogen:

- **Passend**: Wieweit stimmen Sie zu, dass diese Geste passend für die Durchführung der Aufgabe ist?
- **Intuitiv**: Wieweit stimmen Sie zu, dass Sie diese Geste intuitiv ausgeführt hätten, um die Aufgabe zu lösen?
- **Leicht zu lernen**: Wieweit stimmen Sie zu, dass diese Geste leicht zu erlernen ist?
- **Kompliziert:** Wieweit stimmen Sie zu, dass diese Geste zu kompliziert für die Aufgabe ist und es eine einfachere Alternative gäbe?
- **Umständlich**: Wieweit stimmen Sie zu, dass die Durchführung dieser Geste zu umständlich ist?
- **Peinlich:** Wieweit stimmen Sie zu, dass Ihnen die Durchführung dieser Geste zu peinlich ist und Sie diese Geste daher ungern anwenden würden?

Zusätzlich wurden folgende Variablen aus dem NASA TLX (Hart, 2006) ausgewählt, Die Übersetzungen ins Deutsche stammen von (Vertanen):

- **Mental Demand**: Wie viel geistige Anforderung war bei der Informationsaufnahme und bei der Informationsverarbeitung erforderlich (z.B. Denken, Entscheiden, Rechnen, Erinnern, Hinsehen, Suchen ...)? War die Aufgabe leicht oder anspruchsvoll, einfach oder komplex, erfordert sie hohe Genauigkeit oder ist sie fehlertolerant?
- **Physical Demand**: Wie viel körperliche Aktivität war erforderlich (z.B. ziehen, drücken, drehen, steuern, aktivieren ...)? War die Aufgabe leicht oder schwer, einfach oder anstrengend, erholsam oder mühselig?

- **Performance**: Wie erfolgreich haben Sie Ihrer Meinung nach die vom Versuchsleiter (oder Ihnen selbst) gesetzten Ziele erreicht? Wie zufrieden waren Sie mit Ihrer Leistung bei der Verfolgung dieser Ziele?

- **Frustration**: Wie unsicher, entmutigt, irritiert, gestresst und verärgert (versus sicher, bestätigt, zufrieden, entspannt und zufrieden mit sich selbst) fühlten Sie sich während der Aufgabe?

Abschließend wurde die Bewertung der Usability des Gesamtsystems bestehend aus den implementierten Gesteninteraktionen und der Teilkonzepte „Interaktionsbereiche“, „Schatteninterface“ und „Hilfestellungen“ erhoben. Hierfür wurde der System Usability Scale (Brooke, 1996) verwendet.

Mithilfe dieser Aufgaben aus dem zweiten Studienabschnitt sollte folgende **Hypothese 2** überprüft werden:

> **Die im Prototyp implementierten Gesten zur Interaktion mit Powerwall-Visualisierungen sind intuitiv und eignen sich für die jeweilige Aufgabe.**

8.2.4 Studiendurchführung

Zu Beginn der Studie mussten die Probanden einen Fragebogen ausfüllen, der sie zu ihrem Alter und ihren Vorkenntnissen auf dem Bereich der Gesteninteraktion befragte (siehe Anhang A: Allgemeiner Fragebogen). Es wurde erhoben, ob die Teilnehmer Spielekonsolen besitzen, die mithilfe von Gesteninteraktion bedient werden können, Touch-geräte oder Smartphones mit Gesteninterface verwenden. Zusätzlich wurden die Teilnehmer vor Beginn der eigentlichen Aufgaben gefragt, ob sie der Videoaufzeichnung im ersten Studienabschnitt zustimmen.

Abschnitt 1: Gestenfindung

Zunächst wurden die Teilnehmer über Ablauf des ersten Studienabschnitts informiert. In einem Tutorial wurde das Vorgehen zur Bearbeitung der Aufgaben des ersten Studienabschnitts den Teilnehmern erklärt. Anschließend sollten die Teilnehmer die Aufgaben selbst durchführen.

Die Reihenfolge der **Aufgaben 1.1 bis 1.7** wurde vom Studienleiter zufällig gewählt, jedoch lief jede Aufgabe stets gleich ab. Zunächst wurde auf der Powerwall eine Beispielvisualisierung in einer Ausgangsposition dargestellt (siehe Abbildung 94). Anschließend beschrieb der Studienleiter die Manipulation des Objekts, die der Teilnehmer nun durchführen sollte. Bei der Beschreibung der durchzuführenden Manipulation wurde darauf geachtet, dass der Studienleiter keine Gesten mit seinen Händen durchführte. Damit sollte gewährleistet werden, dass der Teilnehmer nicht von den Gesten des Studienleiters beeinflusst wurde. Nach einem Tastendruck auf dem Notebook änderte sich die dargestellte Beispielvisualisierung so, dass das gewünschte Ergebnis auf der Powerwall angezeigt wurde. Nun erhielten die Probanden Zeit zu überlegen, mit welcher Geste sie die Visualisierung vom Ausgangszustand in den Zielzustand überführen könnten. Gleichzeitig wurde die Videoaufnahme gestartet, sodass alle laut ausgesprochenen Gedanken der Probanden aufgezeichnet werden konnten. Sobald sich die Teilnehmer eine Geste überlegt hatten, wurde durch einen erneuten Tastendruck auf dem Notebook wieder die Ausgangsdarstellung der Visualisierung angezeigt und die Teilnehmer führten die von ihnen erdachte Geste aus. Anschließend wurde die Videoaufnahme gestoppt.

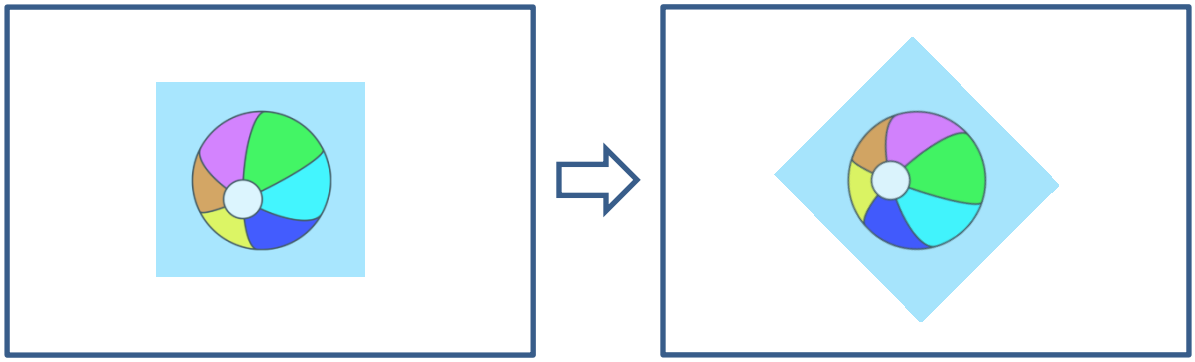


Abbildung 94: Ausgangsdarstellung der Beispielvisualisierung (links) und Zieldarstellung (rechts) bei der Interaktion "Rotieren".

Die Ausgangs- und Zieldarstellungen aller Aufgaben des ersten Studienabschnitts sind in Anhang C: Ausgangs- und Zieldarstellungen des ersten Studienabschnitts an diese Diplomarbeit angefügt.

Nachdem die Teilnehmer eine Aufgabe durchgeführt hatten wurden sie gebeten einen Fragebogen auszufüllen (siehe Anhang B: Fragebogen zur Gestenfindung). Die Teilnehmer sollten sie bewerten, wie kompliziert und ermüdend sie generell eine Geste einschätzen würden, welche die soeben durchgeführte Interaktion ermöglicht. Diese Bewertung wurde auf einer Skala von 1 (nicht kompliziert, nicht ermüdend) bis 20 (sehr kompliziert, stark ermüdend) erhoben.

Zum Abschluss des ersten Studienabschnitts sollten die Teilnehmer in **Aufgabe 1.8** alle sieben Interaktionen nach ihrer Schwierigkeit sortieren.

Abschnitt 2: Beurteilung der Gesten im Prototyp

Im zweiten Teil der Studie sollten die im Prototyp beispielhaft implementierten Gesten zur Interaktion mit abstrakten Powerwall-Visualisierungen evaluiert werden. Hierzu wurden den Teilnehmern zunächst die Interaktionskonzepte des Schatteninterfaces, der Interaktionsbereiche und der Hilfestellungen erklärt und demonstriert.

Anschließend wurden die **Aufgaben 2.1 bis 2.7** wurden ebenfalls in zufälliger Reihenfolge von den Probanden bearbeitet.

Aufgabe 2.1	Objekt verschieben
Aufgabe 2.2	Objekt löschen
Aufgabe 2.3	Objekt skalieren
Aufgabe 2.4	Objekt duplizieren
Aufgabe 2.5	Objekte koppeln
Aufgabe 2.6	Ansicht in einem Objekt wechseln
Aufgabe 2.7	Objektinhalt verändern

Abbildung 95: Interaktionsaufgaben im zweiten Studienabschnitt. Zu jeder Interaktion sollte die implementierte Geste bewertet werden.

Bevor die Teilnehmer eine Interaktion durchführen sollten, wurde der Prototyp in einen Ausgangszustand zurückversetzt, in dem eine Beispielvisualisierung dargestellt wurde. Anschließend wurde den Teilnehmern die Durchführung der Geste zur jeweiligen Aufgabe erklärt und einmal demonstriert. Dann wurde der Prototyp erneut zurückgesetzt und die Probanden führten die Gesteninteraktion selbstständig durch. Die Teilnehmer durften eine Interaktion beliebig oft wiederholen.

Nachdem die Teilnehmer jeweils eine Aufgabe durchgeführt hatten, sollten sie die soeben durchgeführte Geste auf einem Fragebogen bewerten (siehe Anhang D: Fragebogen zur Beurteilung des Prototyps).

Vor Studienende sollten die Teilnehmer noch die Teilkonzepte „Interaktionsbereiche“ (**Aufgabe 2.8**), „Schatteninterface“ (**Aufgabe 2.9**) und „Hilfestellungen“ (**Aufgabe 2.10**) nach denselben Kriterien wie die Gesteninteraktionen bewerten. Abschließend sollten die Teilnehmer das Gesamtsystem nach dem System Usability Scale (Brooke, 1996) bewerten (**Aufgabe 2.11**, siehe Anhang D: Fragebogen zur Beurteilung des Prototyps).

8.3 Ergebnisse

Die Studie wurde als Pilotstudie mit vier Teilnehmern durchgeführt. Die geringe Teilnehmerzahl lässt daher nur qualitative Aussagen zu. Dennoch sind richtungsweisende Ergebnisse zu erwarten, die einen Eindruck über mögliche Gesten und das implementierte Interaktionskonzept vermitteln können.

8.3.1 Abschnitt 1: Gestenfindung

Zunächst werden die Gesten beschrieben, die von den Teilnehmern zur Durchführung der einzelnen Interaktionen vorgeschlagen wurden, anschließend werden die Einschätzungen zu den Eigenschaften „kompliziert“ und „ermüdend“ sowie die Ordnung der Interaktionen nach der Schwierigkeit wiedergegeben

Gestenvorschläge der Teilnehmer

Bei der Durchführung der einzelnen Gesten zu den sieben Interaktionen konnten folgende Beobachtungen dokumentiert werden:

- **Aufgabe 1.1:** Objekt verschieben
  Zwei Probanden markierten das zu verschiebende Objekt mit einer Greifgeste, ein Proband durch eine Zeigegeste mit der Handfläche. Ein Proband markierte das Objekt, indem er mit zwei Fingern darauf deutete.
  Alle Probanden beschrieben anschließend mit ausgestreckten Armen den Pfad, auf dem sich das markierte Objekt zum gewünschten Zielpunkt bewegen sollte.

- **Aufgabe 1.2:** Objekt löschen
  Drei Probanden markierten das Objekt mit einer Greifgeste, ein Proband durch eine Zeigegeste mit der Handfläche.
  Anschließend bewegten drei der vier Probanden das markierte Objekt „an eine bestimmte Stelle“ wie beispielsweise einen Papierkorb um das Objekt zu löschen. Ein Proband bewegte das Objekt aus dem sichtbaren Bereich der Powerwall.

- **Aufgabe 1.3:** Objekt skalieren
  Hier waren sich alle vier Probanden darüber einig, das Objekt an zwei gegenüberliegenden Ecken zu markieren und anschließend durch auseinanderbewegen der Arme das Objekt zu vergrößern oder durch zusammenbewegen der Arme das Objekt zu verkleinern.
  Ein Proband benutzte zum Markieren der Ecken die Zeigefinger beider Hände, die anderen führten Greifgesten mit beiden Händen aus.

- **Aufgabe 1.4:** Objekt duplizieren
  Ein Proband erstellte Kopien eines Objekts, indem er das Objekt mehrfach mit der Handfläche antippte. Die Anzahl der Tipp-Vorgänge bestimmte die Anzahl der Duplikate, die erstellt werden sollten. Die anderen drei Probanden markierten das zu duplizierende Objekt mit einer Hand, zeigten anschließend mit der anderen Hand oder einem Finger der anderen Hand ebenfalls auf das zu duplizierende Objekt und „zogen" eine Kopie aus dem Objekt heraus, indem sie die zweite Hand vom Objekt weg auf eine freie Fläche der Powerwall bewegten.

- **Aufgabe 1.5:** Objekte koppeln
  Die Mehrheit der Probanden zeichnete mit einer Hand oder einem Finger eine Verbindungslinie zwischen den zu koppelnden Objekten. Ein Proband markierte zwei zu koppelnde Objekte mit je einer Hand und führte die beiden Hände anschließend vor dem Körper zusammen.

- **Aufgabe 1.6:** Ansicht in einem Objekt wechseln
  Um die Ansicht innerhalb eines Objekts zu wechseln führten zwei der vier Probanden eine Wischgeste aus. Mit der Richtung der Wischbewegung sollte sich der Inhalt des Objekts verändern.
  Ein Proband führte mit einer Hand eine X-förmige Bewegung in Richtung des Objekts durch um den aktuellen Inhalt zu entfernen. Anschließend bewegte er den neuen Inhalt durch eine Verschiebegeste aus einem Pool von Möglichkeiten in das Objekt hinein.
  Ein anderer Proband markierte den Inhalt des Objekts mit einer Greifgeste und entfernte den Inhalt mit einer schnellen Bewegung in Richtung Boden aus dem Objekt. Anschließend bewegte der ebenfalls den neuen Inhalt aus einem Pool von Möglichkeiten mithilfe einer Verschiebegeste in das Objekt hinein.

- **Aufgabe 1.7:** Objektinhalt manipulieren
  Ein Proband markierte mit einer Hand das Objekt, dessen Inhalt manipuliert werden sollte. Anschließend manipulierte er den Inhalt des Objekts mit einem Finger der anderen Hand. Die restlichen Probanden manipulierten den Inhalt des Objekts ausschließlich mit dem Finger einer Hand. Zur Markierung des Bereichs, welcher innerhalb des Objekts manipuliert werden sollte, zeichnete ein Proband die Umrisse des Bereichs nach. Ein anderer Proband malte mit einem Finger diesen Bereich aus.

Einschätzung der Gesteninteraktionen

Die Probanden prognostizierten, dass die geforderten Interaktionen in Bezug auf die Kompliziertheit zwischen 1,5 (unkompliziert) und 14,0 (recht kompliziert) liegen würden (siehe Abbildung 96). Hierbei wurde die Interaktion „Löschen" als die voraussichtlich komplizierteste Interaktion eingeschätzt, die Interaktion „Verschieben" als die unkomplizierteste. Im Mittel lag die Einschätzung der Kompliziertheit aller sieben Interaktionen bei 6,9 (mäßig kompliziert).

