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

Diplomarbeit Nr. 3303

Tabletop-Computer-basierte Steuerung für Powerwall-Visualisierungen

Edwin Püttmann

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Thomas Ertl
Betreuer:	Prof. Dr. Albrecht Schmidt Dipl.-Phys. Michael Raschke Dipl.-Inf. Bastian Pfleging
begonnen am:	20.02.2012
beendet am:	21.08.2012
CR-Klassifikation:	H.5.2 H.5.3 I.3.6

Kurzfassung

Bei der Analyse von Daten aus komplexen Themengebieten werden trotz moderner visueller Werkzeuge oft interessante Muster und Details nach wie vor leicht übersehen. Aus diesem Grund werden komplexe Daten meist von mehreren Personen analysiert. Dazu können großflächige Displays zur Anzeige der Daten genutzt werden, die jedoch mit herkömmlichen Eingabegeräten, wie Maus und Tastatur, sich nur sehr eingeschränkt bedienen lassen. Seit einigen Jahren stehen Geräte zur kollaborativen Arbeit zur Verfügung beispielsweise in Form von Tabletops. Diese Arbeit stellt ein Interaktionskonzept für die kollaborative Arbeit von Benutzern mit einer Powerwall basierend auf einer Tabletop-Steuerung vor. Es wurde auf Basis des Konzepts ein Framework entwickelt, das die Steuerung einer Powerwall durch einen Tabletop sowie die Einbindung von weiteren Geräten zur Interaktion ermöglicht. Das Interaktionskonzept der Tabletop-Steuerung wurde anhand einer Studie evaluiert und daraufhin ein Prototyp entwickelt.

Abstract

While analyzing data belonging to complex subjects, interesting details or patterns are still easily overlooked, even with modern visualization tools. Therefore, those analyses are mostly performed by several persons. For this, large-scaled screens are used to display the data, but those screens are only limited operable with common input devices, like keyboard and mouse. For a couple of years devices for collaborative work are available, for example in the form of tabletops. This thesis introduces a concept for the interaction of users with a powerwall and a tabletop-based control for collaborative work. Based on this concept, a framework was developed which allows the control of a powerwall through a tabletop and the integration of additional devices. The interaction concept of the tabletop control was evaluated by a study and subsequently a prototype was developed.

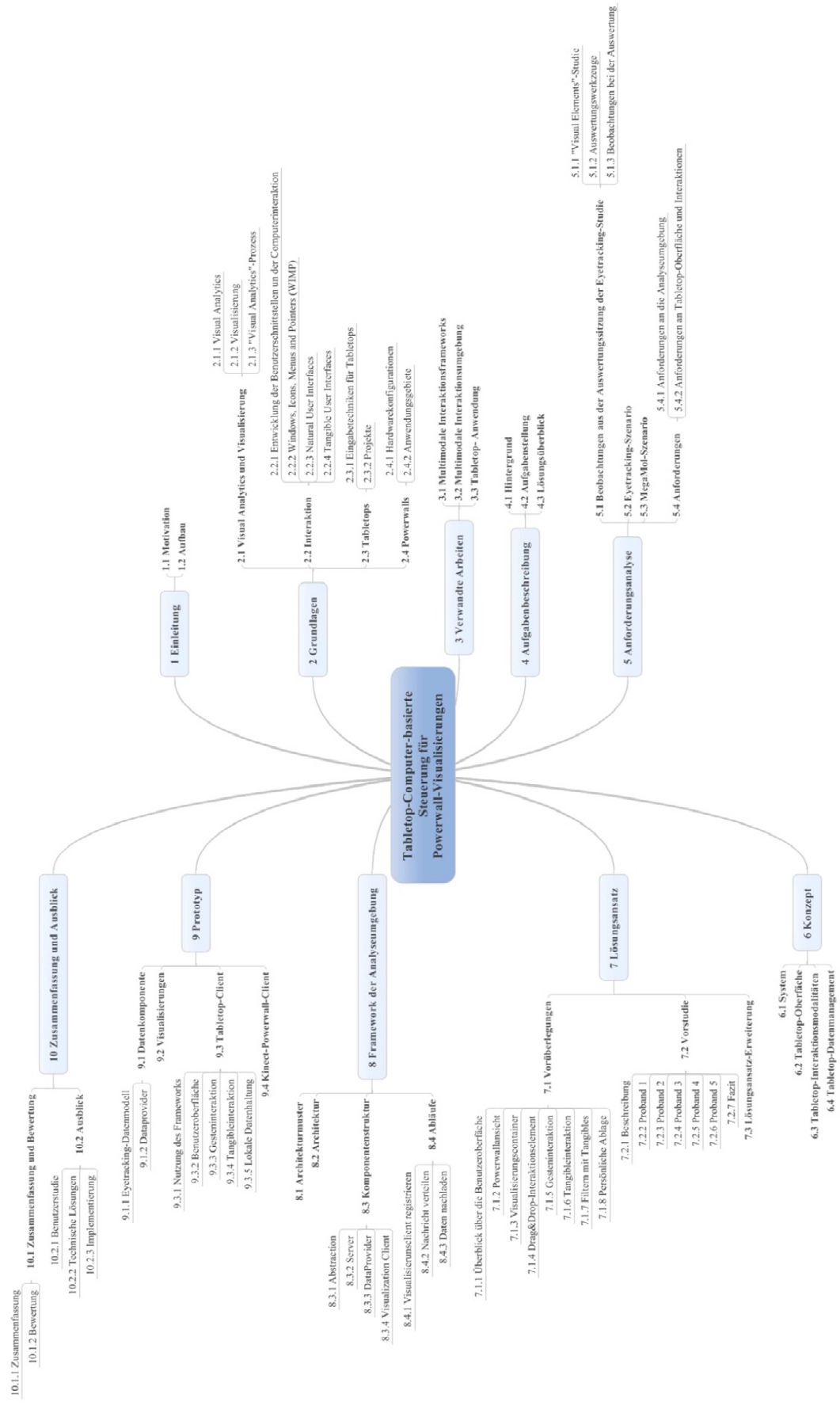
Inhalt

Inhalt als Mind-Map 9
Abbildungsverzeichnis 11
Tabellenverzeichnis 15
1 Einleitung 17
1.1 Motivation 17
1.2 Aufbau 18
2 Grundlagen 19
2.1 Visual Analytics und Visualisierung 19
2.1.1 Visual Analytics 19
2.1.2 Visualisierung 20
2.1.3 „Visual Analytics“-Prozess 22
2.2 Interaktion 24
2.2.1 Entwicklung der Benutzerschnittstellen und der Computerinteraktion 24
2.2.2 Windows, Icons, Menus and Pointers (WIMP) 26
2.2.3 Natural User Interfaces 27
2.2.4 Tangible User Interfaces 28
2.3 Tabletops 30
2.3.1 Eingabetechniken für Tabletops 31
2.3.2 Technologiebeispiele 36
2.4 Powerwalls 37
2.4.1 Hardwarekonfigurationen 37
2.4.2 Anwendungsgebiete 38
3 Verwandte Arbeiten 41
3.1 Multimodale Interaktionsframeworks 41
3.2 Multimodale Interaktionsumgebung 42
3.3 Tabletop-Anwendungen 43
4 Aufgabenbeschreibung 45
4.1 Hintergrund 45
4.2 Aufgabenstellung 45
4.3 Lösungsüberblick 46
5 Anforderungsanalyse 49
5.1 Beobachtungen aus der Auswertungssitzung der Eyetracking-Studie 49
5.1.1 „Visual Elements“-Studie 49
5.1.2 Auswertungswerkzeuge 51
5.1.3 Beobachtungen bei der Auswertung 53
5.2 Eyetracking-Szenario 54

5.3 MegaMol-Szenario ... 60
5.4 Anforderungen ... 64
5.4.1 Anforderungen an die Analyseumgebung ... 64
5.4.2 Anforderungen an Tabletop-Oberfläche und Interaktionen ... 65
6 Konzept ... 67
6.1 System ... 67
6.2 Tabletop-Oberfläche ... 69
6.3 Tabletop-Interaktionsmodalitäten ... 72
6.4 Tabletop-Datenmanagement ... 75
7 Lösungsansatz ... 77
7.1 Vorüberlegungen ... 77
7.1.1 Überblick über die Benutzeroberfläche ... 77
7.1.2 Powerwallansicht ... 78
7.1.3 Visualisierungscontainer ... 79
7.1.4 Drag&Drop-Interaktionselement ... 81
7.1.5 Gesteninteraktion ... 82
7.1.6 Tangibleinteraktion ... 82
7.1.7 Filtern mit Tangibles ... 83
7.1.8 Persönliche Ablage ... 84
7.2 Vorstudie ... 85
7.2.1 Beschreibung ... 85
7.2.2 Proband 1 ... 94
7.2.3 Proband 2 ... 95
7.2.4 Proband 3 ... 96
7.2.5 Proband 4 ... 98
7.2.6 Proband 5 ... 99
7.2.7 Fazit ... 101
7.3 Lösungsansatz-Erweiterung ... 105
8 Framework der Analyseumgebung ... 109
8.1 Architekturmuster ... 109
8.2 Architektur ... 109
8.3 Komponentenstruktur ... 111
8.3.1 Abstraction ... 112
8.3.2 Server ... 113
8.3.3 DataProvider ... 115
8.3.4 VisualizationClient ... 116
8.4 Abläufe ... 121
8.4.1 Visualisierungsclient registrieren ... 121

8.4.2 Nachricht verteilen ... 121
8.4.3 Daten nachladen ... 123
9 Prototyp ... 125
9.1 Datenkomponente der Analyseumgebung ... 125
9.1.1 Eyetracking-Datenmodell ... 125
9.1.2 Dataprovider ... 128
9.2 Visualisierungen ... 128
9.3 Tabletop-Client ... 129
9.3.1 Nutzung des Frameworks ... 130
9.3.2 Benutzeroberfläche ... 130
9.3.3 Gesteninteraktion ... 135
9.3.4 Tangibleinteraktion ... 135
9.3.5 Lokale Datenhaltung ... 138
9.4 Kinect-Powerwall-Client ... 139
10 Zusammenfassung und Ausblick ... 141
10.1 Zusammenfassung und Bewertung ... 141
10.1.1 Zusammenfassung ... 141
10.1.2 Bewertung ... 143
10.2 Ausblick ... 144
10.2.1 Benutzerstudie ... 144
10.2.2 Technische Lösungen ... 144
10.2.3 Implementierung ... 146
Literaturverzeichnis ... 149
Anhang A: Fragebogen ... 154

Inhalt als Mind-Map



Abbildungsverzeichnis

Abbildung 1: Visualisierungspipeline [16] 21
Abbildung 2: Visual Analytics Prozess [14] 23
Abbildung 3: Skizze des Memex [20] 24
Abbildung 4: Sutherland bei der Arbeit mit dem Lichtgriffel 25
Abbildung 5: Ein Beispiel eines Tangible User Interface, die Marbel Answering Machine .. 26
Abbildung 6: Windows 7 hat eine typische WIMP-Benutzerschnittstelle 27
Abbildung 7: TUIs führen weg von der Bedienung des Computers mit Maus und Tastatur. Alle Elemente des Raums dienen zur Interaktion. [24] 29
Abbildung 8: Oben: Ein Slot eines MediaBlocks auf den Bilder übertragen werden. Unten: Ein MediaSequenzer, um Medien zu bearbeiten. [25] 30
Abbildung 9: Benutzer beim Musikmachen mit dem ReacTable. (Interaktion mit Tangibles) [26] 30
Abbildung 10: Skizze zum Oberflächenkapazitätsverfahren [28] 32
Abbildung 11: Skizze zum Verfahren der „Projizierte Kapazität“ [28] 32
Abbildung 12: Darstellung von FTIR [28] 33
Abbildung 13: Darstellung zu DI [28] 34
Abbildung 14: Darstellung zu DSI [28] 35
Abbildung 15: Darstellung der Pixelsense Hardware [8] 35
Abbildung 16: Beispiel für Tiled LCD Panels [34] Links: Die lambdaVision Rechts: Der lambdaTable 37
Abbildung 17: Aufbau der Powerwall im VISUS-Labor mit einem Projektor Array [35] 38
Abbildung 18: Beispiel eines autostereoskopischen Displays(The Varrier) [36] 38
Abbildung 19: Eine Pipeline im Squidy-Tool 41
Abbildung 20: Links ist ein Auschnitt der ZOIL-Oberfläche zu sehen. Rechts sieht man die multimodale Interaktionsumgebung von Roomware. 43
Abbildung 21: Das linke Bild zeigt zwei Personen beim Arbeiten mit G-Nome Surfer. Auf dem rechten Bild sieht man eine Ansicht von Phylo-Genie. 44
Abbildung 22: Stimulus für das Auslesen von 2D-Koordinatensystemen. Die Punkte P1, P2 und P3 mussten abgelesen werden. 50
Abbildung 23: Stimuli für das Bestimmen der größten Fläche. Für beide Stimuli sollte die Farbe der größten Fläche bestimmt werden. 50
Abbildung 24: Stimuli für den Vergleich von Dreiecken. Für beide Dreiecke sollte geprüft werden ob sie geometrisch ähnlich sind. 51
Abbildung 25: Heatmap-Visualisierung eines Stimulus aus Aufgabe 1. Die Farbskala reicht von Rot nach Grün. Rote Bereiche wurden häufig fixiert, grüne Bereiche dagegen selten. 51
Abbildung 26: Scan-Path-Visualisierung eines Stimulus aus Aufgabe 3. Die Abfolge der Fixationen wird durch die blaue Linie dargestellt. 52
Abbildung 27: Parallel-Scan-Paths zweier Probanden für einen Stimulus mit drei AOIs oben: Gaze Duration Sequenz Diagramm mitte: Fixation Point Diagramm unten: Gaze Duration Distribution Diagramm 53
Abbildung 28: Zeigt einen mögliches Bild einer Eyetracking-Auswertung an Desktop-Monitoren. 54
Abbildung 29: Offline-Vorbereitung Der Moderator importiert Aufgaben in das System. 55
Abbildung 30: Vorbereitung der Auswertungssitzung Der Moderator fügt die Aufgaben zur Analyseumgebung hinzu. 55
Abbildung 31: Start der Sitzung Der Moderator erzeugt die erste Visualisierung. 56
Abbildung 32: Duplikation einer Visualisierung 56

Abbildung 33: Ändern des Visualisierungstyps Ein Nutzer wechselt die Visualisierung mit einer Wischbewegung. 57
Abbildung 34: AOIs erstellen 57
Abbildung 35: Verändern des Inhalts einer Visualisierung Ein Nutzer ändert die Reihenfolge der Achsen in einer Parallel-Scan-Path-Visualisierung. 58
Abbildung 36: Entfernen eines Probanden aus einer Visualisierung 58
Abbildung 37: Filtern der Daten Ein Nutzer teilt die Probanden auf zwei Visualisierungen auf. 59
Abbildung 38: Erzeugen eines Screenshots von einer Visualisierung 59
Abbildung 39: Wechseln der Aufgabe in den Visualisierungen 59
Abbildung 40: Offline Vorbereitung Der wissenschaftliche Mitarbeiter importiert Proteindaten ins System. 60
Abbildung 41: Vorbereitung der Präsentation Der wissenschaftliche Mitarbeiter fügt Proteine zur Analyseumgebung hinzu. 61
Abbildung 42: Start der Sitzung Der wissenschaftliche Mitarbeiter erzeugt die MegaMol-Visualisierung. 61
Abbildung 43: Duplizierung der Visualisierung 62
Abbildung 44: Änderung des Mappings der Visualisierung. 62
Abbildung 45: Ändern des Visualisierungstyps Der Vortragende wechselt die Visualisierung mit einer Wischbewegung. 63
Abbildung 46: Manipulation der 3D-Ansicht mit einer Pan-Geste links und einer Pinch-Geste rechts 63
Abbildung 47: Erzeugen eines Screenshots 64
Abbildung 48: Zeigt einige Ausrichtung anhand von Vektorfeldern a) Vektoren als Pfeildarstellung, b) Vektorrichtung als Farbe [49] 70
Abbildung 49: Skizze der Benutzeroberfläche 78
Abbildung 50: Skizze der Powerwallansicht mit drei Visualisierungscontainern 79
Abbildung 51: Visualisierungscontainer 1) Interaktionsmodus, 2) Löschenknopf, 3) Drag&Drop-Interaktionselement 80
Abbildung 52: Zeigt die beiden Modi des Knopf für den Interaktionsmodus. Beim Drücken wird zwischen diesen umgeschaltet. Der linke Teil zeigt den Knopf im Verschiebemodus, der rechte zeigt ihn im Bearbeitungsmodus. 81
Abbildung 53: Interaktionen mit dem Drag&Drop-Interaktionselement oben: Koppeln zweier Visualisierungscontainer unten: Duplizierung eines Visualisierungscontainers 81
Abbildung 54: Gesteninteraktion mit dem Visualisierungscontainer oben: Duplizieren mit Geste unten: Koppeln mit Geste 82
Abbildung 55: Filterkonzept mit Tangibles Die Zylinder sind die Tangibles. Das graue Tangible stellt den Filterbereich her. 84
Abbildung 56: Links: Studienszenario mit Position der Kamera Rechts: Sicht der Kamera zur Dokumentation der Studie 85
Abbildung 57: Ausgangssituation bei Aufgabe 1 87
Abbildung 58: Ausgangssituation von Aufgabe 2 Oben ist die Powerwall und unten der Tabletop zu sehen. 87
Abbildung 59: Ausgangssituation von Aufgabe 3, falls der Proband das Fenster kopiert. 88
Abbildung 60: Ausgangssituation von Aufgabe 3, falls der Proband das Fenster verschiebt. 88
Abbildung 61: Tabletop mit Powerwallansicht und einem Fenster auf der Powerwall 89
Abbildung 62: Fenster mit Miniaturansicht(Drag&Drop-Interaktionselement) auf dem Tabletop 90
Abbildung 63: Fenster mit Löschenknopf 90
Abbildung 64: Zwei Fenster mit Drag&Drop-Interaktionselementen 91
Abbildung 65: Gekoppelte Fenster mit Linie und Knöpfen 91