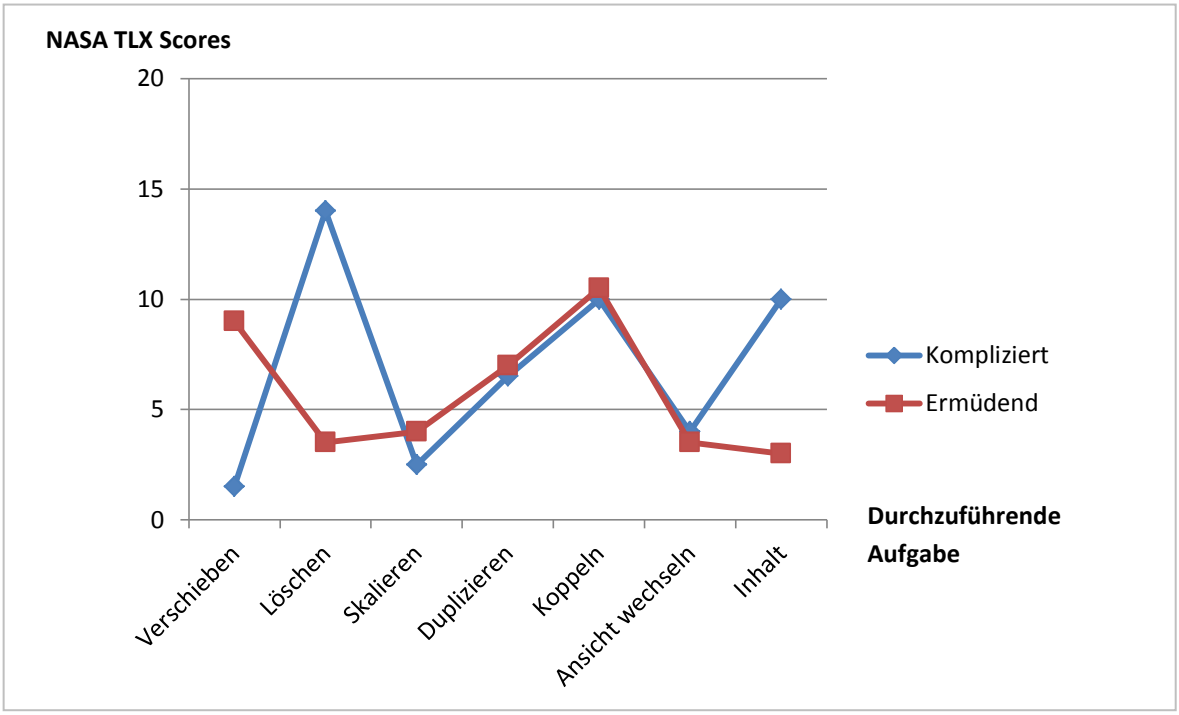


Abbildung 96: Gemittelte Einschätzung der Studienteilnehmer der Gesten aus den Aufgaben 1.1 bis 1.7. in den Eigenschaften „kompliziert“ und „ermüdend“ auf einer Punkteskala von 0 bis 20.

Die Einschätzung der Probanden bezüglich der Eigenschaft „ermüdend“ wies eine ähnliche Streuung auf, wie die bezüglich der Eigenschaft „kompliziert“. Die Schätzungen bewegten sich zwischen 3,0 (leicht ermüdend) und 10,5 (mäßig ermüdend). Im Mittel wurden alle Interaktionen auf „leicht ermüdend“ (5,8) geschätzt.

Anordnung der Gesten nach Schwierigkeit

In der folgenden Tabelle ist die Anordnung der Gesteninteraktionen durch die Probanden anhand der Schwierigkeit von leicht nach schwer dargestellt:

	Ranking	Interaktion
Leicht	1	Objekt löschen
↓	2	Objekt duplizieren
	3	Objekt verschieben
	4	Objekte koppeln
	5	Ansicht in einem Objekt wechseln
		Objekt skalieren
Schwer	7	Objektinhalt manipulieren

Tabelle 4: Anordnung der Interaktionen von „leicht“ nach „schwer“. Die Interaktion „Objekt löschen“ wurde von den Teilnehmern als leichteste, die Interaktion „Objektinhalt manipulieren“ als schwerste eingeschätzt.

8.3.2 Abschnitt 2: Beurteilung der Gesten im Prototyp

Im Folgenden werden die Beurteilungen der Teilnehmer zu den im Prototyp implementierten Gesteninteraktionen dargestellt. Anschließend werden die Bewertungen der Teilkonzepte und des Gesamtsystems beschrieben.

Beurteilung der einzelnen Gesten

Die gemittelten Ergebnisse der **Aufgaben 2.1 bis 2.7** sind in den Diagrammen in Abbildung 97, Abbildung 98 und Abbildung 99 dargestellt. Abbildung 97 visualisiert die Variablen „passend“, „intuitiv“ und „leicht zu lernen“, bei denen für positive Ergebnisse hohe Zustimmung erforderlich ist.

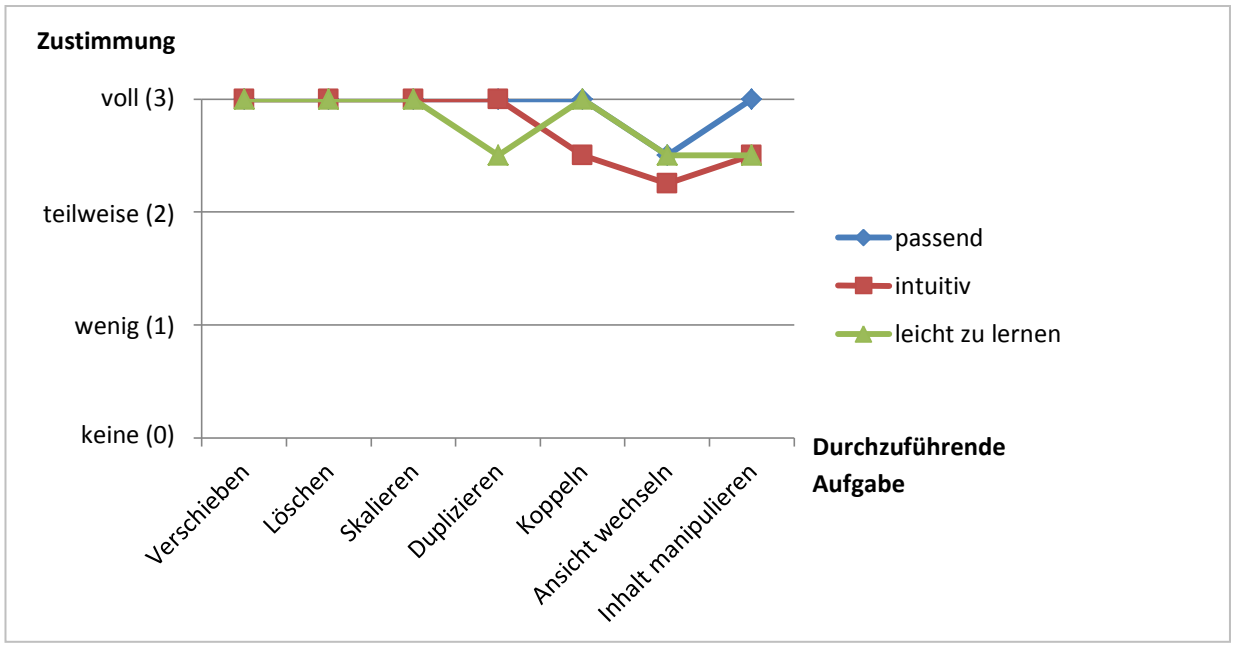


Abbildung 97: Bewertung der Aufgaben 2.1 bis 2.7 nach den Eigenschaften „passend“, „intuitiv“ und „leicht zu lernen“.

Im Mittel wurden die sieben Interaktionen als passend bewertet (2,9). Am unpassendsten wurde die Interaktion „Ansicht wechseln“ mit 2,5 eingestuft. Die restlichen Interaktionen wurden mit dem Maximalwert 3 (volle Zustimmung) bewertet.
Ebenso wurden alle Interaktion im Mittel als intuitiv bewertet (2,7). Am wenigsten intuitiv bewerteten die Probanden die Interaktion „Ansicht wechseln“ mit 2,3. Die beiden Interaktionen „Objekte koppeln“ und „Objektinhalt manipulieren“ wurden mit 2,5 und die restlichen mit dem Höchstwert 3,0 eingestuft.
Auch die Variable „leicht zu lernen“ erhielt ausschließlich positive Bewertungen. Im Mittel wurde sie mit 2,8 bewertet. Dabei wurden die Interaktionen „Objekt duplizieren“, „Ansicht wechseln“ und „Objektinhalt manipulieren“ mit 2,5 und die restlichen Interaktionen mit 3,0 (volle Zustimmung) eingestuft. In Abbildung 98 sind die Variablen „kompliziert“, „umständlich“ und „peinlich“ dargestellt, bei denen eine geringe Zustimmung zu positiven Ergebnissen führt.

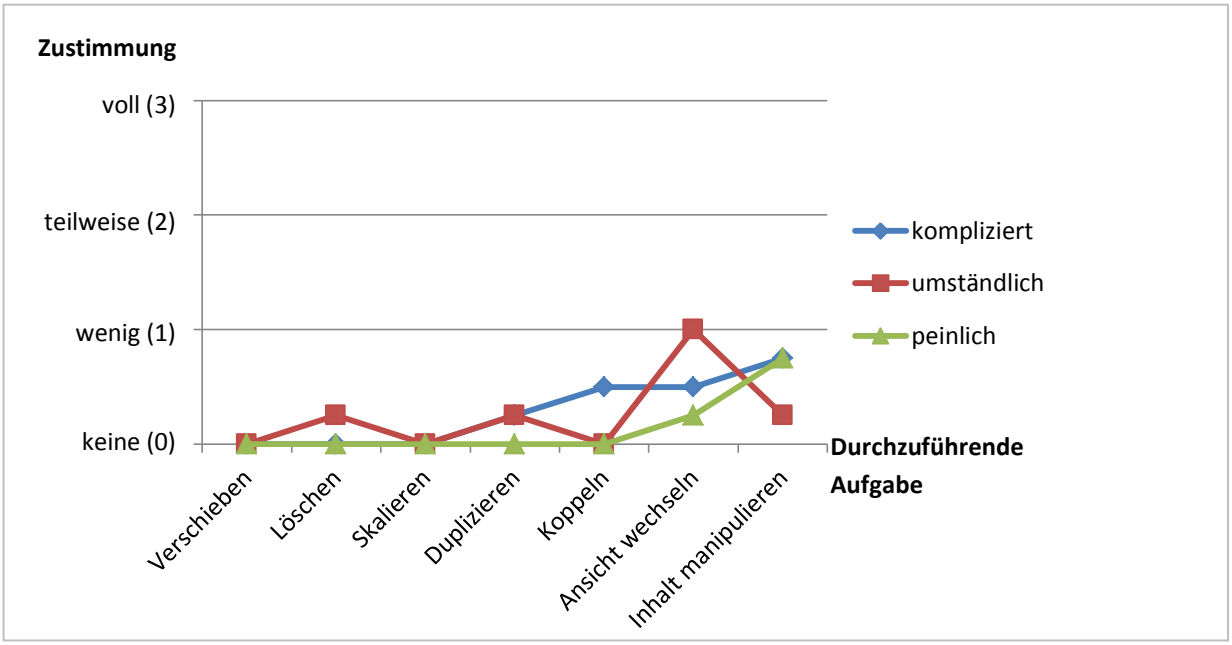


Abbildung 98: Bewertung der Aufgaben 2.1 bis 2.7 nach den Eigenschaften „kompliziert“, „umständlich“ und „peinlich.

Von den Probanden wurde keine Interaktion als kompliziert bewertet. Der Mittelwert liegt mit 0,3 zwischen „keine Zustimmung“ und „wenig Zustimmung“. Am kompliziertesten wurde die Interaktion „Objektinhalt manipulieren“ mit 0,8 empfunden. Als gänzlich unkompliziert wurden die Interaktionen „Objekt verschieben“, „Objekt löschen“ und „Objekt skalieren“ eingestuft. Die Interaktion „Objekt duplizieren“ wurde mit 0,3 und die Interaktionen „Objekte koppeln“ und „Ansicht wechseln“ mit 0,5 bewertet.
Ebenfalls wurde keine Interaktion als umständlich eingestuft. Der Mittelwert beträgt 0,3. Am umständlichsten wurde die Interaktion „Ansicht wechseln“ mit 1,0 („wenig Zustimmung“) bewertet. Die Interaktionen „Objekt löschen“, „Objekt duplizieren“ und „Objektinhalt manipulieren“ wurden mit 0,3 und die restlichen mit 0,0 („keine Zustimmung“) bewertet.
In der Eigenschaft „peinlich“ erhielten die Interaktionen „Ansicht wechseln“ und „Objektinhalt manipulieren“ minimale Zustimmung und wurden mit 0,3 bzw. 0,8 bewertet. Im Mittel wurden die Interaktionen mit einer Zustimmung von 0,1 als peinlich eingestuft.

Zusätzlich sollten die Probanden die beispielhaft implementierten Interaktionen nach ausgewählten Kriterien des NASA TLX (Hart, 2006) bewerten. Hierzu wurden die Variablen „Mental Demand“ (geistiger Anspruch), „Physical Demand“ (körperlicher Anspruch), „Performance“ (Erfolg) und „Frustration“ herangezogen (Abbildung 99). Bei der Auswertung der Ergebnisse sind kleine Werte auf der Skala von 1 bis 20 erstrebenswert.

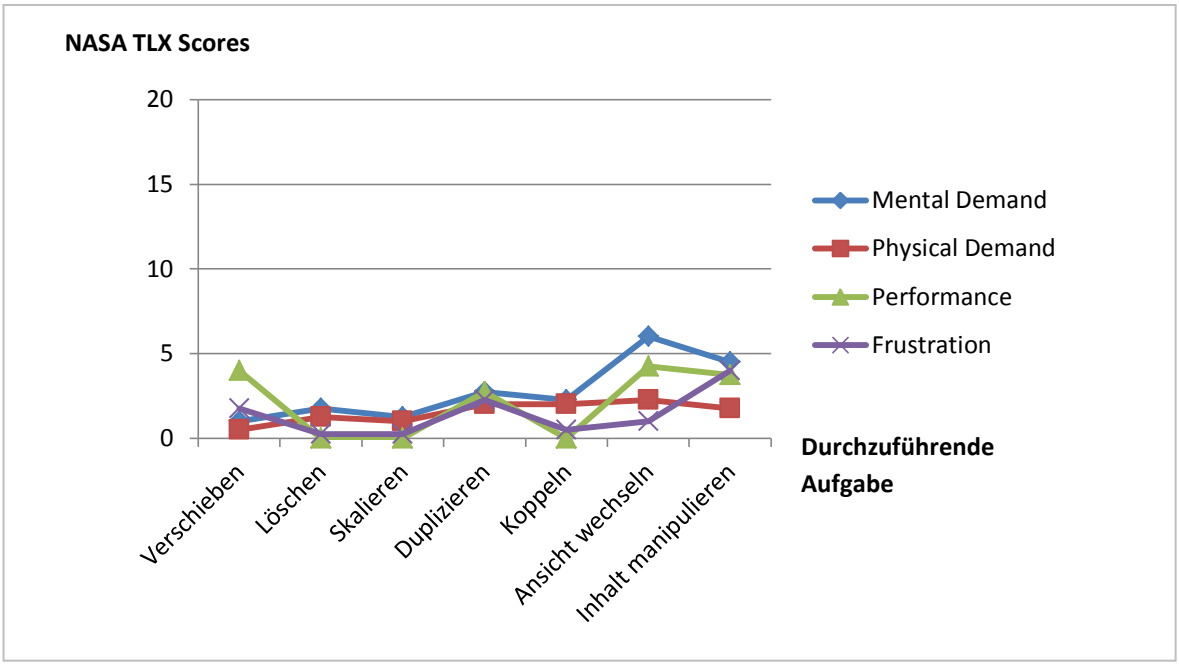


Abbildung 99: Bewertung der Aufgaben 2.1 bis 2.7 nach Kriterien des NASA TLX.

Insgesamt wurde der geistige Anspruch („Mental Demand“) der Interaktionen im Mittel mit 3,4 bewertet. Den größten geistigen Anspruch hatte die Interaktion „Ansicht wechseln“ mit einem Wert von 6,3. Den höchsten körperlichen Anspruch hingegen hatten die Interaktionen „Objekte koppeln“ und „Ansicht wechseln“ mit 2,8. Im Mittel wurden die Interaktionen mit 2,2 in der Eigenschaft „körperlich anspruchsvoll“ eingestuft.
Den größten Erfolg bei der Ausführung der Interaktionen hatten die Probanden bei den Interaktionen „Objekte koppeln“ und „Objekt skalieren“, die sie jeweils mit einem Wert von 1,0 bewerteten. Den geringsten Erfolg lieferten die Interaktionen „Objekt verschieben“ und „Ansicht wechseln“ (4,5). Im Mittel bewerteten die Probanden den Erfolg bei den Interaktionen mit 2,8.
In der Eigenschaft „Frustration“ wurden die Interaktionen im Mittel mit 1,9 bewertet. Den größten Frust erfuhren die Probanden bei der Interaktion „Objektinhalt manipulieren“ (4,5) und den geringsten Frust bei den Interaktionen „Objekt löschen“ und „Objekt skalieren“ (1,0).

Beurteilung der Teilkonzepte

In **Aufgabe 2.8** wurden die Probanden gebeten das Konzept der Interaktionsbereiche für Nah- und Ferninteraktionen nach denselben Kriterien wie die einzelnen Interaktionen zu bewerten. In **Aufgabe 2.9** wurden dieselben Variablen für das Konzept des Schatteninterfaces und in **Aufgabe 2.10** für das Konzept der Hilfestellungen erhoben (siehe Abbildung 100).

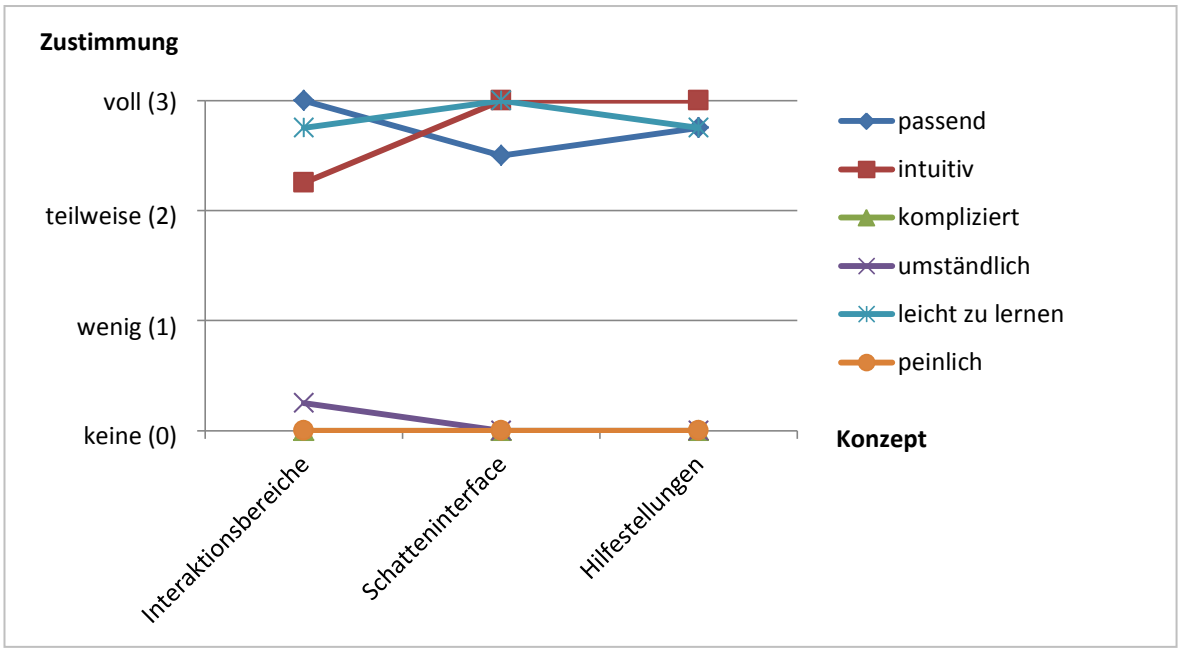


Abbildung 100: Beurteilung der Teilkonzepte "Interaktionsbereiche", "Schatteninterface" und "Hilfestellungen".

Im Mittel wurden die drei Konzepte mit einer Zustimmung von 2,7 als „passend“, mit 2,8 als „intuitiv“ und mit 2,8 als „leicht zu lernen“ bewertet. Die Eigenschaften „kompliziert“, „umständlich“ und „peinlich“ wurden im Mittel mit einer Zustimmung von 0,0 und 0,3 sowie 0,0 bewertet.

Beurteilung des Gesamtsystems

Die **Aufgabe 2.11** verlangte von den Probanden eine Bewertung des Gesamtsystems nach den Kriterien des System Usability Scale (Brooke, 1996). Der SUS bewertet die Usability eines Systems auf einer Skala von 1 bis 100.

Die Probanden der Pilotstudie bewerteten das im Prototyp implementierte System zur Gesteninteraktion mit Powerwall-Visualisierungen mit den sieben Interaktionen und den drei Teilkonzepten „Interaktionsbereiche“, „Schatteninterface“ und „Hilfestellungen“ im Mittel mit 89 von 100 Punkten. Maximal erhielt das System 92,5 und minimal 85 Punkte.

8.4 Diskussion

In diesem Abschnitt werden die Ergebnisse der Pilotstudie ausgewertet. Dabei wird berücksichtigt, dass die Pilotstudie vor allem qualitative Ergebnisse liefert, da sie nur mit vier Probanden durchgeführt wurde. Weiterführende Ergebnisse werden aus der Benutzerstudie erwartet, die nach Abschluss dieser Diplomarbeit durchgeführt werden wird.

8.4.1 Abschnitt 1: Gestenfindung

Die Auswertung der Gesten, welche die Probanden für die sieben Interaktionen vorgeschlagen haben liefern richtungsweisende Ergebnisse. Insgesamt waren die vier Pilot-Probanden ähnlicher Meinung bzgl. der Findung intuitiver Gesten: Im Schnitt haben in jeder Aufgabe drei Probanden dieselbe oder zumindest eine ähnliche Geste ausgeführt um die jeweilige Interaktion auszuführen. Bei fünf der sieben Aufgaben stimmte sogar die vorgeschlagene Geste mit der im Prototyp implementierten Geste überein.

Dieses Ergebnis legt nahe, dass **Hypothese 1** in der folgenden Benutzerstudie weiter bekräftigt wird. Es scheint intuitive Gesten zur Interaktion mit Powerwall-Visualisierung zu geben.

Betrachtet man die Einschätzung der Probanden aus den **Aufgaben 1.1 bis 1.7**, in denen die Probanden dieselben sieben Interaktionen bezüglich der Eigenschaften „kompliziert“ und „ermüdend“ bewerten sollten, ergeben sich Differenzen: Beispielsweise wurde die Interaktion „Objekt löschen“ von den Probanden als die komplizierteste Interaktion bewertet. Man sollte erwarten, dass die komplizierteste Interaktion auch zu den schwierigsten zählt.
Nimmt man an, dass sowohl der geistige Anspruch an den Probanden bei der Durchführung einer Interaktion (erhoben durch die Variable „kompliziert“) als auch der körperliche Anspruch (erhoben durch die Variable „ermüdend“) zu gleichen Teilen in die Schwierigkeit einer Aufgabe einfließen, lässt sich die Ordnung der Interaktionen folgendermaßen darstellen:

	Ranking	Interaktion
Leicht	1	Objekt skalieren
	2	Ansicht in einem Objekt wechseln
	3	Verschieben
	4	Objektinhalt manipulieren
	5	Objekt duplizieren
	6	Objekt löschen
Schwer	7	Objekte koppeln

Tabelle 5: Legt man die Antworten der Teilnehmer der Pilotstudie zugrunde, müsste so die Anordnung der Interaktionen nach Schwierigkeit aussehen. Dieses Ergebnis weicht von den Angaben der Teilnehmer in Aufgabe 1.8 (vgl. Tabelle 4) ab.

Möglicherweise unterscheiden sich die Einschätzungen der Probanden während der Durchführung der Interaktionen und nach Durchführung aller Interaktionen, weil sie bei der zweiten Erhebung bereits das volle Interaktionsspektrum kennen und Vergleiche ziehen können.

Zusammenfassend lassen sich in den durch Probanden der Pilotstudie vorgeschlagenen Gesten zur Interaktion mit Powerwall-Visualisierungen folgende Tendenzen ableiten:

- Interaktionen, die das gesamte Objekt und nicht dessen Inhalt manipulieren, werden mit der ganzen Hand ausgeführt. Entweder durch eine Greifgeste oder durch Markierung mit der Handfläche.
- Interaktionen, die den Inhalt eines Objekts manipulieren, werden überwiegend mit den Fingern ausgeführt.
- Es gibt viele Parallelen zu den Gesten, die im Prototyp implementiert wurden.

8.4.2 Abschnitt 2: : Beurteilung der Gesten im Prototyp

Insgesamt wurden alle im Prototyp implementierten Gesten positiv bewertet. Dies resultiert nicht zuletzt aus den zuvor genannten Parallelen zwischen den vorgeschlagenen Gesten durch die Probanden und den implementierten. Die Auswahl intuitiver und einfacher Gesten zur Interaktion mit Powerwall-Visualisierungen im Prototyp ist bestätigt worden. Es bleibt abzuwarten, ob die implementierten Gesten auch von einer größeren Anzahl von Probanden bestätigt werden. Darüber hinaus wurde keine Geste als besonders ermüdend oder körperlich anspruchsvoll bezeichnet. Als ermüdendste Interaktionen wurden die Interaktionen „Objekte koppeln“ und „Ansicht wechseln“ mit jeweils 2,8 bewertet. Auch eine körperliche Ermüdung über die Dauer der einzelnen Studien konnte nicht festgestellt werden. Dies unterstützt weiterhin die erfolgreiche Auswahl der Gesten, die prototypisch implementiert wurden.

Dennoch sind zwei Punkte negativ hervorzuheben:

- Die Frustration ist in **Aufgabe 2.7** („Objektinhalt ändern“) mit 4,5 am größten bewertet worden. Eine Erklärung hierfür ist, dass die Handerkennung im Prototyp schlecht funktioniert und nur von geübten Anwendern problemlos bedient werden kann.
- In **Aufgabe 2.6** („Ansicht wechseln“) wurde die Interaktion in der Eigenschaft „geistiger Anspruch“ („Mental Demand“) mit 6,3 eingestuft. Während der Studiendurchführung war auch in dieser Aufgabe der größte Erklärungsbedarf. Das Problem fand sich im Feedback bei der Interaktion. Sobald die Interaktion erkannt wurde, lieferte der Prototyp noch kein ausreichendes Feedback um den Probanden zu vermitteln wie die Geste weiterzuführen ist.

Für diese und andere Punkte werden in Kapitel 8.5.2 Vorschläge zur Optimierung in Hinsicht auf die Hauptstudie gemacht.

Vergleich der Gestenbeurteilung aus beiden Studienabschnitten

Nachdem die Probanden eine Geste in **Aufgabe 1** zur Interaktion mit Powerwall-Visualisierungen vorgeschlagen hatten, wurden sie befragt wie kompliziert sie eine solche Interaktion einschätzen würden. **In Aufgabe 2** sollten die implementierten Gesteninteraktionen unter anderem in der Eigenschaft „kompliziert“ bewertet werden. Abbildung 101 vergleicht die Angaben aus der ersten und zweiten Aufgabe:

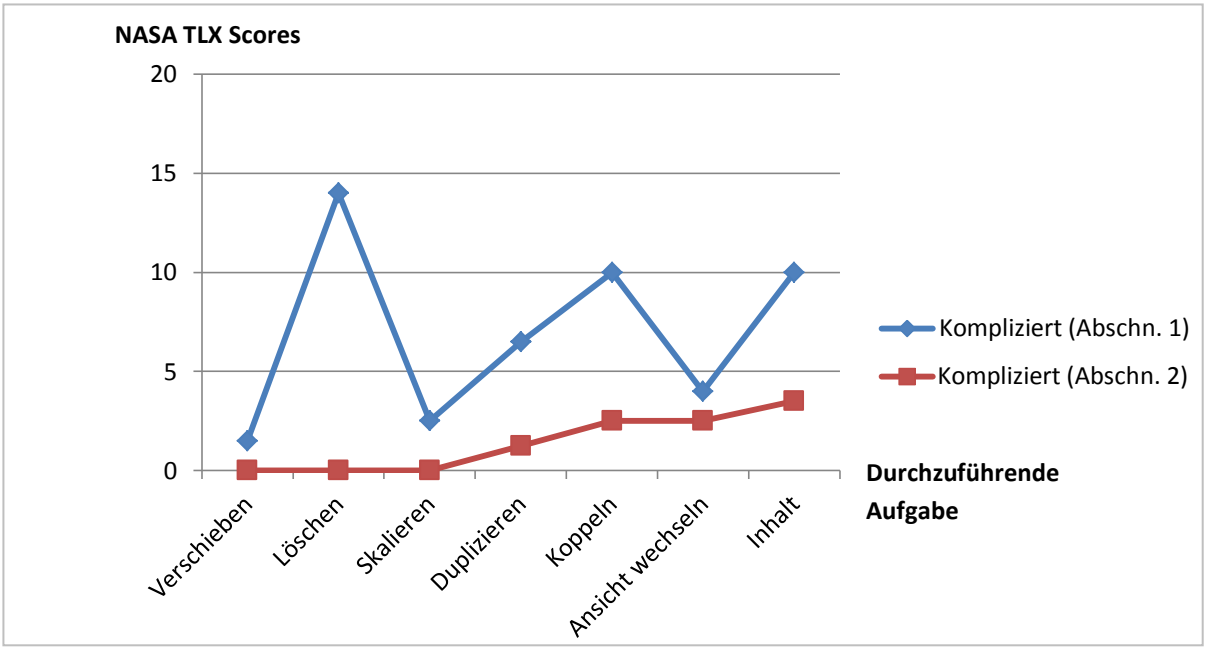


Abbildung 101: Vergleich der Beurteilungen der Gesteninteraktionen in der Eigenschaft "kompliziert". Beurteilung im ersten Studienabschnitt (blau) und im zweiten Studienabschnitt (rot).

Hier wird deutlich, dass die Probanden die Interaktionen schwerer eingeschätzt haben, als sie die Interaktionen bei der Durchführung in der zweiten Aufgabe empfanden. Im ersten Teil wurden die Interaktionen im Mittel mit 6,9 und im zweiten nur noch mit 1,4 als „kompliziert“ eingestuft.