Abbildung 66: Id-Karte erzeugt private Ablage für Fenster(rechts) 92
Abbildung 67: Kreis als Filterbereich 92
Abbildung 68: Fenster mit den beiden Modi des Knopfs 93
Abbildung 69: Entkoppeln zweier Fenster durch Durchstreichen der Linie 94
Abbildung 70: Schieben eines Fensters auf die Powerwallansicht 95
Abbildung 71: Erklärung von Koppeln mit Konnektorpunkten 97
Abbildung 72: Proband beim Ausprobieren des vorgeschlagenen Filterkonzepts 99
Abbildung 73: Die Probandin schiebt ein Fenster vom Tisch 100
Abbildung 74: Das Koppeln mit einem Tangiblepaar wird angedeutet 101
Abbildung 75: Änderung beim Verschieben auf die Powerwallansicht 106
Abbildung 76: Architektur des Systems 110
Abbildung 77: Komponentenstruktur des Systems und deren Zusammenhänge 111
Abbildung 78: Klassenstruktur der Abstraction-Komponente 112
Abbildung 79: Klassenstruktur des Servers 113
Abbildung 80: Klassenstruktur der DataProvider-Komponente 115
Abbildung 81: Klassenstruktur des VisualisierungsClient-Backends 117
Abbildung 82: Klassenstruktur des Plugin-Systems für Visualisierungen 119
Abbildung 83: Ablauf beim Registrieren eines Visualisierungsclients 121
Abbildung 84: Ablauf beim Verteilen von Nachrichten im System 122
Abbildung 85: Zustände des message brokers beim Verteilen der Nachrichten 123
Abbildung 86: Ablauf beim Nachladen von Daten aus einem DataProvider 123
Abbildung 87: Eyetracking Datenmodell für den Daten-Provider des Prototyps Die Pfeile zwischen den Klassen stellen 1:N-Beziehungen dar. 127
Abbildung 88: Visualisierungen für Eyetracking-Daten 1) Stimulus-Visualisierung, 2) Heatmap-Visualisierung, 3) Gaze-Duration-Sequenz-Visualisierung 129
Abbildung 89: Der Microsoft Pixelsense des VIS 129
Abbildung 90: Benutzeroberfläche des Clients 1) privater Visualisierungscontainer, 2) Powerwallansicht 3) Datenexplorer, 4) Administrationsdialog 131
Abbildung 91: Powerwallansicht 1) öffentlicher Visualisierungscontainer, 2) Minimap 132
Abbildung 92: Datenexplorer mit Stimuli 133
Abbildung 93: `AbstractVisWindow` Stellt die grundlegenden Funktionen eines Visualisierungscontainer bereit. 133
Abbildung 94: Privater Visualisierungscontainer (`SurfaceVisWindow`) 1: Duplizierenknopf, 2: Koppelnknopf, 3: Interaktionsmodus, 4: Löschenknopf, 5: Visualisierung 134
Abbildung 95: Öffentlicher Visualisierungscontainer(`PowerwallVisWindow`) mit ausgeblendeten Steuerelementen 134
Abbildung 96: Tangible mit Byte-Tag 136
Abbildung 97: Verschiedene Tangibles des Client 1) Datenrepräsentationstangibles, 2) Kopplungstangiblepaar, 3) Duplizierungstangible 136
Abbildung 98: Kopplung von zwei Visualisierungscontainer mit Tangibles Die Kreisanimation zeigt den Fortschritt an. 137
Abbildung 99: `LocalDataManager`-Klasse 138
Abbildung 100: Aufbau für den Powerwall-Client 139
Abbildung 101: Eine Auswahl an SLAP Widgets a) Radiobuttons, b) Drehknopf, c) Regler, d) Tastatur 146

Tabellenverzeichnis

Tabelle 1: Teilnehmer der Studie 86
Tabelle 2: Vorschläge bezüglich der Powerwallsteuerung (X: der Proband erwähnt die Funktionalität) 103
Tabelle 3: Vorschläge für Gesten (-- : keine Aussage oder wurde nicht gefragt) 104
Tabelle 4: Vorschläge für Tangibleinteraktion (-- : keine Aussage) 105

1 Einleitung

1.1 Motivation

Wissenschaft und Industrie erzeugen heutzutage eine unglaubliche Menge an Daten. Laut der „Digital Universe“ Studie des IT-Dienstleisters EMC wurden für das Jahr 2011 1.8 Zetabyte digitale Daten vorhergesagt [1]. Das entspricht 1.8 Milliarden 1TB-Festplatten. Davon sind große Mengen privater Natur, in Form von Videos, Bilder, Zeitschriften und Büchern. Der andere Teil sind jedoch Daten mit denen ein Mensch zuerst nichts anfangen kann. Dazu gehören die meisten Daten aus der Wissenschaft, Medizin und statistische Daten von Unternehmen. Um diese Daten sinnvoll anzuzeigen und verstehen zu können, müssen Visualisierungen genutzt werden. Darum betreiben Universitäten und Unternehmen großen Aufwand neue Visualisierungen zu entwickeln, um solche Daten besser zu verstehen. [2]

Die entwickelten Visualisierungen können jedoch bei großen Datenmengen oft nicht sinnvoll auf normalen Displays angezeigt werden. Beispiele dafür sind Daten aus der Biologie [3] oder Astrophysik [4]. Des Weiteren müssen z.B. bei statistischen Daten Charakteristiken mehrerer Datenmengen miteinander verglichen werden. Dies kann zusätzlich zu Platzproblemen auf herkömmliche Desktop-Displays führen. Zudem spielt Teamwork eine immer größere Rolle und unter dem Grundsatz „Vier Augen sehen mehr als zwei“ wird auch die Analyse von Daten in Gruppen durchgeführt [5]. Diese Gruppen benötigen großflächige Displays, um sich im Team über die visualisierten Daten auszutauschen. Darum werden inzwischen für solche Szenarien große, hochauflösende Displays entwickelt [6]. Am Ende einer Analyse können die Displays zur Präsentation der Ergebnisse vor einem Plenum genutzt werden.

Die Bedienung solcher Displays bzw. die Interaktion mit ihnen ist jedoch aufwendig. Normalerweise werden Desktop-PCs genutzt deren Displays erweitert werden. Jedoch eignet sich die Bedienung mit Maus nicht für größere Displays. In Besprechungsräumen finden sich jedoch oft auch Tische, um Unterlagen oder Laptops abzulegen. Statt Tischen könnten Tabletop-PCs eingesetzt werden. Diese interaktiven, horizontalen Displays sind speziell für kollaborative Arbeit an Computern ausgelegt. Universitäten und wissenschaftliche Einrichtungen experimentieren schon länger mit dieser Art von Computer [7]. In der Industrie nimmt Microsoft hier die Vorreiterrolle ein. In ihrem Microsoft Pixelsense Projekt(früher Microsoft Surface) wurden zwei Generationen von Tabletop-PCs entwickelt [8]. Auf Basis der Pixelsense-Tabletops wurden schon Anwendungen implementiert. Kommerziell werden sie in Casinos [9] und Läden eines Mobilfunkanbieters [10] in den USA eingesetzt. Auch in der Wissenschaft sind schon Interaktionskonzepte für den Pixelsense-Tabletop oder ähnliche Tabletops entworfen worden.

In Film und Fernsehen werden Powerwalls und Tabletops und ähnliche Geräte oft als Interaktionsmittel der Zukunft dargestellt. Ein Beispiel findet sich in der Krimiserie CSI Miami, in der ein Tabletop in Kombination mit einem großen Display genutzt wird. Das geht auch in die Richtung der Vision von Mark Weiser für den Computer des 21. Jahrhunderts [11]. Weiser sieht den Computer in seiner Vision als unsichtbare Unterstützung für anfallende Arbeit, wobei die Computer im Prinzip mit der Arbeitsumgebung verschmelzen.

In Anlehnung an diese Vision, soll in dieser Arbeit der Tabletop-PC in einer Analyseumgebung mit einer Powerwall genutzt werden. Zugrunde liegt, als tatsächliches Szenario, die Eyetracking-Analyse. Aktuelle Forschungsarbeiten in dieser Richtung beschränken sich meist auf reine Fernsteuerung von großen Displays oder reine Mehrbenutzeranalyse auf dem Tabletop. Diese Arbeit versucht beides zu verbinden.

1.2 Aufbau

Kapitel 1 ist die Einleitung und Übersicht.

In Kapitel 2 führt in die Grundlagen des „Visual Analytics“ und die Visualisierung ein, um einen groben Überblick über die Prinzipien von visueller Analyse zu geben. Im Weiteren werden einige Interaktionsparadigmen vorgestellt, die sich inzwischen durchgesetzt haben oder Themen der Forschung sind, darunter Natural User Interfaces(NUI) und Tangible User Interfaces(TUI). Danach wird das Prinzip der Tabletop-PCs und verschiedene existierende Hardware dazu vorgestellt. Zum Schluss wird in das Thema der großflächigen, hochauflösenden Displays eingeführt.

Das Kapitel 3 stellt verwandte Arbeiten im Bereich der multimodalen Interaktionssysteme und Anwendungen auf Tabletops vor.

Kapitel 4 führt in die Aufgabenstellung ein und erklärt die Vorgehensweise der Arbeit.

Das 5. Kapitel beschreibt die Ergebnisse der durchgeführten Anforderungsanalyse auf Basis einer im Vorfeld vom Institut durchgeführten Eyetracking-Studie. Bei der Ergebnisanalyse der Eyetracking-Studie wurde versucht eine Powerwall und verschiedene Software-Tools zu verwenden. Diese Analysesitzung der Studie wurde beobachtet und anschließend mit den Analysten diskutiert. Auf Grundlage dieser Beobachtungen wurde daraufhin ein Szenario für eine Analyseumgebung, zur Analyse von Eyetracking-Studien, entwickelt und mit den Analysten verfeinert. Die Analyseumgebung soll dabei durch einen Tabletop-PC gesteuert werden und Visualisierungen auf einer Powerwall anzeigen können. Zusätzlich wurde noch ein Szenario für eine Analyseumgebung für wissenschaftliche Visualisierungen speziell für Moleküle [3] erstellt. Mit Hilfe der Beobachtungen und Szenarien werden die Anforderungen an die Analyseumgebung und die Tabletop-Steuerung identifiziert.

Im 6. Kapitel wird anhand der Anforderungen ein abstraktes Konzept für Analyseumgebungen mit Powerwalls und Tabletop-Steuerung entworfen. Das Konzept beschreibt dabei den grundlegenden Aufbau des Systems der Analyseumgebung. Außerdem erläutert es den Einsatz eines Tabletops innerhalb des Systems.

Kapitel 7 beinhaltet den Lösungsansatz. Er baut auf dem Konzept auf und konkretisiert es mit Interaktionen und Oberflächenentwurf. Zuerst werden einige Entwurfsvorüberlegungen vorgestellt. Mit den Vorüberlegungen wurde eine Vorstudie entworfen, bei der das Lösungskonzept mit Probanden anhand von Paperprototyping erprobt werden soll. In einem zweiten Teil wird der Lösungsansatz anhand der Ergebnisse der Vorstudie erweitert.

In Kapitel 8 werden die Architektur und ausgewählte Implementierungsdetails des Kommunikationsframeworks beschrieben, das für die multimodale Analyseumgebung entwickelt wurde. Dabei wird gezeigt, wie das im Konzept entworfene System umgesetzt wurde. Der Prototyp für die Tabletop-Anwendung zur Steuerung von Powerwall-Visualisierungen basiert auf diesem Framework.

Kapitel 9 beschreibt die Implementierung des Prototyps der Tabletop-Anwendung zur Steuerung von Powerwall-Visualisierungen. Dabei wird beschrieben welche Teile des Lösungsansatzes im Prototyp umgesetzt wurden.

Zum Schluss werden in Kapitel 10 die Ergebnisse der Arbeit zusammengefasst. Zusätzlich wird ein Ausblick auf mögliche Weiterentwicklungen des Konzepts und der Implementierung gegeben.

2 Grundlagen

Das Grundlagenkapitel gibt einen Überblick über die Themen, die in dieser Arbeit angesprochen werden.

2.1 Visual Analytics und Visualisierung

Dieses Kapitel gibt eine kurze Einführung in Visual Analytics und geht innerhalb von diesen auf Visualisierung etwas näher ein. Visual Analytics wird angesprochen, da es unter anderem Visualisierung und Interaktion miteinander vereinigt. Außerdem werden Daten exploriert, was eine Grundlage dieser Arbeit darstellt.

2.1.1 Visual Analytics

Visual Analytics ist eine interdisziplinäre Fachrichtung, um mit großen und komplizierten Daten umzugehen. Es beinhaltet mehrere Prozesse und hat eine große Bandbreite an Anwendungsgebieten. Die Grundidee bei Visual Analytics ist es, die Daten visuell zu repräsentieren und damit zu interagieren. Dadurch wird dem Benutzer ein Einblick gewährt, um schlussendlich Aussagen zu treffen und bessere Entscheidungen zu fällen. Da eine fundierte Entscheidungsfindung unweigerlich menschliche Einwirkung bedingt, um Flexibilität und Kreativität zu gewährleisten. Deshalb müssen die Daten von den Involvierten überschaut und verstanden werden können. Visual Analytics soll es den Analysten ermöglichen, sich auf die wesentliche Entscheidungsfindung zu konzentrieren und dabei leistungsstarke Rechenfähigkeiten und interaktive Visualisierung bereitzustellen. [12]

Eine Definition für Visual Analytics ist:
„…the science of analytical reasoning facilitated by interactive visual interfaces" (…die Wissenschaft der analytischen Beweisführung unterstützt durch interaktive visuelle Schnittstellen) [13].

Daher verbindet Visual Analytics die wissenschaftliche Analyse mit der Visualisierung. Das beinhaltet eine Vorverarbeitung der Daten, die anschließende Visualisierung und Interaktion und zum Schluss die Entscheidungsfindung anhand der gewonnenen Erkenntnisse. Um mit den großen komplexen und uneinheitlichen Daten zurechtzukommen, verbindet Visual Analytics die Fähigkeiten von Menschen mit denen von Computern. Dies geschieht durch die Verbindung von automatischen statistischen und mathematischen Methoden mit interaktiven Visualisierungen, die dem Menschen erlauben, Umstände zu erkennen und Schlüsse zu ziehen.

Eine etwas modernere Definition ist daher:
„Visual analytics combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex datasets." (Visual Analytics kombiniert automatische Analysetechniken mit interaktiven Visualisierungen, um ein effektives Verständnis, eine effektive Schlussfolgerung und Entscheidungsfindung auf der Basis von sehr großen und komplexen Datensätzen zu erlangen.) [14].

Laut Keim et al. [14] sind die zentralen Ziele von Visual Analytics folgende:

- Informationen über massive, dynamische, mehrdeutige und oft widersprechende Daten aufzubauen und einen Einblick in diese gewähren
- Erwartete Tatsachen zu entdecken

- Unerwartete Umstände aufzudecken
- Zeitnahe, vertretbare und verständliche Beurteilungen zu liefern
- Handlungsoptionen anzubieten

Entwickelt hat sich die Visual Analytics aus der klassischen Visualisierung. Allerdings ist Visual Analytics mehr als Visualisierung. Wie schon angedeutet, wird versucht den Menschen in den Analyseprozess miteinzubinden und mit automatischen Datenanalyse zu unterstützen. Der Teil der automatischen Datenanalyse spielt in dieser Arbeit allerdings keine Rolle, daher wird darauf nicht weiter eingegangen. Jedoch wird die Anwendung solcher Verfahren auch nicht ausgeschlossen.

2.1.2 Visualisierung

Die Visualisierung besteht grundsätzlich aus zwei Gebieten [15] [12]:

Die Erstere, ist die *wissenschaftliche Visualisierung*. Sie beschäftigt sich, wie der Name schon sagt, mit wissenschaftlichen Daten, die über Sensoren, Simulationen und Laborversuche gewonnen werden. Diese Daten spannen oft zwei, drei oder mehr Dimensionen auf und bestehen aus enormen Datensätzen. Um solche große und komplexe Datenmengen überhaupt zu verstehen, benötigt man Darstellungsformen, die es erlauben die Daten in ihrer Bedeutung so detailgetreu wie möglich zu erfassen. Die wichtigsten wissenschaftlichen Visualisierungen sind Strömungsvisualisierung und Volumenrendering. Diese haben wiederum viele Anwendungen zum Beispiel in der Medizin, der Geographie, der Meteorologie, der Weltraumforschung und in industriellen Anwendungen. Formal kann man sagen, dass die wissenschaftliche Visualisierung hauptsächlich bei kontinuierlichen Daten angewandt wird.

Die Zweite, ist die *Informationsvisualisierung*, sie beschäftigt sich mit diskreten Daten. Sie versucht im Gegensatz zur wissenschaftlichen Visualisierung abstrakte Daten darzustellen, die auch teilweise keinen Bezug zur physischen Welt haben. Beispiele sind virtuelle Daten wie Aktien. Bei abstrakten Daten fehlt die Anlehnung an die reale Welt, vor allem räumliche Informationen, daher müssen intuitive Wege gefunden werden die Daten verständlich zu visualisieren.