Derselbe Effekt ist auch beim Vergleich der Ergebnisse bei der Bewertung nach der Eigenschaft „ermüdend“ bzw. „körperlicher Anspruch“ zu finden (siehe Abbildung 102):

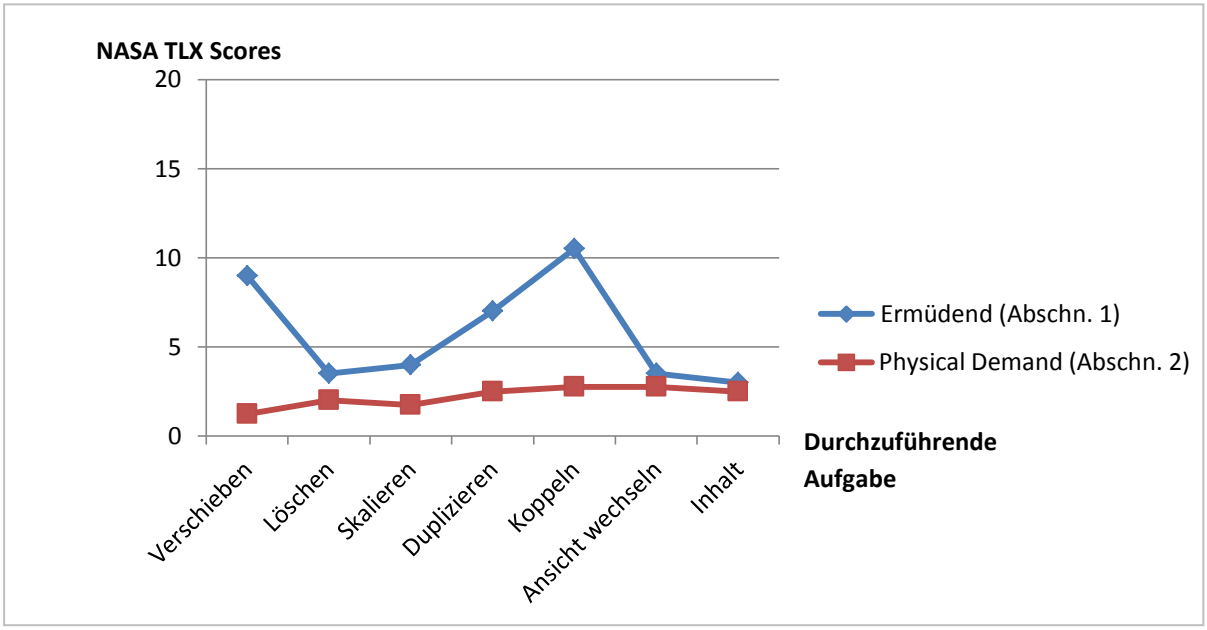


Abbildung 102: Vergleich der Beurteilung der Gesteninteraktionen in der Eigenschaft „ermüdend" bzw. „physical Demand". Beurteilung im ersten Studienabschnitt (blau) und im zweiten Studienabschnitt (rot).

Im ersten Aufgabenteil bewerteten die Probanden die Interaktionen im Mittel mit 5,8 und im zweiten Teil nur noch mit 2,2 als „körperlich anstrengend".

Die Probanden der Pilotstudie hatten kaum tiefgehendes Vorwissen auf dem Gebiet der Gesteninteraktion. Möglicherweise schätzen sie daher Gesten zur Interaktion mit Powerwall-Visualisierungen grundsätzlich komplizierter und ermüdender ein. Durch die große Übereinstimmung zwischen den von den Teilnehmern vorgeschlagenen Gesten und den Gesten, die im Prototyp implementiert wurden, fühlten sich die Probanden in ihrer Gestenauswahl bestätigt. Sie konnten erfahren, wie die von ihnen vorgeschlagenen Gesten in einem Interaktionssystem umgesetzt werden können. Dies könnte zu einer besseren Bewertung im zweiten Abschnitt geführt haben.

Ein weiterer Faktor könnte das Feedback spielen: im ersten Teil der Studie erhielten die Probanden keinerlei Feedback zu ihren Interaktionen. Bei der Interaktion mit den beispielhaft implementierten Interaktionen konnte der Proband seinen Einfluss auf die Visualisierungen sofort erkennen, was zu einer besseren Bewertung führte. Einen derartigen Effekt konnte auch in der Studie (Wobbrock, Morris, & Wilson, 2009) zur Gestenfindung für Tabletop-Computer durch Benutzer nachgewiesen werden.

Die qualitativen Ergebnisse des zweiten Studienabschnitts und die Ähnlichkeit zwischen den Interaktionsgesten, die von den Teilnehmern vorgeschlagen wurden, und den im Prototyp implementierten legen nahe, dass auch **Hypothese 2** in einer groß angelegten Benutzerstudie bekräftigt wird: Die im Prototyp implementierten Gesten eigenen sich zur Interaktion mit Powerwall-Visualisierungen.

8.5 Zusammenfassung

Dieser Abschnitt fasst die Pilotstudie, die im Rahmen dieser Diplomarbeit durchgeführt wurde, nochmals zusammen. Anschließend werden Optimierungen für das Studiendesign der folgenden Benutzerstudie vorgeschlagen.

8.5.1 Zusammenfassung der Pilotstudie

Im Rahmen dieser Diplomarbeit zur Gesteninteraktion mit Powerwall-Visualisierungen wurde eine kombinierte Pilotstudie mit vier Teilnehmern durchgeführt. Für diese Pilotstudie wurden nur qualitative Ergebnisse erwartet. Es sollten folgende Hypothesen mit dieser Studie untersucht werden:

- **Hypothese 1**: Es gibt intuitive Gesten für bestimmte Freihand-Interaktionen mit Powerwall-Visualisierungen, die von den meisten Anwendern vorgeschlagen werden.
- **Hypothese 2**: Die im Prototyp implementierten Gesten zur Interaktion mit Powerwall-Visualisierungen sind intuitiv und eignen sich für die jeweilige Aufgabe.

Die Studie war in zwei Abschnitte aufgegliedert. Im ersten Abschnitt sollten die Teilnehmer Gesten zu den Interaktionen „Objekt verschieben“, „Objekt löschen“, „Objekt skalieren“, „Objekt duplizieren“, „Objekte koppeln“, „Ansicht in einem Objekt wechseln“ und „Objektinhalt verändern“ vorschlagen. Dabei wurde die Methode des „lauten Denkens“ angewendet und die Teilnehmer bei der Durchführung der Geste gefilmt. Anschließend sollten die Teilnehmer die Schwierigkeit der durchgeführten Gesten einschätzen.
Im zweiten Abschnitt sollten die Teilnehmer die Gesten bewerten, die in der Prototypsoftware zu dieser Diplomarbeit implementiert wurden. Die Teilnehmer sollten diese Gesten nach den Kriterien „passend“, „intuitiv“, „kompliziert“, „umständlich“, „leicht zu lernen“ und „peinlich“ bewerten. Ebenfalls wurden ausgewählte Eigenschaften des NASA TLX (Hart, 2006) erhoben („Mental Demand“, „Physical Demand“, „Performance“ und „Frustration“). Abschließend sollten die Teilnehmer das Gesamtsystem des Prototyps mit dem System Usability Scale (Brooke, 1996) bewerten.

Die Pilotstudie wurde mit vier Teilnehmern durchgeführt. Die Ergebnisse lassen einen Trend erkennen, der die beiden aufgestellten Hypothesen bekräftigt: In sechs von sieben Aufgaben schlugen jeweils drei der vier Probanden dieselbe oder eine ähnliche Geste zur Interaktion vor. In fünf Fällen stimmte die von den Teilnehmern vorgeschlagene Geste mit der tatsächlich im Prototyp implementierten Geste überein. Darüber hinaus bewerteten die Teilnehmer die im Prototyp implementierten Gesten weder als „kompliziert“ noch „umständlich“ oder gar „peinlich“. Die Teilnehmer stimmten überein, dass die Gesten des Prototyps durchweg „passen“, „intuitiv“ und „leicht zu erlernen“ sind. Im Vergleich zu den Gesten, die von den Teilnehmern selbst vorgeschlagen wurden, wurden die Gesten des Prototyps und deren Implementierung besser bewertet. Dies liegt möglicherweise am fehlenden Feedback im ersten Abschnitt der Pilotstudie. Dieser Effekt wurde ebenfalls bei der Studie zur Gestenfindung für Multitouch-Tabletop-Computer festgesellt (Wobbrock, Morris, & Wilson, 2009). Das Gesamtsystem und das Interaktionskonzept wurden mit durchschnittlich 89 von 100 möglichen Punkten im System Usability Scale bewertet.

Diese qualitativen Ergebnisse deuten auf eine weitere Bekräftigung der Hypothesen in der Benutzerstudie, die nach Abschluss dieser Diplomarbeit durchgeführt wird. In Kapitel 8.5.2 werden Maßnahmen zur Optimierung des Studiendesigns vorgeschlagen.

8.5.2 Optimierung des Studiendesigns für die Benutzerstudie

Durch die Ergebnissen der vier Pilot-Probanden und den Erfahrungen bei der Durchführung der Pilotstudie wurden einige Punkte im Studiendesign und der Studiendurchführung gefunden, die vor der Durchführung einer ausführlichen Studie verbessert werden sollten. Hierzu zählen die Vereinheitlichung der Variablennamen, die Vereinheitlichung der Skalen in den Fragebögen und ein besseres Feedback bei der Interaktion „Ansicht wechseln“. Ebenfalls kam die Idee eines Fünf-Minuten-Interaktionstests auf. Auf diese Punkte wird im Folgenden näher eingegangen.

Vereinheitlichung der Variablennamen

Im ersten Teil der Studie sollen die Probanden zu sieben Interaktionen zunächst eine Geste beschreiben, mit der sie die jeweilige Interaktion durchführen wollten. Anschließend sollten die Probanden die von ihnen vorgeschlagene Geste nach den Eigenschaften „kompliziert“ und „ermüdend“ auf einer Skala von 1 bis 20 bewerten.
Nach Abschluss des ersten Studienteils bewerten die Probanden im zweiten Teil beispielhafte Implementierungen derselben Interaktionen mithilfe des Prototyps. Hier wird unter anderem auch die Eigenschaft „Physical Demand“ („körperliche Anstrengung“) erhoben.

Die körperliche Anstrengung der durchgeführten Gesteninteraktionen in beiden Teilen der Studie soll miteinander verglichen werden. Damit die Probanden in beiden Studienteilen dieselben Eigenschaften bewerten, müssen die Variablennamen vereinheitlicht werden um einen wirklichen Vergleich zu ermöglichen.

Vereinheitlichung der Skalen

Auf den Fragebögen zu den Gesteninteraktionen in der ersten Aufgabe werden die Bewertungen der Eigenschaften „kompliziert“ und „ermüdend“ auf einer Skala von 1 bis 20 erhoben. Dies entspricht der Skala, die auch von den ausgewählten Fragen des NASA TLX zur Bewertung der Interaktionen im zweiten Studienteil verwendet wird.
Allerdings werden im zweiten Teil die Eigenschaften „passend“, „intuitiv“, „kompliziert“, „umständlich“, „leicht zu erlernen“ und „peinlich“ auf einer vierstufigen Likert-Skala erhoben.

Wie bereits beschrieben werden die Bewertungen bezüglich der Eigenschaften „kompliziert“ und „ermüdend“ aus den beiden Aufgabenteilen in der Studienauswertung miteinander verglichen. Ein Vergleich der Eigenschaft „ermüdend“ aus Teil 1 mit der Eigenschaft „Physical Demand“ aus Teil 2 ist problemlos möglich. Beim Vergleich der Eigenschaft „kompliziert“ müssen die Bewertungen auf der vierstufigen Likert-Skala zunächst auf die Skala des NASA TLX umgerechnet werden. Hierbei geht der Vorteil einer feineren Abstufung der Skala von 1 bis 20 verloren. In der Hauptstudie sollten daher alle Variablen auf der Skala des NASA TLX erhoben werden.

Verbesserung des Feedbacks

Aufgabe 2.6 bereitete den vier Pilot-Probanden deutliche Probleme. In dieser Aufgabe sollten die Probanden die Ansicht innerhalb eines Objekts auf der Powerwall ändern. Die Schwierigkeiten bei der Aufgabendurchführung spiegeln sich auch in der Bewertung der Eigenschaft „Mental Demand“ („geistiger Anspruch“) wider: Mit einem Wert von 6,25 wurde die Interaktion „Ansicht wechseln“ als Interaktion mit dem höchsten geistigen Anspruch eingestuft. Als Hauptursache stellte sich das späte Feedback heraus:
Um die Interaktion durchzuführen mussten die Probanden zunächst das Objekt mit einer Hand

markieren. Nachdem das Objekt markiert wurde, musste die andere Hand ebenfalls über das Objekt bewegt werden. Bis zu diesem Zeitpunkt erhielten die Probanden noch keinerlei Feedback. Nun mussten die Probanden die zweite Hand wieder zu einer beliebigen Seite des Objekts herausbewegen, um ein Feedback über die anderen möglichen Ansichten zu erhalten. Dieser Schritt gestaltete sich für alle Probanden schwierig.

Zur Verbesserung des Feedbacks soll die Interaktion „Ansicht wechseln“ im Prototyp für die Hauptstudie folgendermaßen abgeändert werden:
Sobald der Proband die zweite Hand über das bereits markierte Objekt bewegt, werden sofort einige der möglichen Folgeansichten als Vorschau angezeigt. Ansichten, die durch eine Wischgeste nach links aus dem Objekt heraus ausgewählt werden können, werden rechts vom Objekt dargestellt, Ansichten die durch eine Wischgeste nach unten ausgewählt werden können, werden oberhalb des Objekts dargestellt, usw. (siehe Abbildung 103)

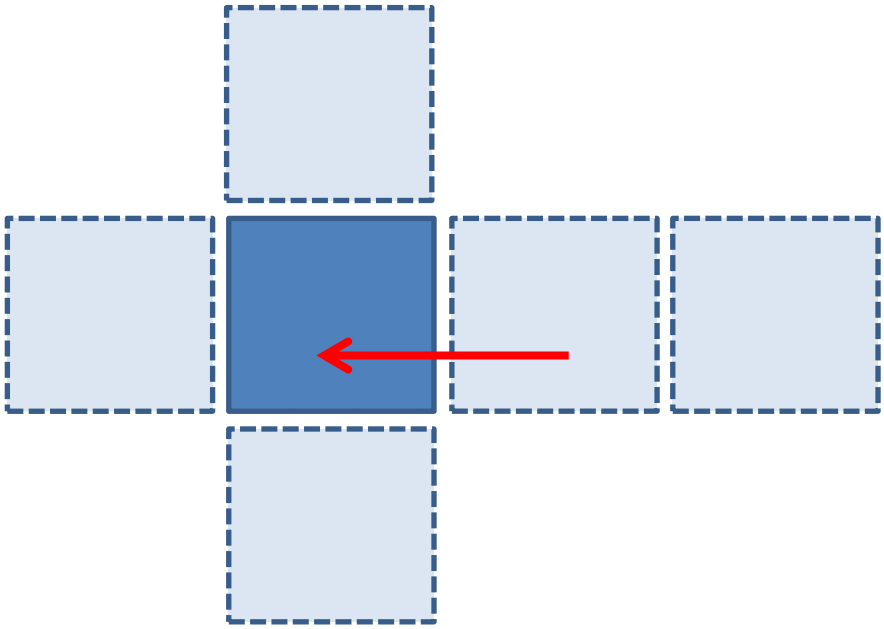


Abbildung 103: Optimiertes Feedback zur Interaktion "Ansicht wechseln". Sobald die zweite Hand über das Objekt bewegt wird, werden oben, unten, links und rechts die Ansichten eingeblendet, zu denen gewechselt werden kann. Mit einer Wischgeste nach links kann die erste Ansicht auf der rechten Seite in das Objekt gezogen werden.

Zusätzlich soll es möglich sein auch mit der ersten Hand die Wischgeste durchzuführen. Seither kann die erste Hand nach dem Markieren des Objekts nicht weiter verwendet werden.

Fünf-Minuten-Interaktionstest

Obwohl alle Interaktionen mit ausgestreckten Armen und häufig auch mit zwei Händen durchgeführt werden mussten, wurde keine Interaktion mit einem größeren Wert als 2,7 als „ermüdend“ bezeichnet. Um diese Beobachtung zu bestätigen, wird vorgeschlagen, eine Art Langzeittest durchzuführen. Die Probanden sollen über eine Zeitspanne von beispielsweise fünf Minuten unterbrechungsfrei eine bestimmte Interaktion durchführen und anschließend die körperliche Belastung bewerten.
Allerdings ist zu beachten, dass dies einen enormen Zeitaufwand darstellt und den Rahmen einer einzelnen Sitzung überschreitet. Möglicherweise könnte ein Fünf-Minuten-Interaktionstest für eine Interaktion pro Proband im zeitlichen Rahmen liegen.