Die Ziele der Visualisierung sind im Allgemeinen [16]:

- **Explorative Analyse**
  Es existieren noch keine Hypothesen sondern nur die reinen Daten. Dabei wird durch Interaktion mit der Visualisierung nach Mustern oder Begebenheiten gesucht. Diese Informationen werden verwendet um eine Hypothese zu finden.
- **Konfirmative Analyse**
  Hier existiert eine Hypothese bereits. Daher wird mit der Visualisierung zielgerichtet versucht die Hypothese zu überprüfen. Hierbei kann das Ergebnis sein, dass die Hypothese bestätigt oder verworfen wird.
- **Präsentation**
  Die Aussagen und die Hypothesen liegen vor und sind bereits bestätigt. Die Visualisierung soll hier die Fakten klar darstellen, so dass Außenstehende sie erkennen können.

Um die verschiedenen Ziele einer Visualisierung zu erfüllen, muss diese einigen Qualitätsanforderungen gerecht werden [16]:

- **Expressivität**
  Die Datenmenge sollte möglichst unverfälscht dargestellt werden. Es soll kein falscher Eindruck entstehen.
- **Effektivität**
  Bei verschiedenen Möglichkeiten die Datenmenge darzustellen, unter Einhaltung des Expressivitätskriterium, sollte die Wahl auf die Visualisierung fallen, die der Zielgruppe und der Problemstellung am besten gerecht wird.
- **Angemessenheit**
  Nicht nur die Technologiekosten sondern auch die Belastung des Benutzers muss beachtet werden. Der Aufwand sollte dem Nutzen gerecht werden.

Der Prozess der aus Daten eine Visualisierung erstellt wird meist als eine Pipeline angegeben, der sogenannten *Visualisierungspipeline.* [16]

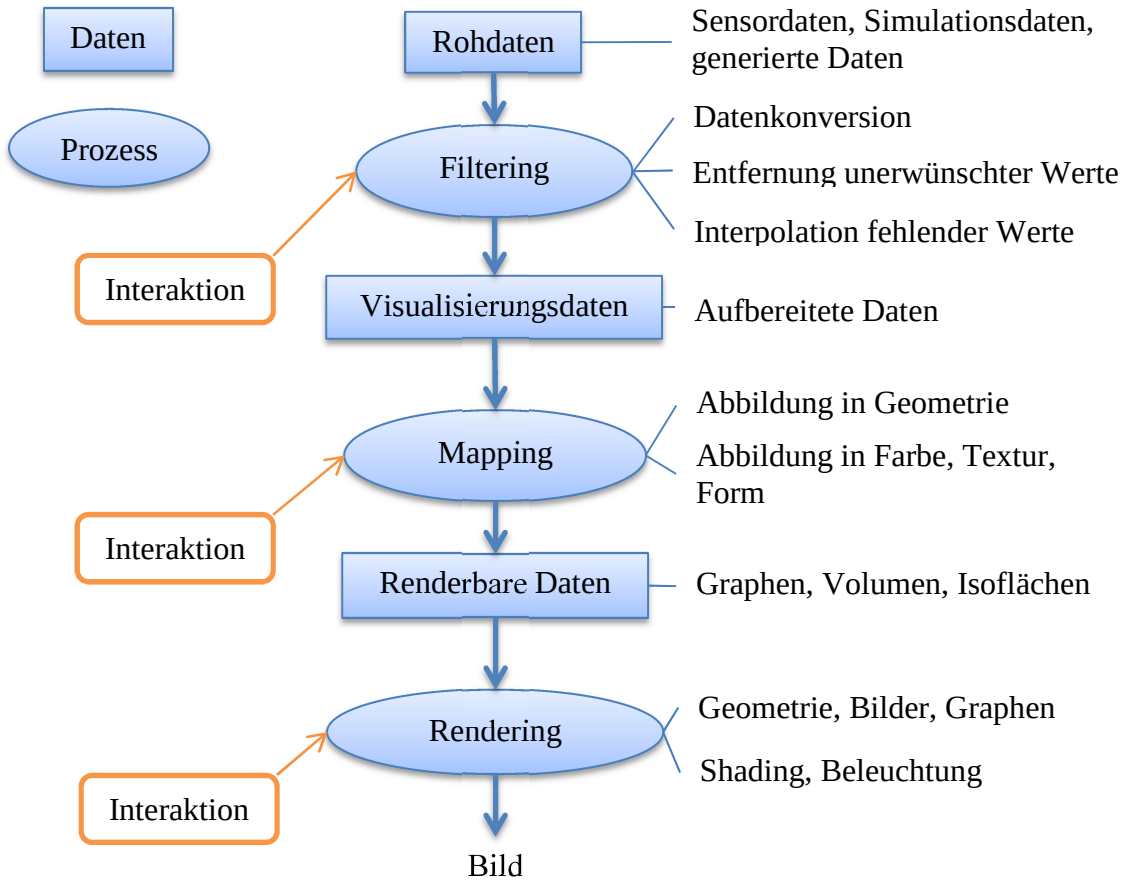


Abbildung 1: Visualisierungspipeline [16]

Die Visualisierungspipeline transformiert die Daten zu einem Bild. Die Rohdaten können aus der realen Welt stammen und werden dann meist von Sensoren gewonnen. Im Gegensatz dazu gibt es Simulationsdaten die auf mathematischen Modellen basieren. Diese bilden zwar meist die reale Welt ab beziehen sich jedoch nicht auf reale Ereignisse. Eine weitere Datenquelle sind Daten aus der digitalen Welt, dabei kann es sich um statistische Daten aus Datenbanken oder Softwaresystemen handeln, aber auch realitätsnahe Daten aus sozialen Netzwerken [17]. Eyetracking für Bedienungsoberflächen kann hierbei als Hybridform zwischen Daten aus Sensoren und Daten aus der digitalen Welt angesehen werden.

Im Schritt *Filtering* werden die Daten vorverarbeitet. Das kann wie der Name schon sagt bedeuten, dass Daten entfernt werden. Diese können entweder Werte sein, die durch schlechte Umgebungsumstände verfälscht wurden oder Daten die für die weitere Untersuchung irrelevant sind. Außerdem können Fehler auch korrigiert oder Werte geglättet werden. Weiterhin werden die Daten bei diesem Schritt auch aufbereitet. Das heißt, es werden bei kontinuierlichen Daten fehlende Werte interpoliert. Das Errechnen von markanten Stellen in den Daten findet ebenfalls in der Aufbereitungsphase statt. Das können zum Beispiel mathematische Extrema bei kontinuierlichen Daten sein oder Cluster bei diskreten Daten. In diesem Schritt kann der Benutzer durch Interaktion die angewendeten Operationen parametrisieren. Wichtig ist vor allem die Möglichkeit, die Daten nach gewissen Merkmalen einzuschränken.

Das *Mapping* ist der entscheidende Schritt, bei dem die Visualisierung größtenteils entsteht. Bei diesem Schritt werden die Visualisierungsdaten in visuell darstellbare Form gebracht. Das sind meist geometrische Daten und ihre Attribute wie Farbe oder Textur. Die interaktive Anpassung dieser Stufe erlaubt es, das Aussehen der Visualisierung grundlegend zu ändern.

Am Ende der Pipeline steht das *Rendering*. Hier entsteht aus den anzeigbaren Daten das eigentliche Bild. Der Aufwand kann sehr unterschiedlich sein. So kann es zum Beispiel ein tatsächliches Rendering von Dreiecken mit Shading und weiteren Effekten sein oder nur das Plotten eines 2D-Graphen. Auch hier ist es dem Benutzer möglich den Prozess anzupassen.

Die verschiedenen Stufen mit Interaktion plus die Datenakquise erlauben verschiedene Szenarien der interaktiven Visualisierung. Der Visualisierungsprozess kann keinerlei Interaktion erlauben wodurch am Ende ein statisches Video oder Bild entsteht. Auf der anderen Seite kann vollständige Interaktion über alle Schritte möglich sein und sogar die Datenakquise zum Beispiel bei einer Simulation parametrisiert werden.

Die allgemeine Vorgehensweise von interaktiver Visualisierung bei Exploration von Daten wird von Shneiderman folgendermaßen zusammengefasst: *„Overview first, zoom and filter, details on demand“* [18]. Daher sollte die Visualisierung als erstes eine gute Übersicht liefern. Die Interaktion sollte Zoomen und Filtern der Daten ermöglichen und ein optionales Anzeigen der Details. „Visual Analytics“ erweitert diese Idee noch.

2.1.3 „Visual Analytics“-Prozess

Der „Visual Analytics“-Prozess kombiniert visuelle mit automatischen Analysemethoden. Abbildung 2 zeigt eine grobe Übersicht.

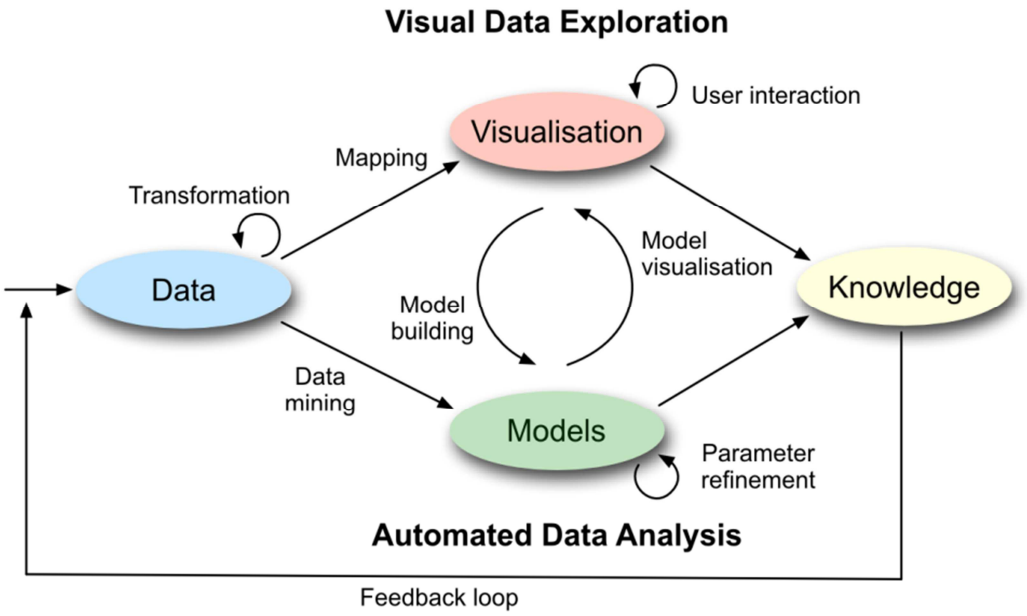


Abbildung 2: Visual Analytics Prozess [14]

Wie auch bei der Visualisierungspipeline im Filtering-Schritt vorgesehen, müssen die Daten oft transformiert werden bevor sie für visuelle oder andere Analyse verwendet werden können. Dies geschieht im „Visual Analytics“-Prozess als erstes.

Nachdem die Daten vorbereitet wurden, muss beim „Visual Analytics“-Prozess entschieden werden ob eine automatische Exploration oder eine manuelle stattfinden soll.

Im Falle einer automatischen Analyse werden Methoden aus dem Datamining angewendet, um geeignete Modelle für die Daten zu finden. Die Modelle können dann selbst visualisiert werden, um für bessere Ergebnisse angepasst und verfeinert zu werden. Dabei werden die Modelle meist auf die Daten angewandt und visualisiert, so dass der Analyst ein besseres Verständnis für die Modelle bekommt und sie auch gleich bewerten kann. In diesem Sinne werden die Daten durch abwechselnde Anwendung von visuellen und automatischen Methoden exploriert. So können auch irreführende Ergebnisse früh erkannt werden. Im günstigsten Fall erhält der Analyst zum Schluss der Analyse die Ergebnisse der Analyse selbst und ein Modell, um solche Daten zu analysieren.

Wenn der Analyst mit einer visuellen Analyse beginnt, wird er zuerst eine Hypothese erstellen, die er später mit automatischen Methoden auf dem gesamten Datensatz bestätigen kann. Hierfür werden Hilfsmittel benötigt, um die Daten interaktiv zu explorieren. Der Analyst sollte dabei zumindest in der Lage sein die Daten zu filtern und innerhalb dieser zu zoomen. Während der Exploration kann der Analyst ein Modell für die automatische Analyse entwickeln.

Das „visual seeking mantra“ von Shneiderman kann somit folgendermaßen erweitert werden. Bei sehr großen Datenmengen ist es schwierig bis unmöglich einen Überblick der Daten anzuzeigen. Das führt dazu, dass der Analyst nicht weiß welche Daten er filtern soll oder auf welche er zoomen könnte. Darum wurde folgende Änderung vorgeschlagen „Analyse first, show the important, zoom/filter, analyse further, details on demand“ [12]. Das bedeutet, zuerst müssen die Daten vorverarbeitet werden, dann erhält der Analyst einen Überblick mit den wichtigsten Aspekten bevor er weiterarbeitet. Danach kann der Analyst die wichtigen Daten entsprechend der oben genannten Vorgehensweise weiter explorieren.

2.2 Interaktion

Das folgende Kapitel gibt einen kurzen Einblick in die verschiedenen Entwicklungsabschnitte der Benutzerschnittstellen und Computerinteraktion. Als erstes wird ein Überblick über frühere Prinzipien der Benutzerschnittstellen und deren Entwicklung zu heutigen Prinzipien gegeben. Danach folgt ein Abschnitt über die momentan aktuellen Benutzerschnittstellen, die dem sogenannten WIMP-Paradigma folgen. Im Anschluss werden noch zwei sogenannte post-WIMP Prinzipien angesprochen.

2.2.1 Entwicklung der Benutzerschnittstellen und der Computerinteraktion

Die Entwicklung der Mensch-Computer-Interaktion ist ausführlich in dem Buch „Mensch-Computer-Interface: Zur Geschichte und Zukunft der Computerbedienung“ [19] beschrieben. Einige Abschnitte der Entwicklung werden kurz beschrieben.

Der Anfang des Computerbooms wurde von großen Mainframes dominiert. Die Größe der Hardware und der Preis für die Herstellung von Computern waren so enorm, dass eine Nutzung von persönlichen Geräten undenkbar war. Anfangs wurden die Eingabe der Geräte als bloße Einfuhr und Abfuhr von Daten und den zugehörigen Programmcode gesehen. Dies resultierte direkt aus der Hardwarearchitektur von Neumanns, die eine Eingabe- und Ausgabeschnittstelle besitzt und ein Speicher für den Instruktionscode.

Schon zu Anfang der Computerentwicklung hatte Vannevar Bush in seiner Arbeit „As We May Think“ [20] die Vision einer Maschine (siehe Abbildung 3), die einen Wissenschaftler bei der Wissensfindung und Verwaltung seiner Dokumente unterstützt. Die Maschine hieß Memex (Memory Extender) und kann als eine der ersten Visionen der Personal Computer (PC) gesehen werden.

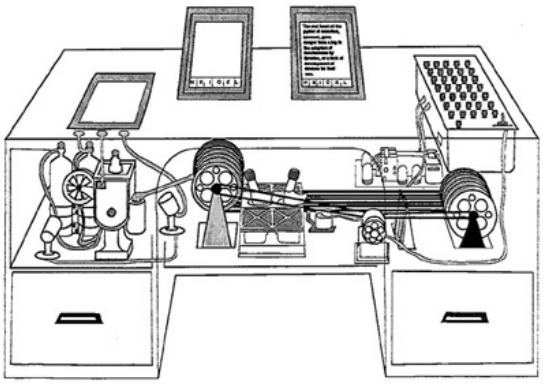


Abbildung 3: Skizze des Memex [20]

Im späteren Verlauf dieser Mainframe-Ära entwickelten sich Terminal-Systeme. Diese Terminals dienten der Eingabe an den Mainframes. Doch ersetzten sie nicht komplett die Lochkarten, mit denen die Eingabe an den Großrechnern gemacht wurde. Die Rechenzeit der Mainframes war nämlich teuer und die Nachfrage groß. Daher gab es einen Operator, der den Mainframe bediente und andere Nutzer mussten diesem per Lochkarten die Eingabe und den Programmcode geben. Nachdem der Ablauf des Programms beendet war, erhielten die Nutzer die Ausgaben des Programms wieder. Gleichzeitig entwickelten sich verteilte Terminals, die es dem Benutzer erlaubten einen Teil der Rechenzeit eines Großrechners zu nutzen. Aber auch hier war die Rechenzeit an dem Großrechner sehr teuer. In diesem Sinne entwickelten sich kleinere Computer, die ähnlich wie die Großrechner bedient werden konnten aber über weit weniger Rechenzeit verfügten. Alle diese Terminals hatten die später als Character User Interfaces (CUI) klassifizierten Benutzerschnittstellen. Mit diesen konnte ein Kommando über

eine Tastatur, die ähnlich wie Schreibmaschinen ein QWERTY-Layout hatten, eingegeben werden. Diese Kommandos werden ausgewertet und über ein Textdisplay ausgegeben.

Ein großer Durchbruch in Richtung der grafischen Benutzerschnittstellen gelang Ivan E. Sutherland. Er führte das Zeichnen und Zeigen in die Benutzerschnittstelle von Computern mit ein. Das Ergebnis seiner Arbeit ist das Sketchpad [21]. Es nutzte einen sogenannten Lichtgriffel um Linien auf einen damals sehr neuen vektorbasierten Röhrenbildschirm zu zeichnen.

Abbildung 4: Sutherland bei der Arbeit mit dem Lichtgriffel

Aus der Arbeit von Sutherland entwickelten sich daraus folgend Zeigegeräte. Das wichtigste unter diesen ist die Maus. Sie ermöglicht es recht genau Punkte auf dem Bildschirm indirekt anzufahren. Mit der Entwicklung der Maus, der Entwicklung von Rastergrafikdisplays und einer stetig steigenden Rechenleistung der Computer entstanden allmählich, die später als Graphical User Interface (GUI) bezeichneten, grafischen Benutzerschnittstellen. Die ersten kommerziellen Systeme wurden von Xerox PARC eingeführt und der kommerzielle Erfolg gelang Apple. Bei den grafischen Benutzeroberflächen haben sich mit der Zeit vier Konzepte durchgesetzt. Die Eingabe des Systems wird durch den Zeiger (Pointer) der Maus realisiert. Die Benutzeroberfläche wird durch Fenster (Windows) strukturiert. Es gibt Bilder (Icons), die als Stellvertreter von Daten angezeigt werden. Die Auswahl einer Funktion ergibt sich durch Menüs (Menus). Zusammengefasst wird das im WIMP-Prinzip. Die WIMP-Benutzerschnittstellen haben sich inzwischen in den wichtigsten Betriebssystemen wie Windows und Mac OS X vollständig durchgesetzt. Sie sind daher fester Bestandteil heutiger Desktop-PCs.

Nebenbei entstand die Vision von Mark Weiser. In seiner Arbeit „The Computer for the 21st Century“ [11] führt er in das „ubiquitous computing“ ein. Er argumentiert, dass Computer unsichtbar sein sollten und Interaktionen mit ihnen natürlich aus der momentanen Beschäftigung heraus geschehen sollten. Er beschreibt die nötigen Computer als günstige, energiesparende Hardware, die vernetzt werden kann und praktische Displays besitzen. Tatsächlich sind mit dem heutigen Stand der Technik einige seiner Vorschläge umsetzbar.

Mit dem Aufkommen von berührungsempfindlichen Smartphones und Tablet-PCs wurden neue Arten von Benutzeroberflächen nötig. Mit diesen sollen alltägliche Arbeiten auf den Geräten intuitiv und von unerfahrenen Benutzern durchgeführt werden können. Diese Benutzerschnittstellen werden als Natural User Interface (NUI) bezeichnet. Die ersten kommerziellen Erfolge wurden von Apple mit dem IPhone und dem IPad gemacht. Aber auch Google und Microsoft versuchen mit ihren Systemen Apple zu folgen.

In der Wissenschaft und der Fiktion, z.B in dem Film Minority Report, tut sich eine weitere neue Art von Benutzerinterface auf. Diese Benutzerinterfaces entfernen sich von der Interaktion mit virtuellen Objekten wie es bei WIMP-Interfaces und NUIs der Fall ist. Hier werden Objekte aus der realen Welt verwendet, um mit den Computern zu interagieren. Theoretisch werden diese Benutzerschnittstellen schon einige Zeit behandelt, wie z.B. in der Marbel Answering Machine von Durrel Bishop. Zum Einsatz kommen sie heutzutage in den Tabletop-PCs, die solche Objekte beim Stellen auf die Oberfläche erkennen. Diese Benutzerschnittstellen nennt man Tangible User Interface(TUI).

Abbildung 5: Ein Beispiel eines Tangible User Interface, die Marbel Answering Machine

NUIs und TUIs werden oft als post-WIMP bezeichnet, da sie neue Interaktionstechniken nutzen [22].

2.2.2 Windows, Icons, Menus and Pointers (WIMP)

Die WIMP-Benutzerschnittstellen sind vorherrschend bei heutigen Desktop-PCs. Ihr Prinzip wird auch „Direct Manipulation“ genannt [22]. Die Benutzerschnittstellen werden wie ihr Name sagt von den Komponenten Windows, Icons und Menus beherrscht. Ihre Eingabe ist der Mauszeiger (Pointer), daher auch die Bezeichnung Direct Manipulation.

- Windows (Fenster)
  Die Fenster der Benutzerschnittstelle organisieren die angezeigte Oberfläche. Ihr Inhalt ist wie die Symbolik des Namens ausdrückt ein Fenster auf eine bestimmte Anwendung oder Daten. Sie können sich meist überlagern und die meisten können frei verschoben und skaliert werden.
- Icons
  Die Icons sind grafische Darstellungen statt Text. Sie sind meist eine Repräsentation von Daten oder Objekten. Manchmal sind sie mit Beschreibungstexten versehen. Sie sind meist die Grundlage für Drag&Drop-Interaktionen in WIMP-Benutzeroberflächen.
- Menus(Menüs)
  Die Menüs sind meist aufklappbare Auswahlmenüs. Sie repräsentieren meist Funktionen und führen diese beim Auswählen aus.

- Pointer(Mauszeiger)
  Der Mauszeiger ist das primäre Interaktionsmittel der Benutzerschnittstelle. Er wird als Bild dargestellt und kann mit der Maus frei bewegt werden. Je nach Funktion die der Zeiger hat oder dem Systemstatus, kann er unterschiedlich aussehen.

Viele der heutigen Anforderungen an die Benutzbarkeit einer Benutzerschnittstelle haben sich aus den WIMP-Schnittstellen entwickelt. Ein Beispiel ist Fitts's Law welches die benötigte Zeit für eine Interaktion auf Basis der Geschwindigkeit des Zeigers, dem Abstand zum Ziel und die Größe des Ziels abschätzt.

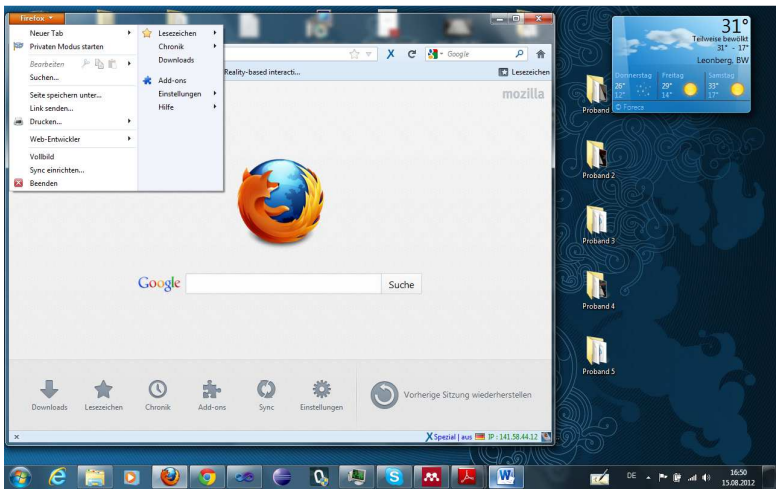


Abbildung 6: Windows 7 hat eine typische WIMP-Benutzerschnittstelle

2.2.3 Natural User Interfaces

Die Natural User Interfaces (NUIs) sollen es dem Benutzer ermöglichen Computer jeglicher Art intuitiver zu bedienen. Das trifft vor allem auf Geräte zu, die im täglichen Leben verwendet werden. Solche Geräte sollten schnell zu erlernen sein und ohne großes Nachdenken eingesetzt werden können. Solche Geräte werden meist sehr häufig genutzt und hin und wieder auch während eine andere Tätigkeit ausgeführt wird. Ein Beispiel sind berührungsempfindliche Smartphones, diese sollen mindestens genauso schnell und einfach bedient werden können wie es bei klassischen Mobiltelefonen der Fall ist. Weiterhin bieten sie dem Anwender viele neue Möglichkeiten. Diese Möglichkeiten werden aber nur genutzt wenn die Bedienung dort ebenfalls einfach und intuitiv ist.

In dem Buch „Brave NUI World" [23] wird die Imitation der Realität nur als ein Teil des Wortes *natürlich* gesehen. Hier wird der Begriff *natürlich* als Grundlage des Designs von Benutzerschnittstellen gesehen. Der Benutzer soll bei diesen Benutzerschnittstellen das Gefühl haben sie natürlich zu bedienen. Experten sollten das Gerät als Erweiterung ihres eigenen Körpers sehen können und Laien sollten das Gefühl haben selbst Experten zu sein. Das bedeutet, ein Experte sollte nicht durch aufdringliche Hilfen oder enge Benutzerführung gestört werden. Der Laie sollte aber soweit unterstützt werden, dass er das Gerät ohne Probleme bedienen kann. Weiter sollte sich die Bedienung für das Gerät natürlich anfühlen. Es sollte nicht unnötig versucht werden die reale Welt nachzuahmen. Alle Bedienelemente sollten den jeweiligen Kontext beachten und die richtigen Metaphern einsetzen. Zum Schluss sollte nicht versucht werden vorhandene Paradigmen für Benutzerschnittstellen zu kopieren. Denn diese sind meistens speziell für bestimmte Eingabegeräte, z.B. die Maus, optimiert. Insgesamt soll keine natürliche Benutzerschnittstelle (*natural* user interface) entworfen

werden, sondern eine Benutzerschnittstelle, die dem Benutzer das Gefühl gibt ein Naturtalent zu sein (*natural user* interface).

Ein Prinzip das beim Erstellen eines Natural User Interface beachtet werden sollte ist, dass der Entwurf einfach aufgebaut werden sollte. Es sollen Möglichkeiten gesucht werden mit einfachen Interaktionen schwierige Aufgaben zu unterstützen. Eine NUI sollte keine WIMP-GUI mit neuen Interaktionsmöglichkeiten sein. Der Entwurf sollte auf dem Prinzip aufbauen, wie ein erfahrener Anwender auf dem entsprechenden Gebiet vorgeht.

Um eine angenehme und natürliche Erfahrung aufzubauen, sollten die Interaktionen in einer NUI immer dem aktuellen Kontext entsprechen. Wenn Funktionen oder Interaktionen nicht dem Kontext entsprechen, verwirrt das den Benutzer und schmälert die positive Erfahrung mit der Anwendung.

Die Natural User Interfaces können auch von mehreren Benutzern verwendet werden. Das stimmt zwar nicht für Smartphones aber für Tabletop-PCs ist es durchaus denkbar. Für dieses gemeinsame Nutzen gibt es verschiedene Ebenen der Zusammenarbeit.

- Starke Zusammenarbeit („highly coupled tasks“)
  Mehrere Benutzer arbeiten gleichzeitig an derselben Aufgabe und helfen sich dabei Gegenseitig.
- Leichte Zusammenarbeit („lightly coupled tasks“)
  Mehrere Benutzer arbeiten an der gleichen Aufgabe. Es gibt aber keine Interaktion der zwischen ihnen. „Divide and Conquer“ könnte hier eingeordnet werden.
- Getrennte Aufgaben („uncoupled tasks“)
  Mehrere Benutzer arbeiten auf der gleichen Arbeitsfläche, aber sie bearbeiten unterschiedliche Aufgaben.

Das nahtlosen Feedback ist eine wichtige Eigenschaft für NUIs. Wenn die Benutzerschnittstelle offensichtliche Diskontinuitäten hat, irritiert es den Benutzer im besten Fall. Das führt dazu, dass der Benutzer die Lust verliert, das System zu verwenden. Im schlimmeren Fall führt es zur Unterbrechung des Interaktionsablaufs und zur Fehlbedienung des Systems.

Eine Erwartung an NUIs ist meist, dass sich die Bedienung wie Magie anfühlen sollte. Damit ist meist gemeint, dass komplexe Funktionen mit einfachen und intuitiven Interaktionen umgesetzt werden können. Dadurch wird der Benutzer positiv überrascht und denkt sich „Das geht so einfach?!“.

Der Begriff der NUIs wird häufig auf berührungsempfindliche Systeme angewandt, da diese sich momentan großer Beliebtheit erfreuen und im Vergleich zur Maus und Tastatur eine grundsätzlich verschiedene Art der Interaktion darstellen. Doch „Brave NUI World“ [23] behauptet, dass ihre Entwurfsideen genauso auf andere Eingabegeräte angewendet werden können.

2.2.4 Tangible User Interfaces

Die Tangible User Interfaces(TUI) sind Benutzerschnittstellen, die es dem Benutzer ermöglichen mit Objekten aus der realen Welt mit dem Computer zu interagieren. Diese Objekte werden meist Tangible genannt. Der Begriff kommt von dem lateinischen Wort „tangere“ und bedeutet berühren oder anfassen. Die Tangibles repräsentieren digitale Daten oder Funktionen. Die Manipulation der Tangibles bewirkt daher eine Manipulation der digitalen Daten.

Ishii et al. [24] beschreiben einige Prinzipien für TUIs.

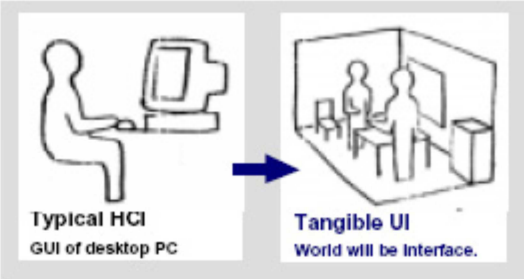


Abbildung 7: TUIs führen weg von der Bedienung des Computers mit Maus und Tastatur. Alle Elemente des Raums dienen zur Interaktion. [24]

Die Arbeit definiert TUIs folgendermaßen: “TUIs will augment the real physical world by coupling digital information to everyday physical objects and environments.” [24]. Das heißt: die reale Welt wird erweitert, in dem digitale Informationen mit physikalischen Objekten und Umgebungen gekoppelt werden (siehe Abbildung 7). Die grundsätzlichen Ideen um das umzusetzen sind dabei:

- „Interactive Surfaces“
  Jede Oberfläche in der realen Welt soll zu einer aktiven Schnittstelle zwischen der realen und der virtuellen Welt werden.
- „Coupling of Bits and Atoms“
  Jedes reale Objekt soll mit digitalen Objekten gekoppelt werden können.
- „Ambient Media“
  Schall, Licht, Luftströmungen und Wasserbewegung sollen als Hintergrundschnittstellen dienen.

Das Ziel wird folgendermaßen zusammengefasst: Jede Art von Materie, flüssige oder feste, soll zu Schnittstellen zwischen der virtuellen und der physikalischen Welt werden.

Eine der ersten Visionen zu den Tangible User Interfaces ist die Marble Answering Machine. Hier wird jeder Anruf symbolisch in einer Murmel gespeichert. Wenn ein Anruf ankommt rollt die Murmel aus dem Gerät. Wird die Murmel in eine bestimmte Kuhle gelegt wird der Anruf abgespielt. Wird die Murmel in das Gerät zurückgelegt wird der Anruf gelöscht.

Neben der Marble Answering Machine haben auch Ullmer et al. [25] eine Vision für ein Tangible User Interface. Dabei können mit Tangibles Daten physikalisch zwischen Geräten ausgetauscht werden. Jedes Gerät besitzt einen Slot, in den ein sogenannter MediaBlock eingelegt werden kann. Dann kann über das Display Daten an den MediaBlock übertragen werden. Dieser MediaBlock kann die Daten dann an einem anderen Gerät wieder ausgeben. Weiterhin existieren Geräte, die Daten von mehreren MediaBlocks manipulieren können. Die Daten des Systems sind digitale Medien wie z.B. Bilder und Videos. (siehe Abbildung 8)

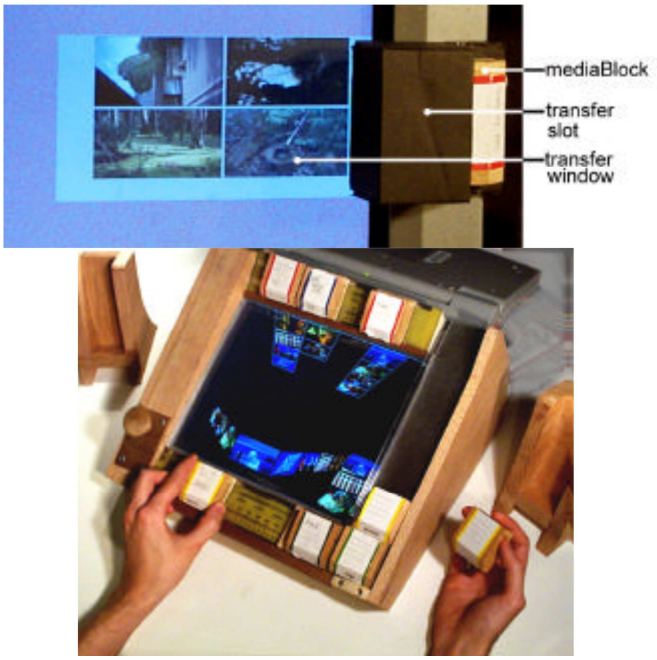


Abbildung 8: Oben: Ein Slot eines MediaBlocks auf den Bilder übertragen werden. Unten: Ein MediaSequenzer, um Medien zu bearbeiten. [25]

Eine kommerziell genutztes TUI ist der ReacTable [26]. Er ist ein digitales Musikinstrument. Er basiert auf einem Tabletop-PC, der hauptsächlich mit tangibles bedient wird. Jedes tangible ist entweder eine Datenquelle für Geräusche oder eine Operation, die eine Geräuschquelle verändert. Die tangibles interagieren, indem sie in der Nähe voneinander auf dem Tabletop-PC platziert werden.

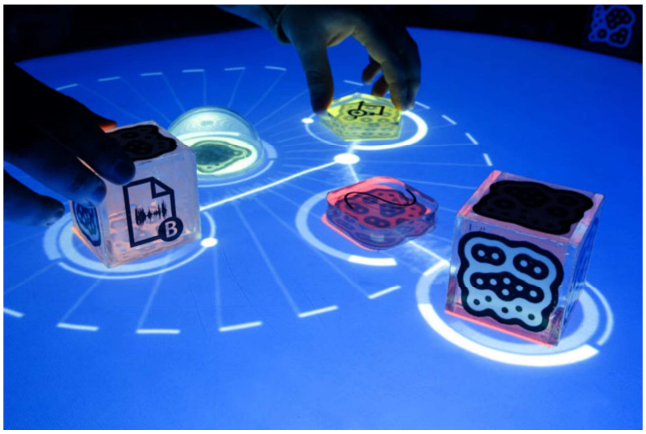


Abbildung 9: Benutzer beim Musikmachen mit dem ReacTable. (Interaktion mit Tangibles) [26]

2.3 Tabletops

Ein „Tabletop-Computer“ ist, wie der Name sagt, ein Computer, der in einen Tisch eingebettet wurde. Die Tischfläche ist größtenteils eine Anzeige, kann aber auch eine Ablagefläche am Rand haben. Die Anzeige kann, entweder durch ein Display, durch Rückprojektion oder über ein „overhead“-Projektor umgesetzt sein. Die Eingabe erfolgt fast immer über ein „multitouch-panel“. Das ermöglicht dem Benutzer die Eingabe durch

Berührung und taktile Gesten. Bei manchen Techniken ist es außerdem möglich mit Tags versehene Objekte also tangibles zu erkennen.