9 Zusammenfassung und Ausblick

In diesem letzten Kapitel werden zunächst alle Ergebnisse dieser Diplomarbeit zusammengefasst und die wichtigsten inhaltlichen Aspekte hervorgehoben. Anschließend werden einige Punkte diskutiert, die im Verlauf der Diplomarbeit besonders positiv auffielen oder bei denen unerwartete Schwierigkeiten auftraten. Abschließend wird ein Ausblick zur Weiterentwicklung des Interaktionskonzepts und der Implementierung der multimodalen Interaktionsumgebung gegeben.

9.1 Zusammenfassung und Bewertung

Dieser Abschnitt fasst die Ergebnisse dieser Diplomarbeit zusammen. Zunächst wurden zwei Anwendungsszenarien entwickelt, aus denen ein Konzept zur Interaktion mit Powerwall-Visualisierungen erstellt wurde. Das Interaktionskonzept wurde in einem Prototyp anhand eines der beiden Szenarien beispielhaft implementiert und in einer Pilotstudie von Anwendern bewertet. Zusätzlich wurde ein Kommunikationsframework entwickelt, um die Interaktion mit Powerwall-Visualisierungen in eine multimodale Interaktionsumgebung integrieren zu können.

Im Anschluss an die inhaltliche Zusammenfassung werden einige Aspekte der Entwicklung des Konzepts und des Prototyps aufgegriffen, die besonders erfolgreich verliefen oder sich als problematisch herausstellten.

9.1.1 Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde ein Interaktionskonzept entwickelt, das eine Interaktion mit Visualisierungen auf einer Powerwall durch Freihandgesten ermöglicht. Es wurde darauf geachtet, dass dieses Interaktionskonzept vielseitig einsetzbar ist, indem es nicht auf einen konkreten Anwendungsfall zugeschnitten wurde. Um dies zu erreichen, wurden zuvor zwei Prototypszenarien aus verschiedenen Gebieten der Visualisierung entwickelt:

1. Das erste Szenario zeigt die Anwendung des Interaktionskonzepts im Bereich der Informationsvisualisierung. Bei der Auswertung einer Eyetracking-Studie wird die große Anzeigefläche einer Powerwall genutzt, um die große Menge an Probandendaten übersichtlich zu visualisieren. Mithilfe des Interaktionskonzepts können die Analyse und Auswertung der Eyetracking-Daten direkt mit der Powerwall ohne die Verwendung von Maus oder Tastatur erfolgen. Somit kann auf Unterbrechungen des Analyseablaufs durch die Interaktion mit dem Steuercomputer der Powerwall verzichtet werden.

2. Im zweiten Szenario wird das Interaktionskonzept auf wissenschaftliche Visualisierungen angewendet. Mit der Visualisierungssoftware MegaMol werden Proteinmoleküle in dreidimensionalen Darstellungen auf der Powerwall visualisiert. Mithilfe des Interaktionskonzepts kann durch die dreidimensionale Darstellung navigiert und die Darstellung der Proteine angepasst werden.

Diese beiden Szenarien stellen unterschiedliche Anforderungen an eine Interaktionsumgebung. Basierend auf diesen beiden Szenarien wurde ein allgemeines Konzept zur Interaktion mit Powerwall-Visualisierungen erstellt.

Die Erkennung der Gesten zur Interaktion mit Powerwall-Visualisierungen erfolgt mit dem Kinect-Sensor von Microsoft. Durch die Verwendung einer Stereokamera kann auf die Verwendung von zusätzlichem Equipment wie beispielsweise Marker zur Personen- und Gestenerkennung verzichtet werden.

Zur Unterstützung des Benutzers bei der Gesteninteraktion mit der Powerwall wurde ein Schatteninterface erstellt. Dieses Interface zeigt dem Benutzer seine Position vor der Powerwall und den Interaktionsradius mit den Visualisierungen auf der Powerwall. Zusätzlich verfügt das Schatteninterface über Menü-Items, die weitere Funktionen zur Interaktion mit den Powerwall-Visualisierungen ermöglichen. Des Weiteren wurden im Sinne eines Natural User Interface folgende Hilfestellungen erstellt, die den Benutzer ebenfalls bei der Interaktion unterstützen:

- **Markierungslinie zum Interaktionsbeginn**
  Eine horizontale Linie auf Hüfthöhe des Benutzers zeigt an, ob das System auf eine Gesteneingabe durch den Benutzer wartet. Bewegt der Benutzer beide Hände über diese Linie, wird die Linie grün und der Benutzer kann Interaktionen mit den Visualisierungen ausführen.

- **Positionsmarkierung der Hände**
  Ein grüner und ein rosa Punkt zeigen dem Benutzer die erkannte Position der rechten bzw. linken Hand auf der Powerwall.

- **Indikatoren zum Auslösen und Beenden von Interaktionen**
  Rote Kreise um die Positionsmarkierung der Hände zeigen an, ob vom System eine Gesteninteraktion mit Powerwall-Visualisierungen erkannt wurde und wann eine Interaktion beendet wird.

Für die Interaktion mit Powerwall-Visualisierungen wurden beispielhaft einige Gesten implementiert. Das Gestenset umfasst Gesten zur Positionierung und Skalierung von Visualisierungen, sowie zur Duplikation und Kopplung. Ebenfalls können verschiedene Visualisierungen für dieselben Daten ausgewählt und die innerhalb der Visualisierung angezeigten Daten manipuliert werden.

Der Interaktionsbereich vor der Powerwall ist in zwei Teilbereiche unterteilt: einen Nah-Interaktionsbereich mit geringer Entfernung zur Powerwall und einen Fern-Interaktionsbereich. Mit Gesten, die innerhalb des weiter entfernten Interaktionsbereichs ausgeführt werden, können Übersichtsinteraktionen durchgeführt werden. Dies sind beispielsweise die Positionierung und Skalierung der Visualisierungen. Gesten, die innerhalb des Nah-Interaktionsbereichs ausgeführt werden, können die innerhalb einer Visualisierung angezeigten Daten manipulieren.

Das Interaktionskonzept zur Interaktion mit Powerwall-Visualisierungen wurde in Form eines Prototyps zur Interaktion mit Eyetracking-Visualisierungen implementiert. Der Prototyp unterstützt Interaktionen, die im Szenario zur Analyse von Eyetracking-Daten aufgeführt werden.

Ebenfalls wurde im Rahmen dieser Diplomarbeit eine Benutzerstudie entwickelt, mit der das Interaktionskonzept durch Probanden bewertet werden soll. Während der Durchführung dieser Arbeit wurde bereits eine Pilotstudie mit einer kleinen Probandenzahl durchgeführt.

Die Studie ist in zwei Teile gegliedert:

1. Im ersten Teil sollen die Probanden selbst Gesten zur Interaktion mit Powerwall-Visualisierungen beschreiben, die für den jeweiligen Probanden „intuitiv“ erscheinen. Für eine zukünftige Evaluation sollen in einer Hauptstudie so Gesten gefunden werden, die allgemein intuitiv und leicht zu erlernen sind.

2. Im zweiten Teil sollen die Probanden das im Prototyp implementierte Gestenkonzept zur Interaktion mit Powerwall-Visualisierungen bewerten. Zur Bewertung werden die Faktoren „passend“, „intuitiv“, „kompliziert“, „umständlich“, „leicht zu erlernen“ und „peinlich“ sowie standardisierte Variablen des NASA TLX (Hart, 2006) erhoben.

Basierend auf den Ergebnissen der Pilotstudie wurden Vorschläge zur Anpassung des Studiendesigns abgeleitet.

Im Rahmen dieser Diplomarbeit wurde auch ein Kommunikationsframework mit implementiert, das den Zusammenschluss von gestengesteuerter Powerwall und anderen Geräten, wie beispielsweise Tabletop-Computern, zu einer multimodalen Interaktionsumgebung ermöglicht. Durch dieses Kommunikationsframework können alle angeschlossenen Powerwalls und Tabletop-Computer auf dieselben Visualisierungsdaten zugreifen und Nachrichten untereinander austauschen. Das Kommunikationsframework verbindet die Prototypen dieser beiden Arbeiten, sodass beispielsweise Manipulationen, die an einer Powerwall-Visualisierung vorgenommen werden an die entsprechende Visualisierung auf einem Tabletop-Computer übertragen werden kann.
Der Fokus bei der Erstellung dieses Frameworks lag hierbei auf der Erweiterbarkeit des Frameworks sowie der Datenunabhängigkeit, sodass dieses Framework in möglichst vielen Szenarien eingesetzt werden kann.

9.1.2 Bewertung

In diesem Abschnitt werden einige Punkte aufgegriffen, die während der Durchführung dieser Arbeit positiv aufgefallen sind oder Schwierigkeiten bereitet haben. Insbesondere das Kommunikationsframework und die Gesteninteraktion zeigen Potential zur Anwendung und Weiterentwicklung. Allerdings sind deutliche technische Mängel bei der Handerkennung aufgefallen. Im Folgenden werden das Kommunikationsframework, das Interaktionskonzept und der Microsoft Kinect-Sensor einzeln betrachtet.

Kommunikationsframework

Bereits vor dem Prototyp zur Gesteninteraktion mit Powerwall-Visualisierungen wurden das Kommunikationsframework und die multimodale Interaktionsumgebung konzipiert. Diese Reihenfolge ermöglichte eine enge und erfolgreiche Zusammenarbeit mit der Diplomarbeit „Gestensteuerung für Tabletop-basierte Visualisierungen“ über die gesamte Dauer dieser Arbeit. Schnell zeigte sich, dass eine kombinierte Interaktion mit Daten durch Powerwall und Tabletop-Computer realisierbar ist.

Durch die Verwendung von Web Services zur Kommunikation zwischen den Geräten der multimodalen Interaktionsumgebung ist es ohne großen Aufwand möglich, zusätzlich zur Powerwall und Tabletop-Computern weitere Gerätetypen wie Notebooks oder Tablet-PCs in die Umgebung zu integrieren. Dabei ist die Integration unabhängig von dem Betriebssystem des Gerätes und der verwendeten Programmiersprache.

Die Visualisierungen zur Darstellung der Daten sind als Plugins implementiert. Dies ermöglicht die nachträgliche Erstellung zusätzlicher Visualisierungen für die Daten der Interaktionsumgebung. Ebenso können die Visualisierungen zwischen den verschiedenen Geräten der Interaktionsumgebung ausgetauscht werden.

Interaktionskonzept

Bei der Konstruktion des Interaktionskonzepts wurde das Szenario zur Analyse von Eyetracking-Daten zugrunde gelegt. Dies mindert zunächst vermutlich die allgemeine Anwendbarkeit des Interaktionskonzepts. Für eine prototypische Entwicklung des Konzepts war dies allerdings die richtige Entscheidung, da das Szenario konkrete Anforderungen an die Interaktionen stellt. Trotzdem wurden das im Prototyp implementierte Interaktionskonzept und insbesondere das Teilkonzept unterschiedlicher Interaktionsbereiche von den Probanden der Pilotstudie sehr gut angenommen. In einer weiteren Evaluation bietet sich die Möglichkeit, zusätzliche allgemeine Gesten zur Weiterentwicklung des Interaktionskonzepts zu finden.
Die Implementierung des Interaktionskonzepts benötigte mehrere Anläufe. Hieraus resultieren die drei verschiedenen Varianten zum Auslösen von Gesteninteraktionen. Die Erkennung war in den ersten Varianten nicht sehr robust. Hier ergeben sich für die Zukunft weitere Forschungsmöglichkeiten.

Microsoft Kinect-Sensor

Der Microsoft Kinect-Sensor bietet eine kostengünstige Variante zur Personen- und Gestenerkennung im Vergleich zu professionellen Tracking-Systemen. Ein weiterer Vorteil besteht darin, dass Benutzer nicht durch spezielle Marker für die Interaktion gekennzeichnet werden müssen. Die Skeletterkennung, die in den Sensor integriert ist, liefert äußerst zuverlässige Ergebnisse. Die Handerkennung, die zusätzlich zur Skeletterkennung für die Interaktion im Nah-Interaktionsbereich eingesetzt wird, funktioniert jedoch nur unbefriedigend: Die Handerkennung ist auf einen bestimmten Bereich vor dem Kinect-Sensor beschränkt und die Erkennungsrate ist sehr gering. Darüber hinaus ist die Position, an der eine Hand erkannt wird, nicht kontinuierlich sondern sprunghaft. Deshalb wurde die Handerkennung mit Skelettdaten kombiniert. Diese Maßnahme verbesserte allerdings nur die Positionserkennung der Hände. Auch die Verwendung des Near-Mode, der bei der neuen Generation des Kinect-Sensors für Windows verfügbar ist, konnte die Erkennungsrate der Hände nicht verbessern. In Kapitel 9.2.1 sind daher Vorschläge aufgelistet, wie durch Anpassung der Algorithmen die Handerkennungsrate verbessert werden könnte.

9.2 Ausblick

Während der Durchführung dieser Arbeit kamen ständig neue Ideen und Möglichkeiten auf, wie das Interaktionskonzept weiterentwickelt, die Gestenerkennung noch effektiver oder die Handerkennung noch genauer gestaltet werden könnten. Nicht alle Ideen konnten im Rahmen dieser Diplomarbeit umgesetzt werden. Im Folgenden werden deshalb weitere Ideen zur Verbesserung der Handerkennung, Vorschläge zur Erweiterung der Interaktion mit Powerwall-Visualisierungen und Ideen für weitere Benutzerstudien vorgestellt.

9.2.1 Verbesserung der Handerkennung

Die Handerkennung durch den Kinect-Sensor stellt das größte Manko bei der Gestenerkennung dar. Es stellte sich heraus, dass die Verwendung der Bibliothek „Candescent NUI“ (Stegmueller, 2012) zur Erkennung von Händen und Fingern keine zuverlässigen Ergebnisse liefert. Folgende Ideen könnten zu einer Verbesserung der Hand- und Fingererkennung beitragen und werden im Folgenden kurz vorgestellt: Einbeziehung von Hintergrundinformationen, Verwendung des Earth Mover’s Distance Algorithmus und Handerkennung mithilfe eines 3D-Modells.

Einbeziehung von Hintergrundinformationen

In (Izadi, et al., 2011) wird der Kinect-Sensor verwendet, um eine statische dreidimensionale Szene digital zu rekonstruieren. Nach der Digitalisierung kann der Benutzer mit der Hand die virtuelle Szene berühren. Da der Hintergrund der Szene durch die digitale Rekonstruktion bekannt ist, kann die Hand im Vordergrund leicht herausgefiltert werden. Es entsteht ein rauscharmes Bild, das eine zuverlässigere Erkennung von Händen und Fingern ermöglicht.

Abbildung 104: Handerkennung durch Differenzbild mit einem statischen Hintergrund (Izadi, et al., 2011).

Da der Hintergrund bei der Interaktion an der Powerwall nicht zwingend statisch sein muss, ist zu prüfen, ob durch Hintergrundinformationen eine Verbesserung der Erkennung der Hände vor der Powerwall möglich ist.

Earth Mover's Distance - Algorithmus

(Ren, Yuan, & Zhang, 2011) schlägt eine Variante des Earth Mover's Distance-Algorithmus zur Hand- und Fingererkennung vor. Der Earth Mover's Distance-Algorithmus ermittelt den minimalen Aufwand zwei Featuresets ineinander zu überführen (Rubner, Tomasi, & Guibas, 1998). Bei der Handerkennung dient das Abstandsprofil der Kontur einer vermutlichen Hand (schwarze Linie in Abbildung 105) zum vermuteten Mittelpunkt (hellblauer Punkt in Abbildung 105) als Featureset. Durch Vergleiche mit zuvor gespeicherten Featuresets erkannter Hände kann bestätigt werden, ob es sich um eine Hand handelt. Ebenso kann verglichen werden, wie viele Finger an der Hand erkannt worden sind oder ob es sich beispielsweise um eine geballte Faust handelt.