Das Kapitel beschreibt als erstes einige Techniken für Eingabehardware, die bei Tabletops eingesetzt wird. Danach werden einige existierende Hardwarekonfigurationen und ihre Software aufgezählt.

2.3.1 Eingabetechniken für Tabletops

Für multitouch-Eingabe gibt es viele verschiedene Verfahren. Unterteilt werden können sie in resistive, kapazitive, auditive und optische Verfahren. Einen Überblick über die Verfahren geben Chang et al. [27] und Brandl et al. [28].

Resistiv

Ein resistiver Touchscreen nutzt zwei elektrisch leitende Schichten, zwischen denen bei einer Berührung ein leitender Kontakt hergestellt wird. Die Schichten bestehen aus Indiumzinnoxid, einem lichtdurchlässigen Halbleiter. Zwischen den beiden Schichten ist ein Abstandshalter, der eine Berührung ohne Nutzereinwirkung verhindert. Bei diesen handelt es sich oft um kleine Silikonpunkte. An der Oberfläche muss sich eine flexible Schicht befinden damit eine Krafteinwirkung auf die leitenden Schichten möglich ist. Die Oberflächenschicht besteht oft aus Polyester, was zwar flexibel ist, aber auch anfällig für Beschädigungen.

Der Berührungspunkt wird durch Messen der Spannung an den Rändern ermittelt. Wenn an die obere Schicht eine Spannung angelegt wird und durch Berührung mit der unteren verbunden wird, lässt sich an deren Rändern zwei Spannungen messen. Diese geben durch den Spannungsverlust Aufschluss wo sich der Berührungspunkt befindet.

Resistive Touchscreens lassen sich recht günstig herstellen und kommen daher in vielen einfachen Anwendungen zum Einsatz. Allerdings benötigt man industrielle Fertigungsmethoden und sie sind daher nicht für den Eigenbau geeignet. Ihre Genauigkeit ist gut, nur benötigt Multitouch-Erkennung komplexere Hardware.

Kapazitiv

Bei kapazitiven Touchscreens gibt es zwei grundsätzliche Techniken. Die Techniken nutzen entweder Oberflächenkapazität oder projizierte Kapazität.

Beim Oberflächenkapazitätsverfahren wird eine gleichmäßige elektrische Ladung auf einer durchsichtigen leitenden Fläche, zum Beispiel Indiumzinnoxid, erzeugt (siehe Abbildung 10). Da menschliche Finger kapazitive Eigenschaften besitzen, daher eine Ladung speichern können, bewirkt das berühren des Displays mit den Fingern einen schwachen elektrischen Fluss. Der Strom wird dabei aus den Ecken der Fläche abgezogen. Dies wird von Sensoren, die in den Ecken positioniert sind gemessen. Die Stärke des Stroms gibt Auskunft wie weit der Berührpunkt von den Ecken entfernt ist. Mit Hilfe eines Mikrocontrollers wird zum Schluss ausgerechnet wo sich der Berührpunkt genau befindet. Auf diese Weise lässt sich eine sehr hohe Genauigkeit beim Erkennen der Position erreichen.

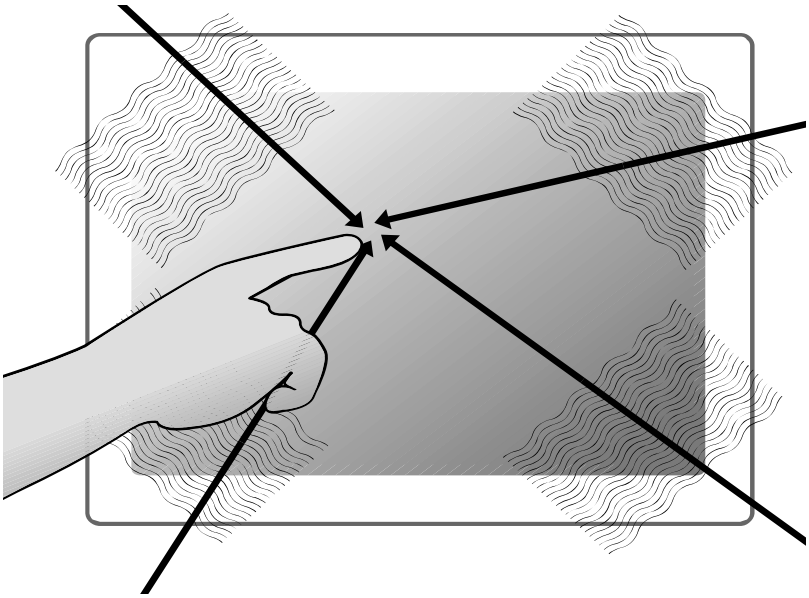


Abbildung 10: Skizze zum Oberflächenkapazitätsverfahren [28]

Das „projizierte Kapazität“-Verfahren funktioniert mit sehr dünnen Kabeln, die als Netz in einer Schicht des Displays verlegt werden.(siehe Abbildung 11) Die Kabel in eine Richtung dienen als Sender eines Signals. Die anderen Kabel dienen als Empfänger. Wenn ein Finger nun das Display berührt verändert das die elektrischen Eigenschaften des empfangenen Signals. Dies geschieht durch die Kapazität die durch den Finger erzeugt wird, wenn er das Display berührt. Durch die Art des Signals und Position im Netz in der es empfangen wurde, kann die Position des Berührpunkts ermittelt werden. Dieses Verfahren hat eine ähnliche Genauigkeit wie das Oberflächenkapazitätsverfahren. Das Netz der Kabel kann so gebaut werden, dass es beinahe unsichtbar ist. Daher bietet das „projizierte Kapazität“-Verfahren eine höhere Transparenz als die bisher vorgestellten Verfahren. Außerdem kann bei diesem Verfahren eine dickere Schutzschicht eingesetzt werden ohne dass die Genauigkeit beeinträchtigt wird. Ein erheblicher Nachteil des Verfahrens sind die hohen Produktionskosten.

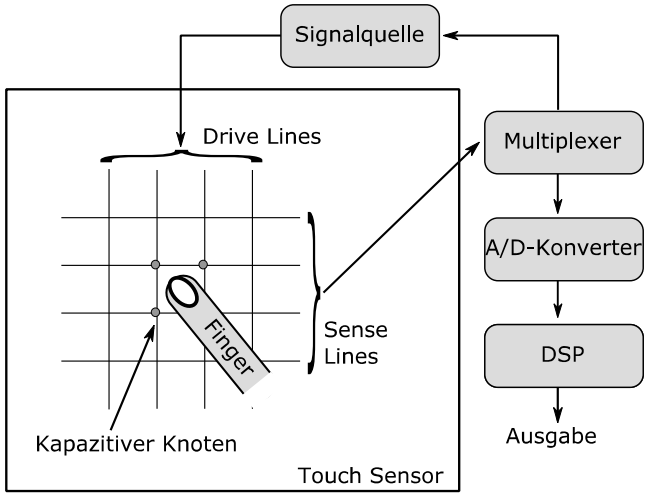


Abbildung 11: Skizze zum Verfahren der „Projizierte Kapazität“ [28]

Allgemein sind kapazitive Verfahren teurer und benötigen industrielle Herstellungsmethoden. Ihre Genauigkeit ist akzeptabel vor allem bei Fingereingabe und sie können sehr resistent gebaut werden. Multitouch-Erkennung ist zudem sehr einfach möglich.

Oberflächenwellen
Das Oberflächenwellen-Verfahren oder Surface Wave (SAW) in Englisch basiert auf Schallwellen. Auf einer Oberfläche werden vertikal und horizontal Ultraschallwellen erzeugt und an den Rändern wieder gemessen. Wenn ein Finger oder Objekt auf die Oberfläche kommt, verändert sich die Wellenstruktur und lässt Rückschlüsse auf die Position des Berührpunkts zu.

Das Verfahren ist sehr nützlich für öffentliche Nutzung, da es günstige und recht resistente Hardware nutzt. Die meisten Verfahren unterstützen Dualtouch. Die Genauigkeit ist eher schlecht.

Frustrated Total Internal Reflection (FTIR)
FTIR ist ein optisches Verfahren. Es basiert auf einem Konzept, das von Han [29] [30] vorgestellt wurde. Es basiert, wie der Name schon sagt, auf dem physikalischen Prinzip der inneren totalen Reflexion. Es besagt, dass elektromagnetische Wellen innerhalb eines Materials total reflektiert werden, wenn das Material einen höheren Brechungsindex hat als die Umgebung und die Auftreffwinkel genügend klein sind. Der übliche Aufbau einer FTIR-Umgebung ist daher eine Plexiglasscheibe an deren Seiten Infrarot-LEDs angeordnet sind. Da Plexiglas einen höheren Index hat und das Infrarotlicht sehr flach auf die Flächen der Scheibe treffen, werden sie total reflektiert. Wenn ein Benutzer die Scheibe berührt, ändert sich der Brechungsindex der Umgebung und das Infrarotlicht verlässt die Scheibe und wird stattdessen an dem Finger reflektiert. Unter der Scheibe befindet sich eine Kamera, die das vom Finger reflektierte Licht detektiert. Die Kamera kann so alle Berührpunkte auf der Scheibe sehen. Um störendes Umgebungslicht auszublenden, hat die Kamera in der Regel noch einen Filter, der nur Infrarotlicht durchlässt. So hat die Kamera ein Bild auf dem nur die Berührpunkte zu sehen sind. Dieses Bild wird mit Bilderkennungsmethoden analysiert und die Position der Berührpunkte berechnet. Um ein Bild auf der Scheibe anzuzeigen hat sie an der Oberfläche noch eine Projektionsfläche auf die ein Beamer von unter der Scheibe ein Bild projizieren kann.

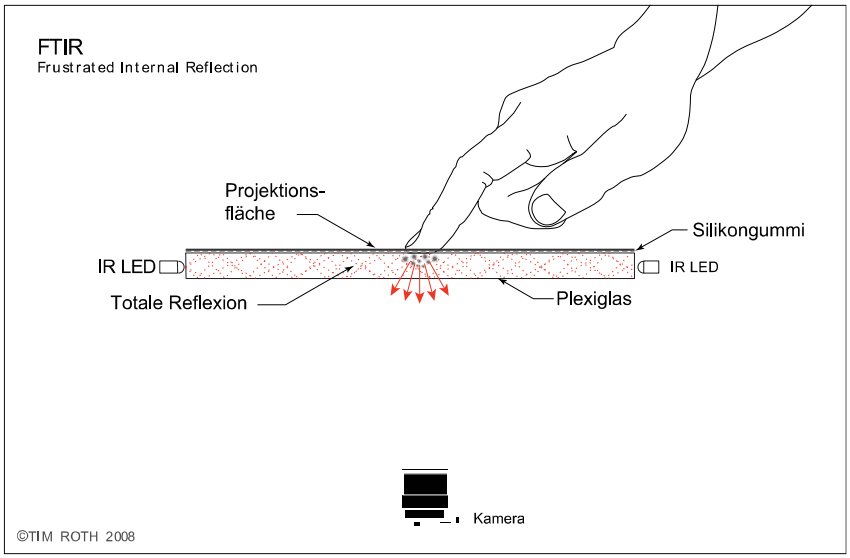


Abbildung 12: Darstellung von FTIR [28]

Mit dieser Technik lässt sich mit einfachen Methoden eine rückprojizierte interaktive Oberfläche erstellen. Wenn das System gut kalibriert ist, hat das Verfahren eine hohe Genauigkeit. Das Verfahren unterstützt zudem beliebiges Multitouch und wird grundsätzlich nur durch die Bilderkennung begrenzt. Das Erkennen von „tagged objects“ ist leider nicht möglich, da die Oberfläche und Farbe eines Objekts nicht erkannt wird. Starkes infrarotes Umgebungslicht kann die Erkennung stören.

Diffuse Illumination (DI)
Der Aufbau des Systems ist bei diesem Verfahren ähnlich wie bei FTIR. Hier wird die Plexiglasscheibe jedoch von unten mit Infrarotlicht angestrahlt. Daher ist nicht nur die Scheibe mit Infrarotlicht durchsetzt, sondern das Licht geht über die Scheibe hinaus. Die Kamera nimmt wieder das reflektierte Infrarotlicht auf. Im Gegensatz zu FTIR kann man so schon Objekte detektieren, die noch gar nicht die Scheibe berühren. Außerdem wird auch die Oberfläche der Objekte sichtbar. So können zum Beispiel Markierungstags auf flachen Oberflächen erkannt werden.

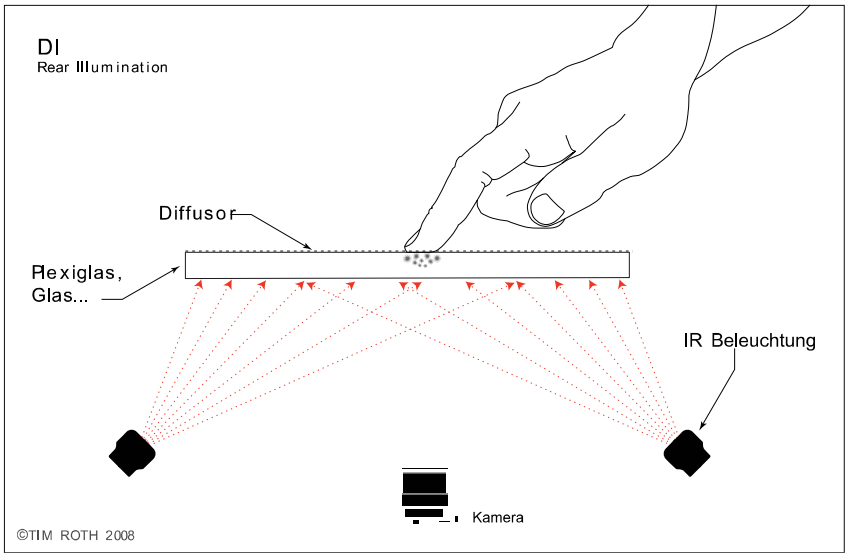


Abbildung 13: Darstellung zu DI [28]

Zum Schutz kann über das Display zusätzlich noch eine transparente Schutzschicht montiert werden. Die Detektion wird dadurch nicht gestört. Wie FTIR unterstützt DI eine mehr oder weniger unbegrenzte Anzahl an Berührungspunkten. Die Erkennung ist je nach Kalibrierung und Bilderkennungsalgorithmus sehr genau. Starkes Infrarotlicht in der Umgebung kann auch hier die Erkennung beeinträchtigen.

Diffused Surface Illumination (DSI)
DSI ist eine Erweiterung von FTIR. Bei dem Verfahren wird die Ausstrahlung des Lichts aus der Scheibe hinaus verbessert. Dadurch wird es möglich Objekte, die nicht die Scheibe berühren, zu erkennen. Der Aufbau ist prinzipiell der gleiche wie bei FTIR nur wird bei dieser Technik eine spezielle Plexiglasscheibe verwendet. Die Scheibe enthält kleine Teilchen, die wie kleine Spiegel wirken und einfallendes Licht gleichmäßig über die Oberfläche der Scheibe verteilen. Der Markenname einer solchen Scheibe ist „Plexiglass Endlighten“.

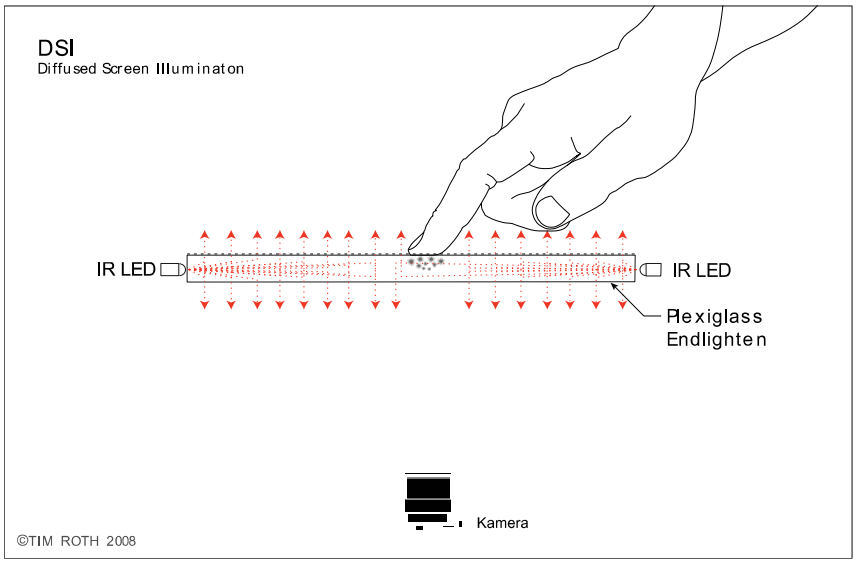


Abbildung 14: Darstellung zu DSI [28]

Damit ist es möglich „tagged objects“ zu erkennen. FTIR Aufbauten lassen sich relativ leicht zu DSI umbauen. Ein Nachteil ist, dass DSI weniger Kontrast liefert als DI, da Licht auch in Richtung der Kamera reflektiert wird. Aus diesem Grund stellt infrarotes Umgebungslicht auch ein größeres Problem dar, als bei den anderen Verfahren.