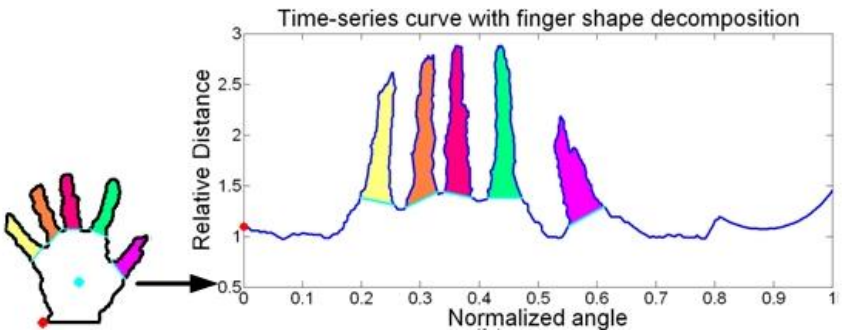


Abbildung 105: Hand- und Fingererkennung mithilfe eines Abstandsprofils.

Es ist zu prüfen, ob die Verwendung dieses Algorithmus eine Verbesserung der Hand- und Fingererkennung liefern könnte. Besonders wichtig ist die Frage, ob Hände mithilfe des Earth Mover's Algorithmus auch in einem Abstand von mehr als einem Meter zur Powerwall zuverlässig erkannt werden können.

Handerkennung mit 3D-Modell

An der Universität Kreta wurde eine Methode entwickelt mit dem Kinect-Sensor und einem dreidimensionalen Modell Hände zu erkennen. Mit dieser Methode ist es möglich die Position aller Gelenke der Hand und somit die aktuelle Pose dieser Hand zu erkennen. In (Oikonomidis, Kyriazis, & Argyros, 2012) wird angegeben, dass die Erkennung zuverlässig bis zu einem Abstand von 2,5 Meter zum Sensor mit einem maximalen Fehler von 8mm funktioniert.

Im Gegensatz zu den Bilderkennungsverfahren werden in dieser Methode Tiefenbilder berechnet, die das 3D-Modell der Hand erzeugen würde, wenn es in einer bestimmten Pose vom Kinect-Sensor erfasst würde. Als Ergebnis wird diejenige Pose der Modellhand gewählt, deren erwartetes Tiefenbild am ehesten mit dem vom Kinect-Sensor erkannten Tiefenbild übereinstimmt. Da diese Methode sehr rechenintensiv ist, ist ein Tracking bisher nur mit einer Frequenz von 4 Hz möglich.

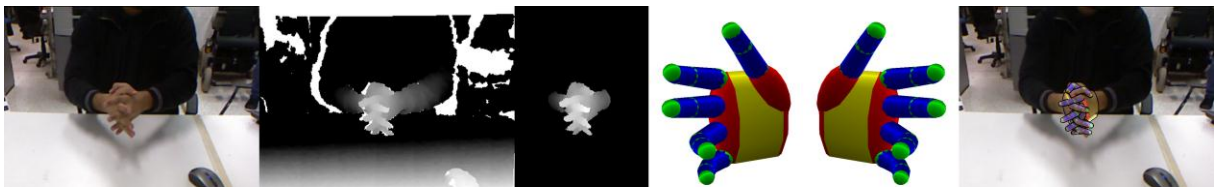


Abbildung 106: Erkennung zweier interagierender Hände. RGB-Bild, Tiefenbild, Hände im Tiefenbild, 3D-Modelle der Hände, Überlagerung der erkannten Hände mit den Modellen (v.l.n.r.) (Oikonomidis, Kyriazis, & Argyros, 2012).

9.2.2 Erweiterungen der Interaktion

Dieser Abschnitt präsentiert einige Ideen, mit denen sich die Gesteninteraktion mit Powerwall-Visualisierungen und die multimodale Interaktionsumgebung erweitern lassen. Folgende Ideen werden vorgestellt: Mehrbenutzer-Interaktion, Customization, Multi-Kinect-Szenario, weitere Visualisierungen und weitere Geräte für die Interaktionsumgebung.

Mehrbenutzer-Interaktion

Der im Rahmen dieser Diplomarbeit erstellte Prototyp ist für die Interaktion mit Powerwall-Visualisierungen durch lediglich einen Benutzer ausgelegt. Dies führt zu ungewünschtem Verhalten, wenn sich mehrere Personen im Erkennungsbereich des Kinect-Sensors aufhalten.
Es sollte eines der in Kapitel 5.1.4 vorgeschlagenen Muster zur Unterscheidung mehrerer Personen und zur Identifikation des Benutzers implementiert werden.

Customization

Gelingt es mehrere Personen im Interaktionsbereich des Kinect-Sensors zu unterscheiden, könnte folgende Erweiterung des Natural User Interfaces vorgenommen werden: Für jeden Benutzer, der mit der Powerwall interagiert wird ein Benutzerprofil erstellt. In diesem Profil werden Informationen gespeichert, wie sicher der Benutzer im Umgang mit dem Gesteninterface ist. Entsprechend der Erfahrung des Benutzers können die Hilfestellungen angepasst werden.
So könnte beispielsweise ein erfahrener Benutzer auf die Hilfestellung verzichten, die anzeigt, an

welchen Punkten der Powerwall seine Hände für die Interaktion erkannt wurden. Ebenso könnte die Aktivierungszeit für Interaktionen bei erfahrenen Benutzern reduziert und so eine schnellere Interaktion mit den Visualisierungen ermöglicht werden.

Multi-Kinect-Szenario

Der Prototyp, der im Rahmen dieser Diplomarbeit erstellt wurde, verwendet zum Zeitpunkt der Fertigstellung dieser Arbeit nur die Hälfte der Powerwall im VR-Labor des Informatikgebäudes. Der VisualizationClient für die Powerwall-Interaktion läuft auf einem Rechner, der an einen Beamer der Powerwall angeschlossen ist.

Um die Interaktionsfläche zu vergrößern, könnte die andere Hälfte der Powerwall ebenfalls über einen Computer mit einem zweiten Kinect-Sensor und einer zweiten Instanz des VisualizationClients zur Powerwall-Interaktion angesteuert werden.

Darüber hinaus muss der VisualizationClient angepasst werden, sodass ein fließender Übergang von der einen zur anderen Hälfte der Powerwall möglich wird. Ebenso wäre mit dieser Konfiguration die Interaktion mit der Powerwall durch mehrere Benutzer gleichzeitig denkbar.

Weitere Visualisierungen

Der Prototyp zur Interaktion mit Eyetracking-Daten an der Powerwall soll zur Analyse von Eyetracking-Studien, wie sie am VIS-Institut durchgeführt werden, eingesetzt werden können. Im Prototyp wurden beispielhaft einige Visualisierungen im Rahmen dieser Diplomarbeit implementiert. Weitere Visualisierungen für Eyetracking-Daten werden in (Chen, 2011) vorgeschlagen. Diese sollten ebenfalls in Form von Plugins für die multimodale Interaktionsumgebung implementiert werden. Das Plugin-System erlaubt die einfache Erstellung dieser Visualisierungen ohne den VisualizationClient ändern zu müssen. Auch für die Benutzer der multimodalen Interaktionsumgebung gliedern sich diese neuen Visualisierungen ohne Aufwand in das bestehende System ein.

Weitere Geräte für die multimodale Interaktionsumgebung

Um die Vorteile der privaten Räume nutzen zu können, müssen Ein-Benutzer-Geräte in die Interaktionsumgebung integriert werden. Durch die generische Konstruktion des Kommunikationsframeworks können beliebige Geräte einfach in die Umgebung integriert werden. Beispielsweise könnte ein VisualizationClient für Notebooks und Tablet-PCs entwickelt werden. Über einen QR-Code auf der Rückseite des Notebooks kann das Notebook vom Kinect-Sensor erkannt und in der Interaktionsumgebung angemeldet werden. Gleichzeitig kann das Notebook einem bestimmten Benutzer zugeordnet werden. Dieser Benutzer erhält für sein Schatteninterface ein weiteres Menü-Item, welches das Notebook repräsentiert. Mit einer Geste kann der Benutzer dann Ansichten und Daten in den privaten Bereich des Notebooks kopieren und dort weiterverarbeiten. Ebenso können Ansichten und Daten vom Notebook wieder in den öffentlichen Bereich der Powerwall übertragen werden.

Darüber hinaus kann der VisualizationClient des Notebooks die Powerwall und den Inhalt des öffentlichen Raums fernsteuern.

9.2.3 Studie zur multimodalen Interaktionsumgebung

Im Rahmen dieser Diplomarbeit wurde eine Pilotstudie durchgeführt, in der Probanden die im Prototyp implementierten Gesteninteraktionen bewerten und eigene Gesten zur Interaktion vorschlagen konnten. In einer zukünftigen Evaluation könnte eine große Benutzerstudie mit denselben Fragestellungen durchgeführt werden. Mit dieser Studie würde aber ausschließlich das Interaktionskonzept mit der Powerwall bewertet.

Das Konzept zur Gesteninteraktion mit Powerwall-Visualisierungen kann mithilfe des Kommunikationsframeworks in eine multimodale Interaktionsumgebung integriert werden. Durch den Einsatz weiterer Modalitäten in einer solchen Umgebung ergeben sich Synergieeffekte der verwendeten Interaktionskonzepte. Weiterführende Forschungsschwerpunkte sollten die Synergien der verschiedenen Interaktionsmöglichkeiten durch Probanden bewerten lassen. Folgende Fragestellungen sind beispielsweise in einer Interaktionsumgebung denkbar, die Powerwall-Interaktionen mit Tabletop-Computern kombiniert:

- Was sind die Vorteile der Kombination von Powerwall und Tabletop-Computer bei der Interaktion mit Visualisierungen?
- Inwiefern ergänzen sich die beiden Modalitäten?
- Wo treten Konflikte zwischen den beiden Modalitäten auf?
- Wie wird das Konzept der privaten und öffentlichen Räume bewertet?
- Welche Erweiterungen der multimodalen Umgebung sind gewünscht?
- Wieweit werden Multi-User-Szenarien unterstützt?

Die Ergebnisse solcher Studien können zur Homogenisierung der verschiedenen Interaktionskonzepte dienen und Ideen zur Weiterentwicklung der multimodalen Interaktionsumgebung liefern.

Literaturverzeichnis

The Visualization Handbook. (2005). Oxford, UK: Elsevier Inc., ISBN 0-12-387582-X

Kinect Hacking 103: Looking at Kinect IR Patterns. (17. November 2010). Abgerufen am 20. Juni 2012 von future picture: http://www.futurepicture.org/?p=116

VisWall. (2012). Abgerufen am 20. Juni 2012 von visbox.com: http://visbox.com/imgs/wallCCR4b.jpg

AMD. (2012). *ATI Eyefinity*. Abgerufen am 20. Juni 2012 von AMD: http://www.amd.com/de/products/technologies/eyefinity/Pages/eyefinity.aspx

Arman, F., & Aggarwal, J. K. (1993). Model-based object recognition in dense-range images - a review. *ACM Comput. Surv. 25, 1*, (S. 5-43)., DOI 10.1145/151254.151255

Blascheck, T. (2012). *Eye-Tracking-basiertes Analysekonzept zur Evaluation von Visualisierungen.* Diplomarbeit, Universität Stuttgart, Stuttgart.

Bolt, R. A. (1980). "Put-that-there": Voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph. 14, 3*, (S. 262-270)., DOI 10.1145/965105.807503

Bragdon, A., DeLine, R., Hinckley, K., & Morris, M. R. (2011). Code space: touch + air gesture hybrid interactions for supporting developer meetings. *Proceedings of the ACM International Conference on Interactive Tabletops and Sufraces (ITS '11)* (S. 212-221). New York, NY, USA: ACM., DOI 10.1145/2076354.2076393

Brooke, J. (1996). SUS: a "quick and dirty" usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland, *Usability Evaluation in Industry.* London: Taylor and Francis.

Cabral, M. C., Morimoto, C. H., & Zuffo, M. K. (2005). On the usability of gesture interfaces in virtual reality environments. *Proceedings of the 2005 Latin American conference on Human-computer interaction (CLIHC '05)* (S. 100-108). New York, NY, USA: ACM., DOI 10.1145/1111360.1111370

Chen, X. (2011). *Visuelle Analyse von Eye-Tracking-Daten.* Diplomarbeit Nr. 3183, Universität Stuttgart, Stuttgart.

Collins, C. (2006). DocuBurst: Document Content Visualization Using Language Structure. *Proceedings of IEEE Symposium on Information Visualization (Poster Session).* Baltimore.

Czerwinski, M., Robertson, G., Meyers, B., Smith, G., Robbins, D., & Tan, D. (2006). Large display research overview. *CHI '06 extended abstracts on Human factors in computing systems* (S. 69-74). New York, NY, USA: ACM., DOI 10.1145/1125451.1125471

Dahm, M. (2006). *Grundlagen der Mensch-Computer-Interaktion.* München: Pearson Studium., ISBN 978-3827371751

Davison, A. (2011). *Kinect Chapter 4. Tracking Users in 2D*. Abgerufen am 21. Juni 2012 von Andrew Davison's Home Page at PSU: http://fivedots.coe.psu.ac.th/~ad/jg/nui15

Deller, M., Agne, S., Ebert, A., Dengel, A., Hagen, H., Klein, B., et al. (2008). Managing a Document-Based Information Space. *Proceedings of the 13th international conference on Intelligent user interfaces (IUI '08)* (S. 119-128). New York, NY, USA: ACM., DOI 10.1145/1378773.1378789

Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). *Human-Computer Interaction.* Harlow: Pearson Education., ISBN 978-0130461094

Droeschel, D., Stückler, J., & Behnke, S. (2011). Leraning to interpret pointing gestures with a time-of-flight camera. *Proceedings of the 6th international conference on Human-robot interaction (HRI '11)* (S. 481-488). New York, NY, USA: ACM., DOI 10.1145/1957656.1957822

Duchowski, A. (2007). *Eye Tracking Methodology: Theory and Practice.* Springer., ISBN 978-1846286087

Ertl, T. (2010). *Visualization.* Vorlesung, Universität Stuttgart, Stuttgart.

Fallman, D., Kruzeniski, M., & Andersson, M. (2005). Designing for a collaborative industrial environment: the case of the ABB Powerwall. *Proceedings of the 2005 conference on Designing for User eXperience (DUX'05).* (S. Article 41). New York, NY, USA: American Institute of Graphic Arts., Index: ISBN 1-59593-250-X

Fekete, J.-D., & Plaisant, C. (2002). Interactive information visualization of a million items. *IEEE Symposium on Information Visualization (INFOVIS 2002)*, (S. 117-124)., DOI 10.1109/INFVIS.2002.1173156

Foggon, D., Maharry, D., Ullman, C., & Watson, K. (2003). *Programming Microsoft .NET XML Web Services.* Microsoft Press Books., ISBN 978-0735619128

Fraunhofer IIS. (2010). *Sheet of light imaging*. Abgerufen am 21. Juni 2012 von Optical Inspection Systems - Fraunhofer Institute for Integrated Circuits: http://www.iis.fraunhofer.de/en/bf/ops/licthschnittmess

Fraunhofer IOSB. (2012). *SmartControlRoom*. Abgerufen am 4. Juli 2012 von Fraunhofer IOSB: http://www.iosb.fraunhofer.de/servlet/is/6620/

Garcia, J., & Zalevsky, Z. (2007). *Patentnr. 7,443,024 B2.* USA.