Microsoft PixelSense

Microsoft PixelSense gehört zu den in LCDs integrierten Berührungserkennungssystemen. Die Tatsache, die bei diesen Verfahren ausgenutzt wird ist, dass das infrarote Licht beinahe ungehindert die LCD-Schichten durchdringen kann. (siehe Abbildung 15)

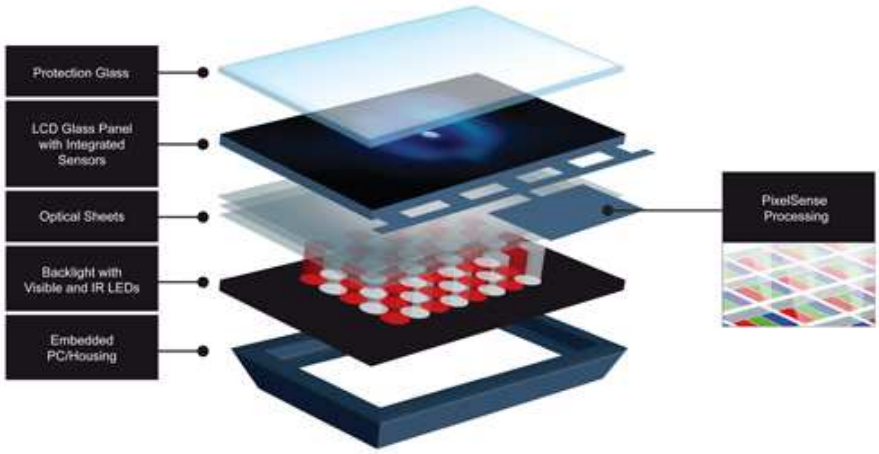


Abbildung 15: Darstellung der Pixelsense Hardware [8]

Unter der LCD-Schicht befindet sich zusätzlich zu der Hintergrundbeleuchtung eine Matrix aus Infrarot-LEDs. Diese strahlen Infrarotlicht durch die LCD-Schicht nach außen. Das Infrarotlicht wird durch Finger oder andere Gegenstände auf oder über der Oberfläche reflektiert. Das zurückgeworfene Infrarotlicht wird innerhalb der LCD-Schicht bei jeder Pixeleinheit detektiert. Dadurch kann wiederum ein Bild von den reflektierenden Objekten

auf der Oberfläche erstellt werden. Ähnlich wie bei den anderen optischen Verfahren dient dieses Bild über „Visual Computing“-Verfahren zur Erkennung der Berührpunkte oder anderer Objekte.

2.3.2 Technologiebeispiele

Dieses Kapitel beschäftigt sich mit existierender Tabletop-Hardwarekonfigurationen, die einige der im vorherigen Kapitel genannten Techniken nutzten.

Microsoft Surface 1 (Codename Milan)
Die Entwicklung des Microsoft Surface 1 begann 2001 in der Konzeptphase. Im Jahr 2003 wurde die Idee dem Vorstand vorgestellt und das Entwicklerteam vergrößert. Daraufhin wurden einige Prototypen und Softwareprojekte entwickelt. Die Veröffentlichung des fertigen Surface 1 ist im Jahr 2007.

Der Surface 1 [8] basiert auf der DI-Technik und kann daher beliebig viele Berührpunkte und tangibles mit Hilfe von „Tags“ erkennen. Microsoft hat dafür zwei Arten von Tags entwickelt. Die Byte-Tags sind 256 Tags deren Position und Ausrichtung erkannt werden kann. Die zweite Art von Tags sind die Identity-Tags, die QR-Codes ähneln und 128 Bits an Daten enthalten. Sie können ebenfalls in der Ausrichtung erkannt werden. Die Tags basieren auf dem Prinzip der „fiducials“, also Markierungen, die von optischen Systemen erkannt werden können.

Der Surface 1 nutzt das Microsoft Surface SDK 1 [31] als Framework. Es bietet die Erkennung von einfachen Gesten und hat Interaktionselemente für Mehrbenutzerunterstützung. Die Benutzerschnittstelle ist nach Entwurfskonzepten für NUIs entwickelt worden. Daher können mit dem SDK vornehmlich NUI-Anwendungen umgesetzt werden.

Microsoft Pixelsense (vorher Microsoft Surface 2)
Als Nachfolger des Microsoft Surface 1 wurde der Samsung SUR40 entwickelt. Der mit dem Microsoft Surface SDK 2 betrieben wird.

Der Samsung SUR40 [8] ist im Gegensatz zum Surface 1 ein etwas dickeres LCD-Display mit Rahmen. Dies ist mit der „Microsoft Pixelsense“-Technik möglich. Dadurch kann der SUR40 mit den entsprechenden Beinen als Tisch verwendet werden, aber auch senkrecht an einer Wand befestigt werden. Das macht seine Verwendung sehr viel flexibler als es bei dem Vorgänger der Fall war.

Das Surface SDK 2 [32] wurde mit dem .NET Framework 4 kompatibel gemacht und unterstützt somit auch die Einbindung des Touch-Frameworks von Windows 7. Das erleichtert eine gemeinsame Entwicklung mit Tablets erheblich. In der Funktionalität ist die Version 2 des Surface SDKs dem Vorgänger ebenbürtig. In der aktuellen Version fehlt leider eine Umsetzung der Identity-Tags.

RaecTable Tisch [33]
Der ReacTable ist ein digitales Musikinstrument. In diesem Projekt wurde ebenfalls ein Tabletop entwickelt um mit Tangibles Musik zu generieren. Die Hardware des reactable basiert auf DI. Zur Erkennung der Tangibles und Berührpunkten wurde ein eigenes Framework mit dem Namen „reacTIVision“ entwickelt. Die Tangibles funktionieren hier ebenfalls mit „fiducials“, die ebenfalls speziell für das Projekt entwickelt wurden. Sie haben den Namen „amoeba“ und sind hierarchisch aufgebaute Punktkarten.

2.4 Powerwalls

Mit Powerwalls sind meist hochauflösende große Displays gemeint. In diesem Kapitel werden einige Hardwarekonfigurationen und Anwendungsbereiche für diese Displays vorgestellt [6].

2.4.1 Hardwarekonfigurationen

Es gibt verschieden Arten von Hardware mit denen großflächige Displays aufgebaut werden können. Im Folgenden werden die Aufbauten „Tiled LCD Panels“, „Projector Arrays“ und „Stereoscopic Displays“ vorgestellt [6].

Tiled LCD Panels

Die Tiled LCD Panels sind mehrere LCDs, die zu in einer Matrix nebeneinander angeordnet werden. Sie können entweder senkrecht an einer Wand angeordnet werden, was der Idee einer Powerwall entspricht. Oder die LCDs werden waagerecht angeordnet und ähnlich einem Tisch aufgestellt. Diese Methode hat verschiedene Vorteile. Sie können einfacher ausgerichtet werden als z.B. Projektoren. Sie sind günstig zu beschaffen. Die LCDs brauchen nicht viel Platz im Gegensatz zu einem Aufbau mit Rückprojektion. Der hauptsächliche Nachteil der Technik sind die Ränder der LCDs. Das wird bei Textdarstellung problematisch, da der Text an den Rändern geteilt wird.

Abbildung 16: Beispiel für Tiled LCD Panels [34]
Links: Die lambdaVision
Rechts: Der lambdaTable

Projector Arrays

Die Projektor Arrays sind Anordnungen von LCD oder CRT Projektoren. Sie sind meist sehr teuer in der Anschaffung und auch die Instandhaltung ist teuer, da die Lampen der Projektoren nach einiger Zeit ausgetauscht werden müssen. Sie sind aber dennoch beliebt, da sie keine Ränder im Bild haben wie die LCDs. Die Projektoren können so angeordnet werden, dass eine leichte Überlappung der Bilder entsteht. Wenn dann an den Rändern der Bilder die Helligkeit nach außen hin abfällt, kann ein beinahe nahtloses Bild entstehen. Weiterhin ist bei der Abbildung der Projektoren die Größe des Bildes nicht an die Größe des Geräts gebunden. Die Größe des Bildes kann durch den Abstand des Projektors zur Projektionsfläche eingestellt werden. So können einfach sehr große Bilder erzeugt werden. Der Nachteil dabei ist jedoch, dass dieser Abstand Platz benötigt.

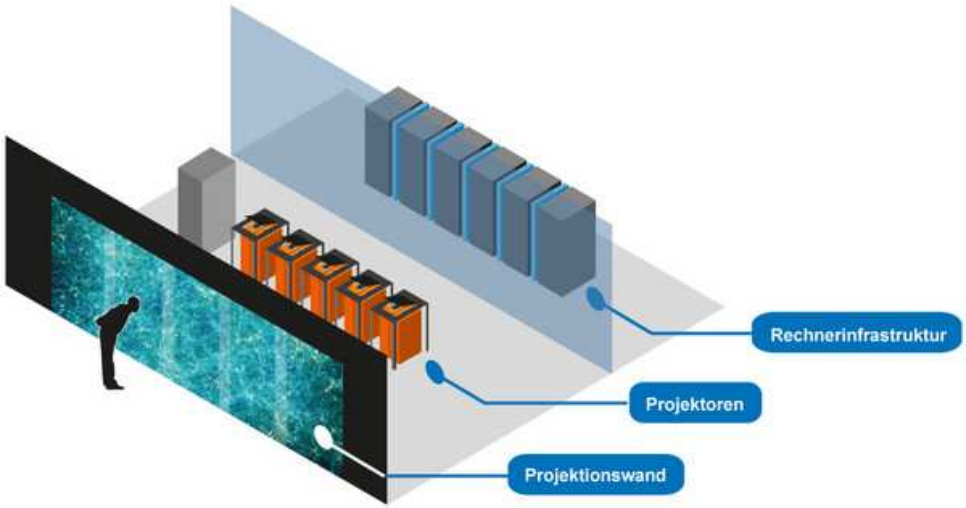


Abbildung 17: Aufbau der Powerwall im VISUS-Labor mit einem Projektor Array [35]

Stereoscopic Displays
Stereoskopische Displays zeigen dem Betrachter ein 3D-Bild an. Das 3D-Bild wird durch zwei Pixel an der gleichen Position generiert. Dabei ist ein Pixel für das linke Auge und ein Pixel für das rechte Auge. Meistens benötigen diese Displays spezielle Brillen für den Benutzer, die die beiden Pixel für das linke und rechte Auge trennen. Inzwischen gibt es aber auch Entwicklungen zu autostereoskopischen Displays, die keine Brillen benötigen. Stereoskopische Ansätze gibt es sowohl für LCDs als auch für Projektoren.

Abbildung 18: Beispiel eines autostereoskopischen Displays(The Varrier) [36]

2.4.2 Anwendungsgebiete
Für die großen hochaufgelösten Displays gibt es einige Anwendungen. Schmidt et al. [6] stellen einige dieser Anwendungsgebiete vor.

- Command and Control
  In einigen Kommandozentralen und Koordinationsstellen von Militär, Flugsicherheit oder Telekommunikation sind heutzutage Powerwalls eingebaut. Das Air Force Research Laboratory entwickelte die „Interactive DataWall“. Sie ist ein großes hochauflösendes Display mit drahtlosen Interaktionsgeräten. Damit sollte die Entscheidungsfindung in Gefechtssituationen unterstützt werden.

- Vehicle Design
  Um Modelle von Fahrzeugen im 1:1 Maßstab darzustellen kann bei dem Entwurf von Fahrzeugen auf die Größe von Powerwalls zurückgegriffen werden. Gleichzeitig kann im Team an den Modellen gearbeitet werden. Nach dem Entwurf können in VR-Umgebungen Konstruktionen simuliert und die Fahreigenschaften getestet werden.

- Geospatial Imagery and Video
  Große hochauflösende Displays bieten ein Gefühl für den Maßstab, der bei geographischen Darstellungen und Videos benötigt wird. Große Displays ermöglichen das Entdecken von kritischen Informationen unter anderem bei dynamischen Systemen, wie z.B. subtile Wirbel bei Ozeanzirkulationsmodellen.

- Scientific Visualization
  Powerwalls sind eine der beliebtesten Möglichkeiten für wissenschaftliche Visualisierungen, denn sie bieten eine Anzeige von Daten in Lebensgröße oder menschenähnlicher Größenmaßstäbe. Außerdem können mit der größeren Menge an Pixeln viele Daten gleichzeitig betrachtet werden.

- Collaboration
  Für das Arbeiten in Gruppen oder Teams eignen sich große Displays ebenfalls sehr. Sie können für die Präsentation oder bei einem gegenseitigen Austausch wie einer Diskussion eingesetzt werden. An der Universität Konstanz wurde hierzu beispielsweise eine Powerwall entwickelt, die mit Laserpointer bedient werden kann.

- Public Information Displays
  Wo einst gedruckte Bilder und oder einfache digitale Bilder zu sehen waren, werden heutzutage Tiled Displays eingesetzt. Dies wird durch die Entwicklungen bei den Flachbildschirmen und Projektoren ermöglicht, welche sehr viel günstiger geworden sind. Ebenfalls sind die Displays nicht mehr auf flache Werbetafeln beschränkt, beinahe jede Oberfläche kann für die Darstellung von Bildern und Informationen verwendet werden.

3 Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten beschrieben. Das schließt Arbeiten zu multimodalen Interaktionsframeworks ein, sowie fertige multimodale Interaktionsumgebungen und Anwendungen auf Tabletops.

3.1 Multimodale Interaktionsframeworks

Das Squidy-Framework [37] ist eine Sammlung von Frameworks und Toolkits für die Erstellung von multimodalen Umgebungen nach der Idee von NUIs. Dabei wird das Augenmerk hauptsächlich auf neuere Eingabegeräte gelegt, wie Multitouch-Displays, Wii-Controller und ähnliche. Im Gegensatz zu klassischen Oberflächen gibt es aktuell in den Entwicklungswerkzeugen keine Unterstützung für die Erstellung von NUIs. Darum ist der wichtigste Bestandteil von Squidy ein grafisches Tool zur Erstellung von solchen Systemen. Das grafische Tool baut eine Architektur nach dem Prinzip von Pipes&Filters auf. Grundsätzlich stellt es den Datenfluss zwischen Eingabe und Ausgabe dar. Dazwischen können Filter eingesetzt werden um z.B. die Eingabe zu verbessern. Die Ausgabe kann alle möglichen Formen annehmen. Es können Aktoren wie Elektromotoren oder LEDs angesteuert werden. Außerdem können Signale oder Nachrichten in verschiedenen Protokollen und Standards als Ausgabe versendet werden. Dadurch lassen sich andere Anwendungen anbinden. Zusammengesetzt werden die entsprechend Pipes&Filters in einem Graph, wobei Eingabeknoten Startknoten bzw. Quellen sind und Ausgabeknoten Endknoten bzw. Senken sind. Die Filterknoten können dazwischen angeordnet werden. Squidy besitzt bereits zu Beginn einige Eingabe-, Filter- und Ausgabeknoten in seiner Knowledgebase. Diese kann um weiter Knoten erweitert werden. Die Knoten können entweder als Plugin eingebunden werden oder über das grafische Tool aus bestehenden Knoten erstellt werden. Squidy hat keine Unterstützung für den Datenaustausch zwischen den Geräten oder Darstellung auf den Geräten, das Framework ist allein für die Interaktion zuständig.

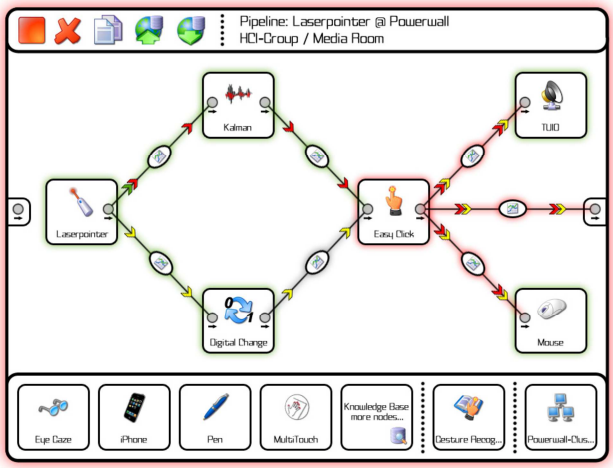


Abbildung 19: Eine Pipeline im Squidy-Tool

Shared Substance [38] geht in eine andere Richtung. Hier wird versucht ein datenorientiertes Framework zu bauen, das seine Ressourcen über mehrere Geräte freigeben kann. Bei den Ressourcen, die in der Umgebung freigegeben werden können, handelt es sich um Interaktionselement, wie z.B den Mauszeiger, oder ganze Displays. Weiterhin können Daten, wie Benutzerinhalte, freigegeben werden. Außerdem bietet das Framework auch die Möglichkeit ganze Anwendungen und deren Funktionalität zwischen den verschiedenen

Geräten zu teilen. Das Framework ist nach dem Paradigma der Datenorientierung entworfen und nützt beim Prototyp die Programmiersprache Python, da sie Meta-Programmierung unterstützt. Shared Substance hat ein sehr umfangreiches Konzept für das Teilen von Anwendungen und Daten. Die Möglichkeit private und öffentliche Bereiche zu nutzen wird aber nicht angesprochen.

3.2 Multimodale Interaktionsumgebung

Das Projekt ZOIL [39] ist ein Prototyp für eine multimodale Interaktionsumgebung für Powerwalls, Tabletops und anderen Geräten. Das Projekt baut auf einer objektorientierten Benutzeroberfläche auf. Die Objekte sind sowohl für die Daten zuständig, als auch für die Elemente, die diese anzeigen. ZOIL verfolgt dabei das Ziel einen breiten Bereich des Personal Information Management umzusetzen. Dabei versucht das Projekt neue Paradigmen für Benutzerschnittstellen im Bereich des Personal Information Managements zu finden. ZOIL unterstützt zwar Mehrbenutzerbetrieb, legt aber keinen gesonderten Blickpunkt auf ein Mehrbenutzersystem. In ZOIL wird in diesem Hinblick hauptsächlich auf Mehrgerätesysteme gesetzt. Bei der Bedienung sticht ZOIL vor allem bei der großen desktopartigen Fläche heraus. Diese hat nämlich die Fähigkeit semantisch gezoomt zu werden. Dadurch können im Überblick unnötige Details ausgeblendet werden und bei der Detailansicht ist alles zu sehen. Auch Tangibles finden in dem Projekt eine Anwendung. Es wird eine Art virtuelle Linse mit einem durchsichtigen Rahmen als Tangible verwendet. Diese erlaubt es Teile der Ansicht zu zoomen. Die gezeigten Anwendungen des Prototyps sind hauptsächlich für Medien wie Videos und Musik.

Im Projekt Code Space [40] von Microsoft Research wird eine Powerwall mit Gestenerkennung für die Besprechung von Codeanalysen verwendet. Mithilfe von Mobilgeräten und Presentern können die Entwickler dargestellte Elemente auf der Powerwall mit einer Zeigegeste markieren und verändern. Ebenfalls können mit Berührungen Elemente auf der Powerwall manipuliert werden. Das Projekt zeigt die Anwendung einer Kombination von Berührungs- und Freihandgesten. Die Arbeit zeigt, dass die Anwendung des Projekts sowohl hilfreich für die Entwickler ist, als auch akzeptabel in ihrem sozialen Charakter.

Das Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung ist mit dem SmartControlRoom [41] dabei, einen „aufmerksamen und mitdenkenden“ Raum zu entwickeln. Dieser soll als Krisenzentrum bei Großschadenslagen eingesetzt werden können. Der Raum hat eine Powerwall und ist rundherum mit Videokameras und Mikrofonen ausgestattet. Er soll auf die Menschen im Raum reagieren. Die Powerwall zeigt Informationen zur Lage und kann mit Gesten oder Sprache gesteuert werden. Der Raum kann ebenso bei der Planung von Großveranstaltungen, bei terroristischer Bedrohung oder dem Schutz kritischer Infrastrukturen eingesetzt werden. Dieses Projekt setzt auf freie Interaktion ohne Eingabegeräte oder anderen Objekten wie Tangibles.

In dem Projekt Roomware [42] wird versucht ein Raum nach dem Prinzip des „ubiquitous computing“ [11] aufzubauen. Dabei wird Informationstechnologie in Komponenten des Raums integriert, wie z.B. Tische, Stühle, Wände. Die Ausprägungen sind die bekannten Powerwalls und Tabletops. Das Ziel ist, dass die „Welt um uns“ zur Schnittstelle mit der virtuellen Welt wird.

Abbildung 20: Links ist ein Auschnitt der ZOIL-Oberfläche zu sehen. Rechts sieht man die multimodale Interaktionsumgebung von Roomware.

3.3 Tabletop-Anwendungen

Der ReacTable [26] ist ein elektronisches Musikinstrument mit einem Tangible User Interface. Die einzelnen Tangibles sind physisch Acrylscheiben oder -würfel. Die Tangibles repräsentieren entweder Steuerelemente oder Datenelemente. Jedes Tangible, das auf den Tisch gelegt wird, bekommt ein Interaktionsfeld und baut eine Verbindung zu anliegenden Tangibles auf. Das Interaktionsfeld kann per Berührung manipuliert werden. Der Reactable kann dabei von mehreren Personen gleichzeitig bedient werden. Ein Tangible repräsentiert eine Datenquelle, in diesem Fall eine Geräuschquelle oder ein Steuerelement. Steuerelemente sind in diesem Fall Filter, die die Daten manipulieren. Außerdem können die Interaktionselemente, die von den Tangibles generiert werden, ebenfalls per Berührung manipuliert oder gesteuert werden. Auch eine Interaktion zwischen mehreren ReacTables ist in dem Projekt miteinbezogen.

Abgesehen von Spielen und Musikinstrumenten kann ein Tabletop auch für wissenschaftliche Analysen eingesetzt werden. Das Projekt „G-Nome Surfer“ [43] ist eine Tabletop-Anwendung zur kollaborativen Exploration von Gen-Daten in der Genetik. Es ist ein Mehrbenutzersystem und ermöglicht es gemeinsam Gen-Daten zu analysieren. Das Ziel ist es, eine kreative und attraktive Lernmethode im Bereich der Genforschung anzubieten. Das System hat ein Natural User Interface auf Basis des Microsoft Surface SDKs. Es bietet verschiedene Repräsentationen der Daten an und es können verschiedene Ansichten auf die Daten verglichen werden. Die Studie über die Software hat ergeben, dass die Studenten im Gegensatz zu alternativer Software entlastet werden und mehr Spaß beim Lernen hätten. (siehe Abbildung 21)

Die Software Phylo-Genie [44] ist ebenfalls eine Lernsoftware, die es Studenten ermöglichen soll das Verfahren der „Tree-Thinking“ zu erlernen. Es wurde entwickelt, da Studenten Schwierigkeiten haben Phylogenetik zu verstehen. Am Ende wurde eine Studie durchgeführt, die den Einsatz von einem Tabletop mit dem Einsatz von einfachem Schreibmaterial vergleicht. Auch hier wurde entdeckt, dass kollaborative Arbeit das Lernverhalten verbessert.

Abbildung 21: Das linke Bild zeigt zwei Personen beim Arbeiten mit G-Nome Surfer. Auf dem rechten Bild sieht man eine Ansicht von Phylo-Genie.

4 Aufgabenbeschreibung

Das Ziel dieser Arbeit ist es, ein Interaktionskonzept zur Steuerung von Powerwall-Visualisierungen mit Tabletop-PCs zu entwickeln. In diesem Kapitel wird die Aufgabenbeschreibung wiedergegeben und mit der Vorgehensweise des Lösungsansatzes erweitert.

4.1 Hintergrund

Der Hintergrund dieser Arbeit begründet sich auf der Auswertung von Eyetracking-Studien. Solche werden unter anderem am Institut für Visualisierung und Interaktive Systeme (VIS) der Universität Stuttgart durchgeführt.

Für Studien dieser Art wurde an diesem Institut eine Diplomarbeit durchgeführt, um neue Visualisierung für die Analyse von Eyetracking-Ergebnissen zu entwickeln. Diese Diplomarbeit heißt „Visuelle Analyse von Eye-Tracking-Daten“ [45] und brachte eine neue Art von Visualisierungen mit drei Ausprägungen hervor.

Diese neuen Visualisierungen wurden bei der Analyse der Eyetracking-Studie „Visual Elements“ eingesetzt, die ebenfalls am Institut VIS der Universität Stuttgart durchgeführt wurde. Bei der Auswertung der Studie mit den neuen Visualisierungen wurde schnell erkannt, dass normale Desktopdisplays zu klein sind, vor allem bei einer Auswertung in der Gruppe. Darum wurde als nächstes eine Powerwall eingesetzt. Allerdings stellte sich dort die Bedienung der entsprechenden Software als schwierig bzw. aufwendig heraus. Denn Maus und Tastatur konnten nur auf dem angeschlossenen Steuerrechner genutzt werden. Aus diesem Grund musste sich für jede Änderung eine Person zu dem Steuerrechner begeben. Die Nutzung der Powerwall hat sich jedoch sonst als sehr vorteilhaft herausgestellt, das motivierte die Idee, die dieser Arbeit zugrunde liegt.

4.2 Aufgabenstellung

Im Rahmen dieser Arbeit soll ein Interaktionskonzept für Tabletop-Computer innerhalb einer multimodalen Analyseumgebung für Powerwalls entwickelt werden. Die Analyseumgebung soll verschiedenste Visualisierungen in den Gebieten der Informations-Visualisierung und der wissenschaftlichen Visualisierung unterstützen. Das Interaktionskonzept soll sinnvolle Interaktion mit der Powerwall ermöglichen. Dabei soll sowohl eine Interaktion von einzelnen Benutzern als auch die Interaktion in einer Gruppe möglich sein. Das wird unter anderem per Interaktion mit Realweltobjekt ermöglicht, die durch entsprechende Tabletops erkannt werden können. Im Rahmen dessen soll ein Natural User Interface und Tangible User Interface entwickelt werden, das eine möglichst schnelle und intuitive Bedienung der Visualisierung und der Analyseumgebung zulässt.

Die Analyseumgebung entsteht in Kooperation mit der Diplomarbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ [46], die ein Interaktionskonzept für Freihandgesten innerhalb der gleichen Analyseumgebung entwickelt.

Der Prototyp der Analyseumgebung soll übliche Visualisierungen im Rahmen von Eyetrackingdaten anzeigen. Außerdem sollen Szenarien wie interaktive Analysen von Eyetrackingdaten als Einzelner oder in Gruppen möglich sein. Dafür sollte eine Anforderungsanalyse für Eyetrackingstudien mit interaktiven Tools durchgeführt werden und auf den Ergebnissen Interaktions- und Visualisierungskonzept entworfen werden. Basierend

auf den Konzepten sollte der Prototyp mit „Microsoft Pixelsense“-Steuerung der Analyseumgebung und den entsprechenden Visualisierungen entwickelt werden.

Weiterhin soll ein zweites Szenario für die Analyse wissenschaftlicher Visualisierungen mit einer multimodalen Analyseumgebung entwickelt werden.

Zur Evaluierung des Interaktionskonzepts sollte eine Studie durchgeführt werden.

4.3 Lösungsüberblick

Ausgehend von der Aufgabenstellung wird in diesem Kapitel ein Überblick über die Vorgehensweise zur Lösung gegeben.

1. **Beobachtung der Auswertung der Eyetracking-Studie**
   Um einen Eindruck für eine Auswertung einer Eyetracking-Studie zu bekommen wird eine Analysesitzung beobachtet. Dabei werden im Gespräch mit den Analysten Probleme und Anforderungen an unterstützende Tools ermittelt.
   Die Beobachtungen werden in Kapitel 5.1 wiedergegeben.

2. **Entwurf der Szenarien**
   Anhand der Beobachtung und den Gesprächen mit den Verantwortlichen der Eyetrackingstudie wird ein Szenario für die Analyse von Eytracking-Studien entworfen. Nach der Fertigstellung des Szenarios wird es mit den Analysten besprochen und angepasst. Das zweite Szenario wird anhand einer Anwendung für die Visualisierung von Molekülen entwickelt.
   Die Szenarien werden in Kapitel 5.2 und 5.3 vorgestellt.

3. **Anforderungsanalyse**
   Basierend auf den Beobachtungen und den Szenarien werden Anforderungen sowohl für die Analyseumgebung, als auch die Tabletop-Steuerung erhoben.
   Die Anforderungen sind in Kapitel 5.4 beschrieben.

4. **Entwicklung des System- und Interaktionskonzepts**
   Auf Basis der Anforderungen wird ein abstraktes Konzept für das System der multimodalen Analyseumgebung erstellt. Danach wird aufbauend auf dieses Konzept ein abstraktes Interaktions- und Oberflächenkonzept für den Tabletop innerhalb der Analyseumgebung aufgestellt.
   Das abstrakte Konzept wird in Kapitel 6 aufgeführt.

5. **Durchführung der Vorstudie und Entwicklung des Lösungsansatzes**
   Das abstrakte Konzept des Tabletops wird in diesem Kapitel um konkrete Interaktionen und Oberflächenentwürfe erweitert. Dies wird darauf in einer Vorstudie evaluiert. Der Vorstudie liegt das Prinzip des „Paperprototyping“ [47] zu Grunde, wobei es durch Komponenten des „User Defined Design“ [48] erweitert wurde. Anhand der Ergebnisse der Vorstudie werden die Entwürfe erweitert. Daraus entsteht der schlussendliche Lösungsansatz.
   Die Vorüberlegungen und Entwürfe, die vor der Studie entstanden sind, sind in Kapitel 7.1 beschrieben. Die Vorstudie und ihre Ergebnisse sind in Kapitel 7.2 zu finden. Eine Erweiterung des Lösungsansatzes anhand der Ergebnisse befindet sich in Kapitel 7.3.

6. **Entwurf und Implementierung der multimodalen Analyseumgebung**
   In Zusammenarbeit mit der Diplomarbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ [46] wird das Konzept für die multimodale Analyseumgebung abgestimmt. Anschließend wird ein Kommunikationsframework anhand des Konzepts entwickelt, das den Einsatz mehrerer Geräte ermöglicht, um Visualisierungen an der Powerwall anzuzeigen und zu steuern. Dieses Framework bildet die Grundlage der multimodalen Analyseumgebung.
   Die Architektur und Implementierungsentscheidungen des Frameworks werden in Kapitel 8 beschrieben.

7. **Implementierung des Prototyps**
   Der Prototyp für die Tabletop-Steuerung der Powerwallvisualisierungen wird auf Grundlage des Kommunikationsframeworks für multimodale Analyseumgebungen implementiert. Das Framework wird benötigt, um mit einem zweiten Client auf der Powerwall zu kommunizieren. Ein solche Client entsteht in der Arbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“. Der Prototyp wird für einen Surface 2 auf Basis des Surface SDK 2.0 entwickelt. Als Vorlage für die Implementierung wird der Lösungsansatz verwendet, der unter Punkt 5 entstanden ist. Um eine tatsächliche Funktion zu erfüllen, müssen ein Datenmodell für Eyetracking und zugehörige Visualisierungen entstehen. Diese werden ebenfalls in Zusammenarbeit mit der Arbeit für Gestensteuerung entwickelt.
   Die Implementierung des Prototyps wird in Kapitel 9 vorgestellt.

5 Anforderungsanalyse

Das folgende Kapitel untersucht die Anforderungen, die an eine multimodale Analyseumgebung und eine zugehörige Tabletop-Steuerung gestellt werden. Dies wird im speziellen anhand der Anwendung auf Eyetracking-Studien erforscht. Dafür wird die Auswertung einer Eyetracking-Studie beobachtet. Auf Basis der Beobachtungen wird ein Szenario für die Auswertung von Studien, in Zusammenarbeit mit den Verantwortlichen der Eyetracking-Studie, entwickelt. Ein zweites Szenario für wissenschaftliche Visualisierungen wird anhand einer Anwendung zur Visualisierung von Molekülen entworfen. Die Beobachtungen und Szenarien sind Grundlage für die dann folgende Untersuchung der Anforderungen.

Die Anforderungen und Szenarien wurden in Zusammenarbeit mit der Diplomarbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ [46] erarbeitet.

5.1 Beobachtungen aus der Auswertungssitzung der Eyetracking-Studie

Dieses Kapitel stellt die Beobachtungen der Auswertungssitzung der Eyetracking-Studie vor. Vorher wird aber die Studie „Visual Elements“ vorgestellt und die Visualisierungen und Tools, die bei der Auswertung genutzt wurden.

5.1.1 „Visual Elements“-Studie

Beim Eyetracking werden die Augenbewegungen einer Person aufgenommen während sie Bilder oder Videos, genannt Stimuli, anschaut. Dies geschieht mit einem Eyetracker. In den meisten Fällen werden dabei die Augen mit Kameras überwacht.

Die Eyetrackingstudie, die zur Beobachtung genutzt wurde, heißt „Visual Elements“. Die Studie wurde beim Institut für Visualisierung und Interaktive Systeme der Universität Stuttgart durchgeführt. In der Studie sollte herausgefunden werden, ob es allgemeine Strategien für die Augenbewegung beim Erkennen von Informationen aus Visualisierungen gibt. Die Studie wurde mit 30 Probanden durchgeführt. Die Probanden mussten drei Aufgabentypen lösen: Die Koordinaten eines Punktes in einem 1, 2 oder 3D-Koordinatensystem auslesen, die größte Fläche in einem Bild bestimmen und Dreiecke auf Ähnlichkeit überprüfen. Für alle Aufgaben haben die Probanden eine Einführung bekommen.

Aufgabe 1: Koordinaten auslesen
Bei dieser Aufgabe bekommt der Proband ein Koordinatensystem mit entweder ein, zwei oder drei Dimensionen. Innerhalb der Koordinatensysteme sind Punkte, deren Koordinaten der Proband auslesen soll.

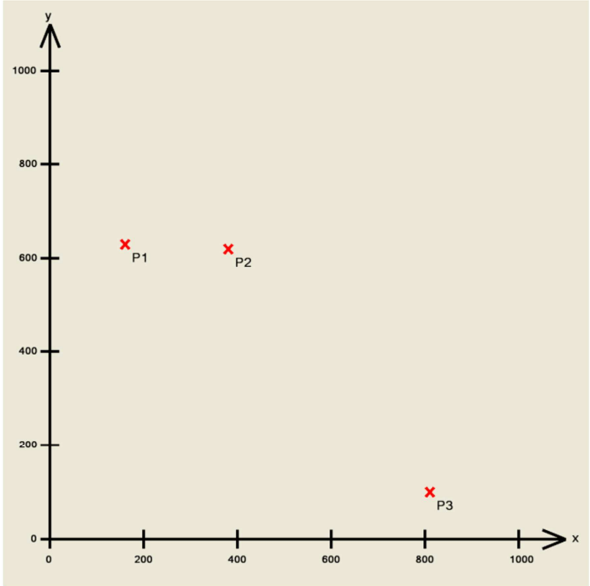


**Abbildung 22: Stimulus für das Auslesen von 2D-Koordinatensystemen.
Die Punkte P1, P2 und P3 mussten abgelesen werden.**

Aufgabe 2: Größte Fläche bestimmen
Bei dieser Aufgabe bekommen die Probanden verschiedene Bilder mit bunten Flächen. Dabei kann eine Farbe auch mehrmals vorkommen. Der Proband soll nun die Farbe mit der größten Fläche bestimmen.