Gjerlufsen, T., Klokmose, C. N., Eagan, J., Pillias, C., & Beaudouin-Lafon, M. (2011). Shared substance: developing flexible multi-surface applications. *Proceedings of the 2011 annual conference on Human factors in computing systems (CHI'11)* (S. 3383-3392). New York, NY, USA: ACM., DOI 10.1145/1978942.1979446

Green, M., & Jacob, R. (1990). Software Architectures and Metaphors for Non-WIMP User Interfaces. *SIGGRAPH '90.*, DOI 10.1145/126640.126677

Grottel, S. (2. April 2012). *Welcome to the MegaMol™-Project-Website*. Abgerufen am 21. Juni 2012 von MegaMol: www.visus.uni-stuttgart.de/megamol

Hart, S. G. (2006). Nasa-Task Load Index (Nasa-TLX); 20 Years Later. *Human Factors and Ergonomics Society Annual Meeting Proceedings*, (S. 904-908). Santa Monica., DOI 10.1177/154193120605000909

Hinckley, K. (December 1996). *Haptic Issues for Virtual Manipulation.* Abgerufen am 20. Juni 2012 von Microsoft Research: http://research.microsoft.com/apps/pubs/default.aspx?id=68165

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcoombe, R., Kohli, P., et al. (2011). Kinect Fusion: real-time 3D reconstruction and interaction using a moving depth camera. *Proceedings of the 24th annual ACM symposium on User interface software and technology* (S. 559-568). New York, NY, USA: ACM., DOI 10.1145/2047196.2047270

Jacob, R. (1990). What you look at is what you get: eye movement-based interaction techniques. *Proceedings of the SIGCHI conference on Human factors in computing systems: Enpowering people (CHI '90)* (S. 11-18). New York, NY, USA: ACM., DOI 10.1145/97243.97246

Jacob, R. J., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovey, E. T., et al. (2008). Reality-based interaction: a framework for post-WIMP interfaces. *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI'08)* (S. 201-210). New York, NY, USA: ACM., DOI 10.1145/1357054.1357089

Johnson, A. (2010). *Next-Generation CAVE*. Abgerufen am 20. Juni 2012 von electronic visualization laboratory: http://www.evl.uic.edu/IE/core.php?mod=4&type=9&indi=421

Kabayama, A. M., & Trabasso, L. G. (2002). Performance evaluation of 3D computer vision techniques. *J. Braz. Soc. Mech. Sci. 24, 3*, (S. 234-238). Rio de Janeiro., DOI 10.1590/S0100-73862002000300013

Karam, M., & Schraefel, M. C. (2005). A taxonomy of Gestures in Human Computer Interaction. *Transactions on Computer-Human Interactions.* ACM., ISBN 0854328335

Keim, D. A. (Januar/März 2002). Information visualization and visual data mining. *Visualization and Computer Graphics*, S. 1-8., DOI 10.1109/2945.981847

Kinect Games. (2012). *Kinect Games List*. Abgerufen am 21. Juni 2012 von kgames.org: http://www.kgames.org/games/

König, W. A., Bieg, H. J., Schmidt, T., & Reiterer, H. (2007). Position-independent interaction for large high-resolution displays. *Proceedings of the IADIS International Conference on Interfaces and Human Computer Interaction* (S. 117-125). IADIS Press.

König, W. A., Rädle, R., & Reiterer, H. (2009). Squidy: a zoomable design environment for natural user interfaces. *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems (CHI EA '09)* (S. 4561-4566). New York, NY, USA: ACM., DOI 10.1145/1520340.1520700

Krumbholz, C., Leigh, J., Johnson, A., Renambot, L., & Kooima, R. (kein Datum). *Lambda Table: High Resolution Tiled Display Table for Interacting with Large Visualizations*. Abgerufen am 20. Juni 2012 von electronic visualization laboratory: http://www.evl.uic.edu/files/pdf/LambdaTable_krumbholz.pdf

Lyman, P., & Varian, H. R. (2000). How Much Information? University of California at Berkeley., DOI 10.3998/3336451.0006.204

Matejka, J., Grossmann, T., Lo, J., & Fitzmaurice, G. (2009). The design and evaluation of multi-finger mouse emulation techniques. *Proceedings of the 27th international conference on Human factors in computing systems (CHI '09)* (S. 1073-1082). New York, NY, USA: ACM., DOI 10.1145/1518701.1518865

Mauney, D. (2012). *Gesture Research Part 1 - 4*. Abgerufen am 20. 06 2012 von TouchThinking: http://www.touchthinking.com

McCormick, B. H., DeFanti, T. A., & Brown, M. D. (1987). *ACM SIGGRAPH Computer Graphics Volume 21 Issue 6.* New York, NY, USA: ACM., ISSN:0098-8930

Microsoft. (2009). *Project Natal - Paint Party E3 Stage Demo [HD]*. Abgerufen am 21. Juni 2012 von http://www.youtube.com/watch?v=Qu8_fJzN2Ik

Microsoft. (21. Oktober 2010). *Microsoft News Center*. Abgerufen am 20. Juni 2012 von Kinect Ads: 'You Are the Controller': http://www.microsoft.com/en-us/news/features/2010/oct10/10-21kinectads.aspx

Microsoft. (16. Juni 2011). *Microsoft Releases Kinect for Windows SDK Beta for Academics and Enthusiasts*. Abgerufen am 20. Juni 2012 von Microsoft News Center: http://www.microsoft.com/en-us/news/press/2011/jun11/06-16MSKinectSDKPR.aspx

Microsoft. (2012). *Case Studies*. Abgerufen am 20. Juni 2012 von Microsoft PixelSense: http://www.microsoft.com/en-us/pixelsense/casestudies.aspx

Microsoft. (2012). *Microsoft Developer Network MSDN*. Abgerufen am 20. 06 2012 von ADO.NET Entity Framework: http://msdn.microsoft.com/de-de/library/bb399572

Microsoft. (20. Januar 2012). *Near Mode: What it is (and isn't)*. Abgerufen am 21. Juni 2012 von KINECT for Windows Blog: http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/20/near-mode-what-it-is-and-isn-t.aspx

Ni, T., Schmidt, G. S., Staadt, O. G., Livingston, M. A., Ball, R., & May, R. (2006). A Survey of Large High-Resolution Display Technologies, Techniques and Applications. *Proceedings of the IEEE Virtual Reality Conference (VR'06)* (S. 223-236). IEEE., DOI 10.1109/VR.2006.20

Niemann, K. D. (1995). *Client-Server-Architektur: Organisation und Methodik der Anwendungsentwicklung.* Braunschweig: 1995., ISBN 3-528-05466-2

Oikonomidis, I., Kyriazis, N., & Argyros, A. A. (2012). *Tracking the Articulated Motion of Two Strongly Interacting Hands.* Universität Kreta, Kreta.

Püttmann, E. (2012). *Gestensteuerung für Tabletop-Computer-basierte Visualisierungen.* Diplomarbeit, Universität Stuttgart, Stuttgart.

Randell, R., Hutchins, G., Sandars, J., Ambepitiya, T., Treanor, D., Thomas, R., et al. (2012). Using a high-resolution wall-sized virtual microsocope to teach undergraduate medical students. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*

Extended Abstracts (CHI EA '12) (S. 2435-2440). New York, NY, USA: ACM., DOI 10.1145/2212776.2223815

Raschke, M., Chen, X., & Ertl, T. (2012). Parallel scan-path visualization. In S. N. Spencer (Hrsg.), *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)* (S. 165-168). New York, NY, USA: ACM., DOI 10.1145/2168556.2168583

Ren, Z., Yuan, J., & Zhang, Z. (2011). Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. *Proceedings of the 19th ACM international conference on Multimedia (MM '11)* (S. 1093-1096). New York, NY, USA: ACM., DOI 10.1145/2072298.2071946

Renambot, L. (kein Datum). Abgerufen am 20. Juni 2012 von http://www.evl.uic.edu/cavern/sage/gallery/PICT0032.jpg

Renambot, L. (kein Datum). Abgerufen am 20. Juni 2012 von http://www.evl.uic.edu/cavern/sage/gallery/lambdatable09.jpg

Renambot, L., Jeong, B., Jagodic, R., Johnson, A., & Leigh, J. (kein Datum). *Collaborative Visualization using High-Resolution Tiled Displays*. Abgerufen am 20. Juni 2012 von electronic visualization laboratory: http://www.evl.uic.edu/files/pdf/SAGE-CHI06-final.pdf

Rubner, Y., Tomasi, C., & Guibas, L. J. (1998). *The Earth Mover's Distance as a Metric for Image Retrieval.* Technical Report, Stanford, CA, USA., DOI 10.1023/A:1026543900054

Ruppert, G., Reis, L., Amorim, P., de Moraes, T., & da Silva, J. (24. April 2012). Touchless gesture user interface for interactive image visualization in urological surgery. *World Journal of Urology*, S. 1-5., DOI 10.1007/s00345-012-0879-0

Saffer, D. (2009). *Designing Gestural Interfaces.* O'Reilly., ISBN 978-0-596-51839-4

Schmidt, A. (2000). Implicit human computer interaction through context. *Personal and Ubiquitous Computing*, S. 191-199., DOI 10.1007/BF01324126

Scott, S. D. (2003). Territory-Based Interaction Techniques for Tabletop Collaboration. *Conference Companion of the ACM Symposium on User Interface Software and Technology UIST'03.*

Shneiderman, B. (1983). Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer 16(8)*, (S. 57-69)., DOI 10.1109/MC.1983.1654471

Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. *Proceedings of the IEEE Symposium on Visual Languages* (S. 336-343). Washington: IEEE Computer Society Press., DOI 10.1109/VL.1996.545307

Shneiderman, B. (2002). *User Interface Design.* Bonn: mitp., ISBN 3-8266-0753-8

Shoemaker, G., Tsukitani, T., Kitamura, Y., & Booth, K. S. (2010). Body-centric interaction techniques for very large wall displays. *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries (NordiCHI'10)* (S. 463-472). New York, NY, USA: ACM., DOI 10.1145/1868914.1868967

Soro, A., Iacolina, S. A., Scateni, R., & Uras, S. (2011). Evaluation of user gestures in multi-touch interaction: a case study in pair-programming. *Proceedings of the 13th international conference on multimodal interfaces (ICMI'11)* (S. 161-168). New York, NY, USA: ACM., DOI 10.1145/2070481.2070508

Spielberg, S. (Regisseur). (2002). *Minority Report* [Kinofilm].

statista. (2012). *Absatz von Smartphones in Deutschland in den Jahren 2008 bis 2010 (in Millionen Stück)*. Abgerufen am 20. Juni 2012 von statista: http://de.statista.com/statistik/daten/studie/77637/tab/2/umfrage/absatzmenge-fuer-smartphones-in-deutschland-seit-2008/

Stegmueller, S. (2012). *Candescent NUI.* Abgerufen am 20. Juni 2012 von codeplex: http://candescentnui.codeplex.com/

Streitz, N., Prante, T., Müller-Tomfelde, C., Tandler, P., & Magerkurth, C. (2002). Roomware: the second generation. *CHI'02 extended abstracts on Human factors in computing systems (CHI EA '02)* (S. 506-507). New York, NY, USA: ACM., DOI 10.1145/506443.506452

Sutherland, I. E. (1964). Sketch pad a man-machine graphical communication system. *Proceedings of the SHARE design automation workshop (DAC '64)* (S. 6329-6346). New York, NY, USA: ACM., DOI 10.1145/800265.810742

Tobii Technology AB. (2011). *Tobii*. Abgerufen am 20. 06 2012 von Tobii: http://www.tobii.com

van Someren, M. W., Barnard, Y. F., & Sandberg, J. A. (1994). *The Think Aloud Method - A practical guide to modelling cognitive processes.* London: Academic Press., ISBN 0-12-714270-3

Vertanen, K. (kein Datum). *NASA-TLX in HTML and JavaScript*. Abgerufen am 20. Juni 2012 von Fragebogen - Teil 1: http://www.keithv.com/software/nasatlx/nasatlx_german.html

Villaroman, N., Rowe, D., & Swan, B. (2011). Teachinig natural user interaction using OpenNI and the Microsoft Kinect sensor. *Proceedings of the 2011 conference of Information technology education (SIGITE '11)* (S. 227-232). New York, NY, USA: ACM., DOI 10.1145/2047594.2047654

VISUS. (24. Mai 2012). *Visualisierungslabor mit hochauflösender Großprojektionswand*. Abgerufen am 20. Juni 2012 von Visualisierungsinstitut der Universität Stuttgart: http://www.visus.uni-stuttgart.de/institut/visualisierungslabor.html

Weiser, M. (1991). The Computer for the 21st Century. *Scientific Am., 3*(265).

Wigdor, D., & Wixon, D. (2011). *Brave NUI World.* Burlington: Elsevier Inc., ISBN 978-0-12-382231-4

Wobbrock, J. O., Morris, M. R., & Wilson, A. D. (2009). User-defined gestures for surface computing. *Proceedings of the 27th international conference on Human factors in computing systems (CHI '09)* (S. 1083-1092). New York, NY, USA: ACM., DOI 10.1145/1518701.1518866

Wu, M., & Balakrishnan, R. (2003). Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. *Proceedings of the 16th annual ACM symposium on User*

interface software and technology (UIST '03) (S. 193-202). New York, NY, USA: ACM., DOI 10.1145/964696.964718

Anhang A: Allgemeiner Fragebogen

Demografische Angaben

Geschlecht:	M	W

Alter:	

Angaben zur Ausbildung

Höchster Bildungsabschluss:	
Beruf:	

Studiengang:					
Fachsemester:					
Vertiefung / Studienschwerpunkt:					
Angestrebter Abschluss:	Bachelor	Master	Staatsexamen	Diplom	

Vorkenntnisse in der Gesteninteraktion

Besitzen Sie eine XBOX 360 mit KINECT?	Ja	Nein
Besitzen Sie eine PlayStation 3 mit Move-Controller?	Ja	Nein
Besitzen Sie eine Nintendo Wii?	Ja	Nein
Wie viele Stunden spielen Sie mit dieser Konsole durchschnittlich pro Woche?		

Besitzen Sie ein Touch-Eingabegerät für Ihren PC? (Touchscreen, Convertible …)	Ja	Nein
Besitzen Sie ein iPad, iPod-Touch, Galaxy-Tab oder ein ähnliches Tablet?	Ja	Nein
Besitzen Sie ein touch-fähiges Smartphone?	Ja	Nein
Wie viele Stunden pro Woche verbringen Sie mit diesen Geräten?		

Besitzen Sie andere Vorkenntnisse im Bereich der Touch-Gesteninteraktion?	Ja	Nein
Falls ja, welche?		

Besitzen Sie andere Vorkenntnisse im Bereich der Freihand-Gesteninteraktion?	Ja	Nein
Falls ja, welche?		

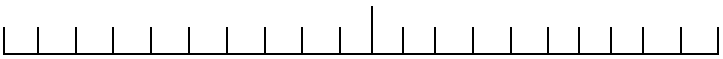
Anhang B: Fragebogen zur Gestenfindung

Für jede der sieben Interaktionen (Objekt verschieben, Objekt löschen, Objekt skalieren, Objekt duplizieren, Objekte koppeln, Objektansicht wechseln und Objektinhalt manipulieren) sollten die Probanden den folgenden Fragebogen ausfüllen. Beispielhaft ist der Fragebogen für die Interaktion „Objekt verschieben“ dargestellt.

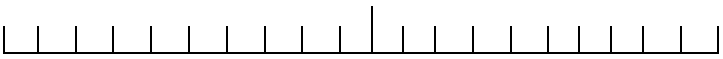
Beantworten Sie bitte jeweils die Fragen zu jeder Geste, die Sie in diesem Teil vorgeschlagen haben.

Interaktion: Objekt verschieben

Wie kompliziert schätzen Sie die von Ihnen vorgeschlagene Geste zum Verschieben von Objekten ein?

Unkompliziert  Kompliziert

Wie ermüdend schätzen Sie die von Ihnen vorgeschlagene Geste zum Verschieben von Objekten ein?