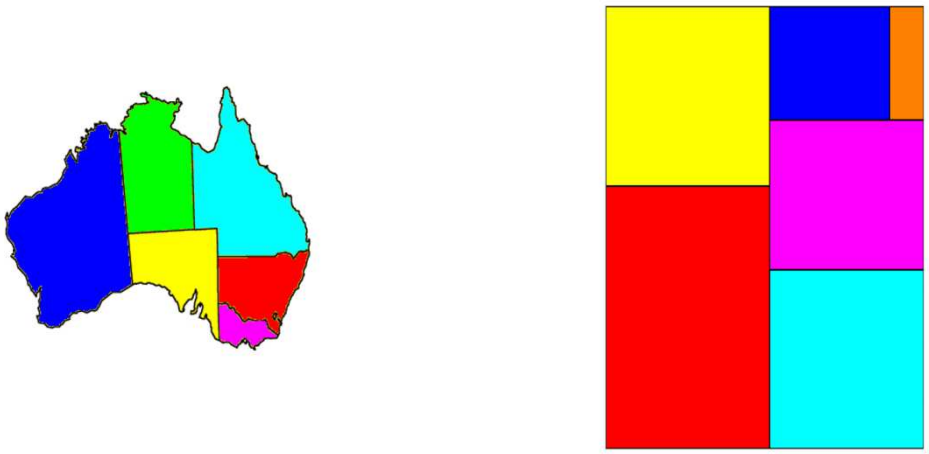


**Abbildung 23: Stimuli für das Bestimmen der größten Fläche.
Für beide Stimuli sollte die Farbe der größten Fläche bestimmt werden.**

Aufgabe 3: Ähnlichkeit von Dreiecken bestimmen
Der Proband erhält bei dieser Aufgabe Bilder mit zwei Dreiecken, die er auf geometrische Ähnlichkeit prüfen soll. Welche Kriterien dabei eine Rolle spielen wird dem Proband vorher mitgeteilt.

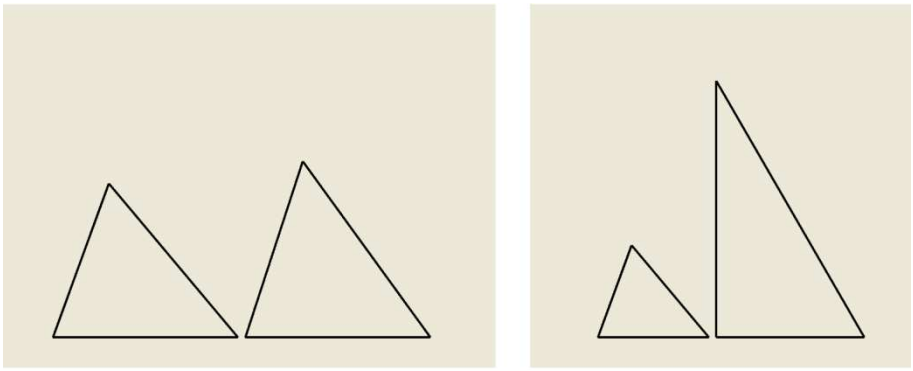


Abbildung 24: Stimuli für den Vergleich von Dreiecken.
Für beide Dreiecke sollte geprüft werden ob sie geometrisch ähnlich sind.

5.1.2 Auswertungswerkzeuge

Bei der Auswertung wurden wiederum verschiedenste Visualisierungstechniken genutzt. Dazu gehören die klassischen Methoden wie Heatmap- und Scanpath-Visualisierung als auch neu entwickelte Visualisierungen.

Heatmap-Visualisierung

Bei einer Heatmap werden die Punkte, die der Proband mit seinen Augen anvisiert hat, genannt Fixationen auf dem Bild aufsummiert. Danach werden den Werten Farben zugeordnet. Der Name Heatmap kommt dabei davon, dass normalerweise für kleine Werte kalte Farben und für höhere Werte wärmere Farben eingesetzt werden.

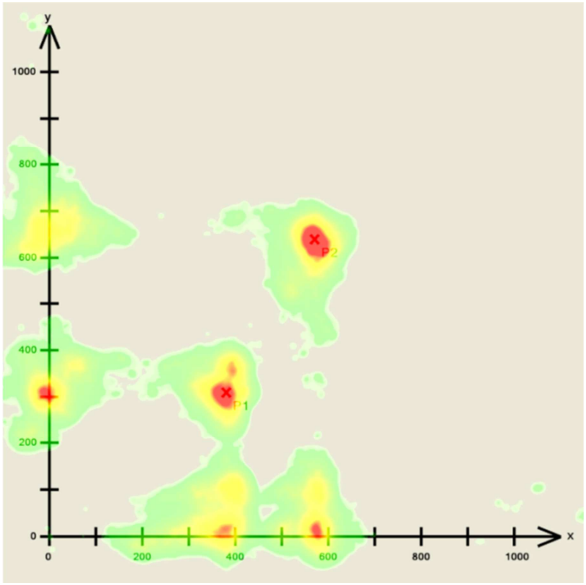


Abbildung 25: Heatmap-Visualisierung eines Stimulus aus Aufgabe 1. Die Farbskala reicht von Rot nach Grün. Rote Bereiche wurden häufig fixiert, grüne Bereiche dagegen selten.

Scan-Path-Visualisierung

Die Scan-Path-Visualisierung zeigt den Verlauf der Augenbewegungen auf dem Bild. Dabei werden alle Fixationen mit Linien verbunden, wie sie vom Probanden nacheinander angeschaut worden sind. Damit lässt sich Reihenfolge von Punkten bestimmen die ein Proband angeschaut hat.

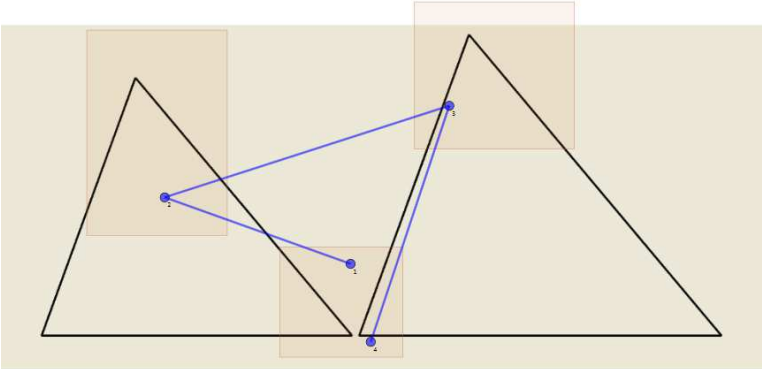


Abbildung 26: Scan-Path-Visualisierung eines Stimulus aus Aufgabe 3. Die Abfolge der Fixationen wird durch die blaue Linie dargestellt.

Parallel Scan-Path Visualisierungen

In einer vor dieser Arbeit durchgeführten Diplomarbeit, mit dem Namen „Visuelle Analyse von Eye-Tracking-Daten“ [45], sind die sogenannten Parallel Scan-Path Visualisierungen entstanden. Von diesen Visualisierungen gibt es drei verschiedenen Arten, Gaze Duration Sequence Diagramme, Fixation Point Diagramme und Gaze Duration Distribution Diagramme. Alle Diagramme beziehen sich auf sogenannte Areas of Interest (AOIs). Diese Areas of Interest sind bestimmte Bereiche auf dem Stimulus. Die AOIs müssen erst definiert werden, bevor eine solche Visualisierung generiert werden kann. In den Diagrammen wird für jede AOI ein Koordinatenkreuz angelegt. Die vertikale Koordinatenachse beschreibt die Zeit und zeigt somit an, dass nach oben die Zeit innerhalb der Eyetracking-Messung zunimmt. Die horizontale Koordinatenachse gibt an, dass nach rechts alle AOI-Achsen eingetragen sind. Die Diagramtypen unterscheiden sich dann folgendermaßen:

- Gaze Duration Sequenz Diagramm
  Dieses Diagramm zeigt die Dauer von Blicken in die jeweiligen AOIs. Außerdem ist die Reihenfolge in der zwischen den AOIs gewechselt wurde zu erkennen. Eine Linie beschreibt das Verhalten eines Probanden. Die Linie fängt unten in einer AOI an. Vertikale Linien auf einer AOI-Achse beschreibt die Dauer, in der der Proband in eine AOI geschaut hat. Eine horizontale Linie zeigt einen Wechsel zwischen zwei AOIs an.

- Fixation Point Diagramm
  Auch hier werden die Wechsel zwischen den verschiedenen AOIs dargestellt. Allerdings wird hier nicht das Ende und der Anfang als Verbindungspunkte zwischen den Koordinatenachse verwendet, sondern der Mittelpunkt zwischen den beiden. Um diesen Mittelpunkt herum werden die einzelnen Fixationen innerhalb der AOI im zeitlichen Verlauf durch Punkte auf der Koordinatenachse markiert.

- Gaze Duration Distribution Diagramm
  Bei diesem Diagrammtyp werden ebenfalls die Wechsel der Blicke zwischen den AOIs dargestellt. Zusätzlich wird die Summe der Verweildauer aller Blicke als Balken dargestellt.

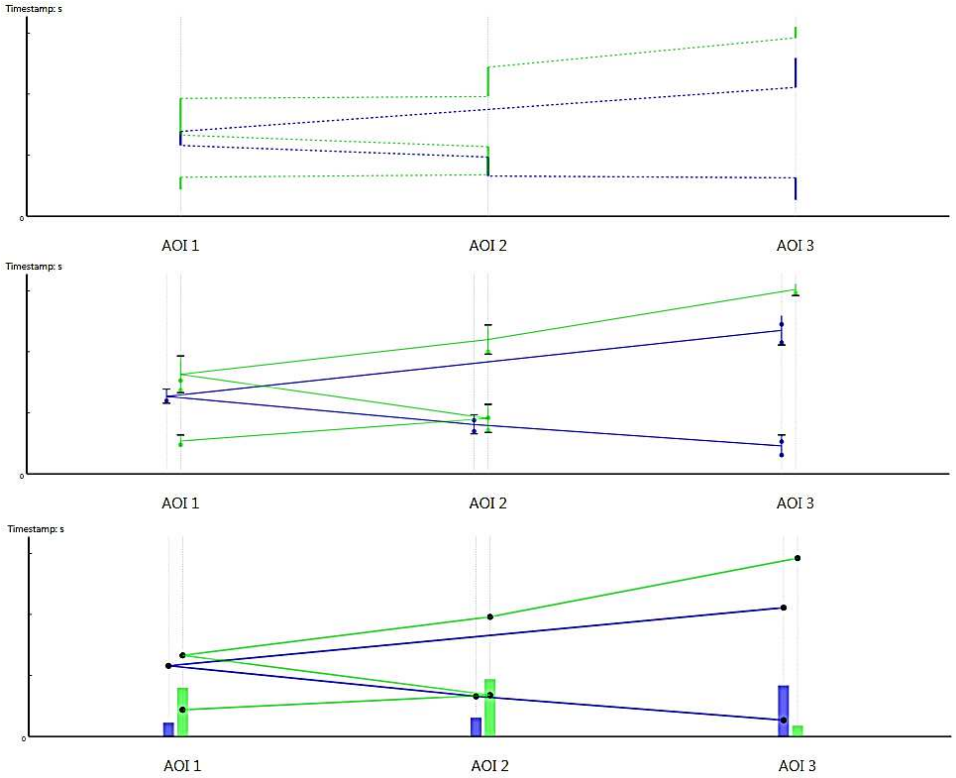


Abbildung 27: Parallel-Scan-Paths zweier Probanden für einen Stimulus mit drei AOIs
oben: Gaze Duration Sequenz Diagramm
mitte: Fixation Point Diagramm
unten: Gaze Duration Distribution Diagramm

5.1.3 Beobachtungen bei der Auswertung

Bevor die Auswertung der Studie begonnen werden konnte, mussten die Daten in die verschieden Programme eingelesen werden, die die Werkzeuge zur Auswertung der Studie bereitstellen. Dazu mussten die Daten teilweise aufwendig von Hand konvertiert werden. Die ursprünglichen Daten liegen in dem Format des Eyetrackers Tobii vor. Daraus mussten die Daten in die Datenbank des Parallel-Scan-Path-Tools eingelesen werden. Zusätzlich wurde auch eine statistische Analyse durchgeführt, bei der die Daten noch einmal extra eingelesen werden mussten.

Als erstes wurde versucht, die Auswertung mit Desktop-Bildschirmen und Ausdrucken durchzuführen. Dabei wurden für einen Probanden mehrere Visualisierungen geöffnet. Dazu gehören ein Bild des Stimulus und Visualisierungen wie Heatmaps um die interessanten Stellen im Bild zu finden. Danach werden für die Visualisierungen, die das benötigen AOIs definiert und im Anschluss werden diese Visualisierungen auch geöffnet. Um nun die Ergebnisse bei mehreren Probanden zu vergleichen werden für diese ebenfalls weitere Visualisierungen geöffnet.

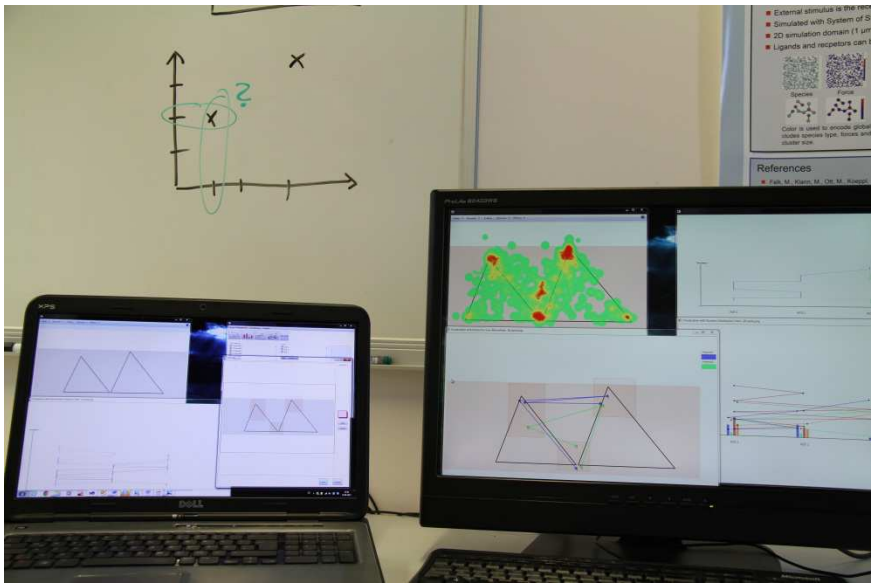


Abbildung 28: Zeigt einen mögliches Bild einer Eyetracking-Auswertung an Desktop-Monitoren.

Um die Visualisierungen alle darzustellen und für verschiedene Probanden zu vergleichen, reichen normale Bildschirme schnell nicht mehr aus. Des Weiteren findet eine solche Auswertung oft in Gruppen statt, was ein Arbeiten an einzelnen Bildschirmen weiter erschwert. Im Fall der genannten Studie wurde entschieden die Analyse an einer Powerwall durchzuführen.

Diese spezielle Powerwall besteht aus einer Projektionsfläche auf die über Projektion von hinten das Bild über vier Beamer erzeugt wird. Diese Beamer haben eine Full-HD Auflösung und um Stereoskopie zu ermöglichen, projizieren jeweils zwei Beamer eine Fläche. Die Trennung der Bilder geschieht über Polarisationsfilter. Zusammen ergibt sich dadurch eine Fläche mit Full HD Auflösung. Die Rückprojektion wird daher verwendet, weil dadurch Personen vor der Projektionsfläche stehen können ohne das Bild zu verdecken. Die Beamer werden von einem einzelnen PC angesteuert. Mit diesem kann man eine normale Bildschirmausgabe auf der Powerwall anzeigen.

Mit der Powerwall und einigen Tools, die die genannten Visualisierungen erzeugen, wurde erneut versucht die Eyetracking-Daten zu analysieren. Die große Fläche der Powerwall ermöglichte es viele Visualisierungen gleichzeitig anzuzeigen. Daher ist das Problem der kleinen Anzeigefläche behoben. Jedoch bleibt das Problem der geringen Auflösung, wenn sonst Full-HD-Bildschirme eingesetzt wurden. Die Interaktion mit der Powerwall ist dagegen schwieriger. Im Laufe der Analyse wurde es oft nötig, einzelne Visualisierungen zu verschieben oder die Datenmenge zu ändern. Normalerweise sind die Analysten jedoch vor der Powerwall und diskutieren über die Ergebnisse. Eine Interaktion über den Steuer-PC durchzuführen führt immer zu einer Unterbrechung des Analyseflusses. Zusätzlich gibt es für die bedienende Person einen Aufmerksamkeitssprung zwischen dem Bildschirm des PCs und der Powerwall.

5.2 Eyetracking-Szenario

Das erste Szenario bezieht sich auf die Auswertung von Eyetracking-Studien und ist damit eine Anwendung im Bereich der Informationsvisualisierung. Es wird auf Grundlage der Beobachtungen entworfen. Der Entwurf wurde daraufhin in Zusammenarbeit mit Verantwortlichen der „Visual Elements“-Studie besprochen und weiter verbessert. Im

Szenario wird eine mögliche Auswertungssitzung skizziert, in der Analysten Eyetracking-Studien mit einer Analyseumgebung auswerten können. Alle Interaktionen beziehen sich dabei auf den Tabletop. Die Ansicht der Powerwall entspricht in dem Szenario immer der des Tabletops. Alle Skizzen zeigen links die Powerwall und rechts den Tabletop.

Offline-Vorbereitung der Eyetracking-Analyse
Der Moderator einer Eyetracking-Studie bereitet die Daten für die Auswertung einer Studie an seinem Laptop vor. Bevor er weiterarbeiten kann, müssen die Daten in die Analyseumgebung importiert werden. Nach dem Import wählt der Moderator drei Aufgaben aus, die als Grundlage der Besprechung dienen sollen.

Abbildung 29: Offline-Vorbereitung
Der Moderator importiert Aufgaben in das System.

Kurz vor Beginn der Auswertungssitzung
Der Moderator begibt sich vor Beginn der Sitzung in den Besprechungsraum um die Sitzung vorzubereiten.