Sehr ermüdend  Kaum ermüdend

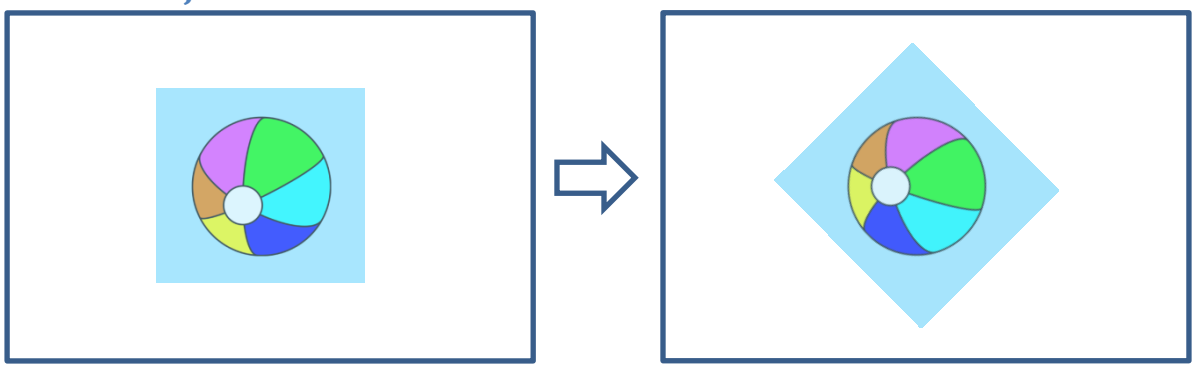
Wenn Sie mehrere Gestenvarianten zum Verschieben von Objekten vorgeschlagen haben, bewerten Sie diese bitte:

Anhang C: Ausgangs- und Zieldarstellungen des ersten Studienabschnitts

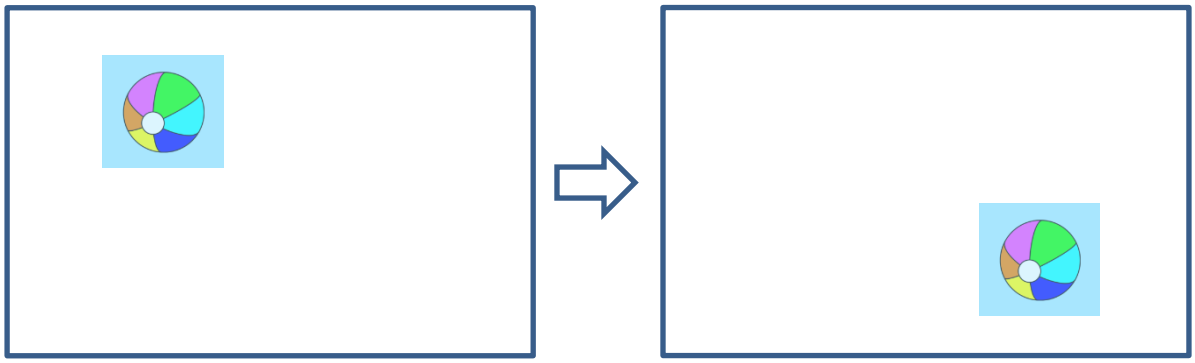
In diesem Anhang sind die Darstellungen aufgeführt, die den Teilnehmern der Pilotstudie im ersten Studienabschnitt gezeigt wurden. Jeweils auf der linken Seite ist die Ausgangsdarstellung zu sehen und rechts die Zieldarstellung, welche die Probanden durch das Ausführen einer Geste aus der Ausgangsdarstellung erreichen sollten.

Zu Beginn sind die Darstellungen aus dem Tutorial dargestellt. Aus der Ausgangsposition sollten die Teilnehmer die Ansicht im Uhrzeigersinn rotieren.

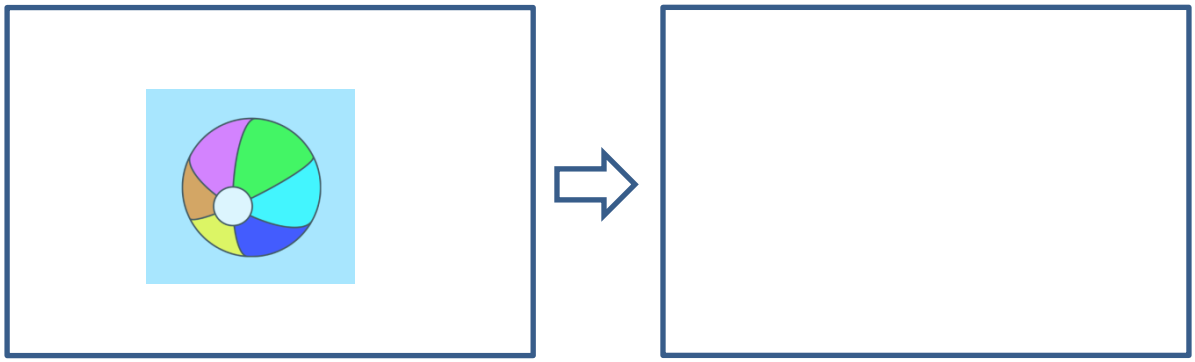
Tutorial: Objekt rotieren



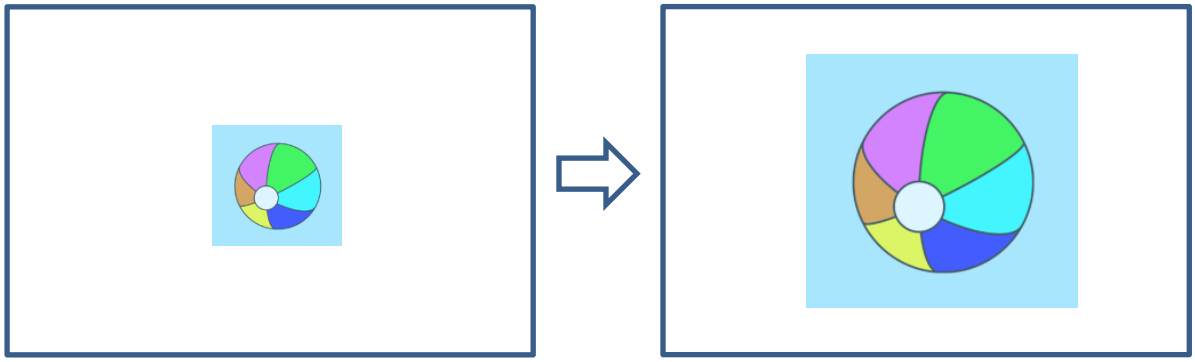
Aufgabe 1.1. Objekt verschieben



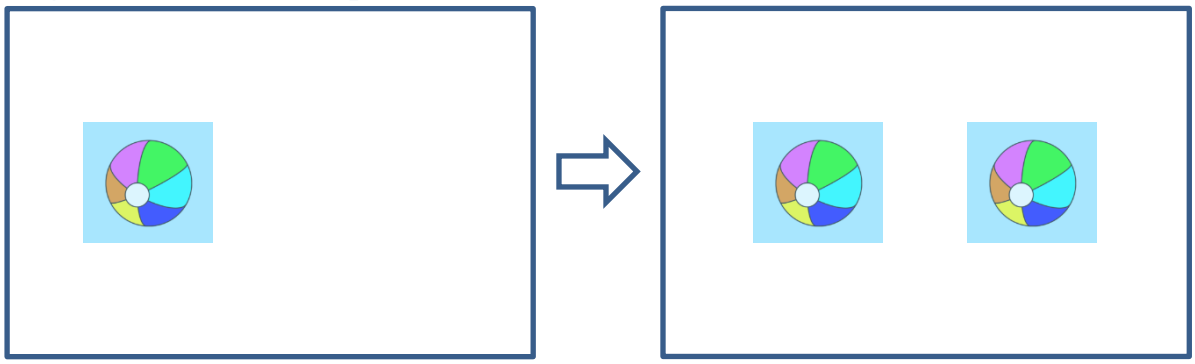
Aufgabe 1.2. Objekt löschen



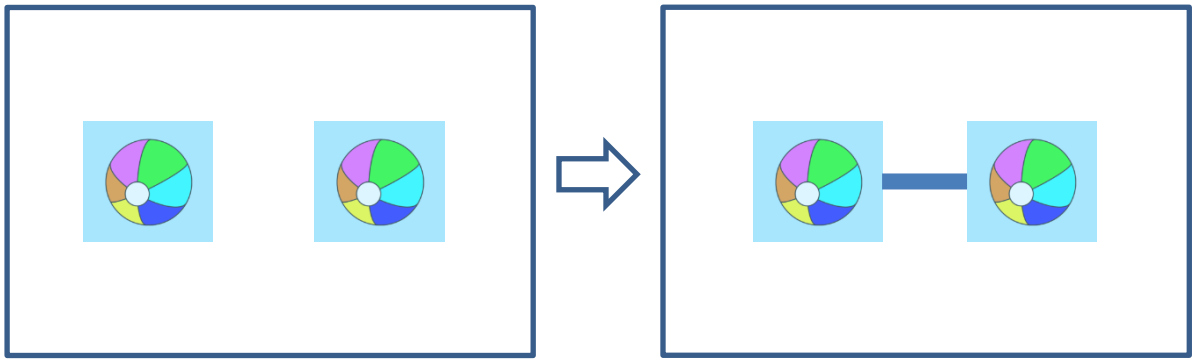
Aufgabe 1.3. Objekt skalieren



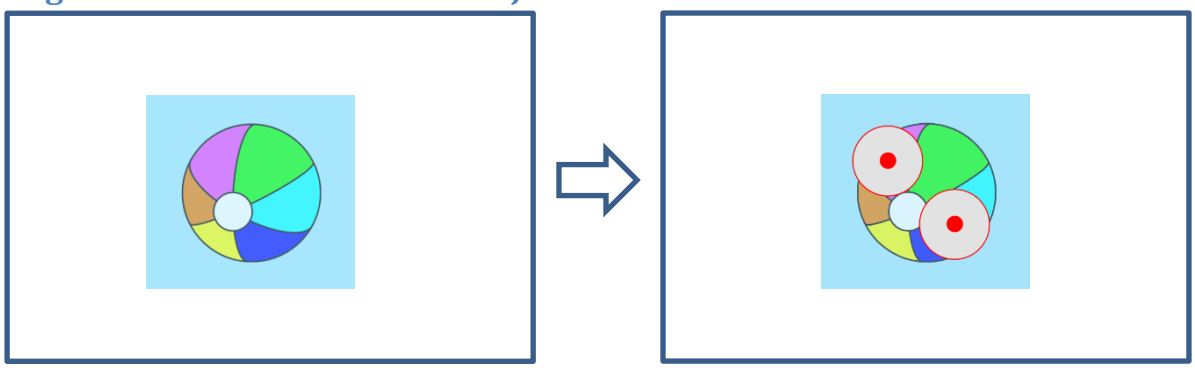
Aufgabe 1.4. Objekt duplizieren



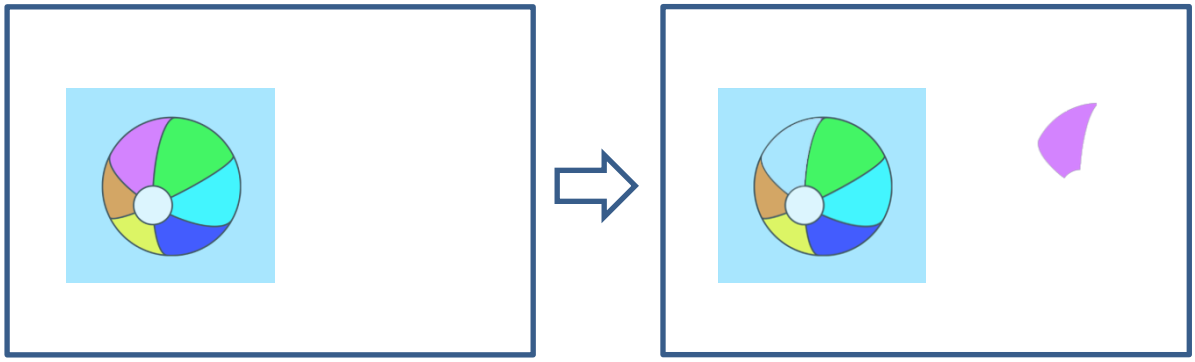
Aufgabe 1.5. Objekte koppeln



Aufgabe 1.6. Ansicht in einem Objekt wechseln



Aufgabe 1.7. Objektinhalt verändern



Anhang D: Fragebogen zur Beurteilung des Prototyps

Für jede der sieben Interaktionen (Objekt verschieben, Objekt löschen, Objekt skalieren, Objekt duplizieren, Objekte koppeln, Objektansicht wechseln und Objektinhalt manipulieren) sollten die Probanden den folgenden Fragebogen ausfüllen. Beispielhaft ist der Fragebogen für die Interaktion „Objekt verschieben" dargestellt.

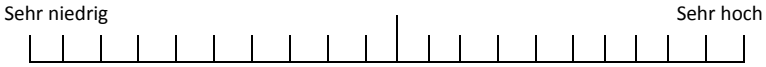
Beantworten Sie bitte nach jeder durchgeführten Interaktion die Fragen aus dem zugehörigen Abschnitt und bewerten sie die jeweilige Interaktion.

Interaktion: Objekt Verschieben

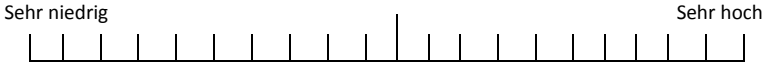
Meiner Meinung nach ist die Interaktion zum Verschieben eines Objekts auf der Powerwall …

	Ich stimme voll zu	Ich stimme teilweise zu	Ich stimme weniger zu	Ich stimme überhaupt nicht zu
… passend.				
… intuitiv.				
… kompliziert.				
… umständlich.				
… leicht zu erlernen.				
… peinlich.				

Mental Demand
Wie hoch war der geistige Anspruch der Interaktion?

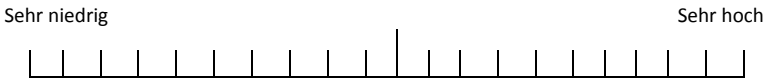


Physical Demand
Wie hoch war der körperliche Anspruch der Interaktion?



Performance
Wie erfolgreich war die Durchführung der Interaktion?

Frustration
Wie groß waren die Unsicherheit, der Ärger,
die Verwirrung oder die Entmutigung bei der Interaktion?



An dieser Interaktion hat mir folgendes besonders gefallen:

An dieser Interaktion ist mir folgendes negativ aufgefallen:

Anhang E: Fragebogen zum Gesamtsystem

Bewerten Sie bitte abschließend das gesamte System der Verwendung von Freihandgesten zur Interaktion mit Visualisierungen an einer Powerwall, inklusive der Hilfestellungen und der Nah-/Ferninteraktionsbereiche:

	Ich stimme voll zu	Ich stimme teilweise zu	Neutral	Ich stimme weniger zu	Ich stimme überhaupt nicht zu
Ich möchte dieses Konzept häufig benutzen.					
Das Konzept ist unnötig komplex.					
Das Konzept ist einfach anzuwenden.					
Man braucht einen Spezialisten um das Konzept anwenden zu können.					
Die verschiedenen Funktionen sind im Konzept gut integriert.					
Das Konzept ist inkonsistent.					
Die meisten Menschen können dieses Konzept einfach erlernen.					
Das Konzept ist umständlich.					
Ich fühlte mich sicher in der Anwendung des Konzepts.					
Ich musste viele Erklärungen erhalten bevor ich das Konzept anwenden konnte.					

	Ich stimme voll zu	Ich stimme teilweise zu	Ich stimme weniger zu	Ich stimme überhaupt nicht zu
… körperlich zu anstrengend / ermüdend um es längere Zeit verwenden zu können				

Meiner Meinung nach ist die Verwendung des gesamten Systems …

Beschreiben Sie bitte, in welchen Situationen oder in welchen Bereichen Sie sich vorstellen könnten, dass ein solches System Anwendung findet:

An diesem Konzept hat mir folgendes besonders gefallen:

An diesem Konzept ist mir folgendes negativ aufgefallen: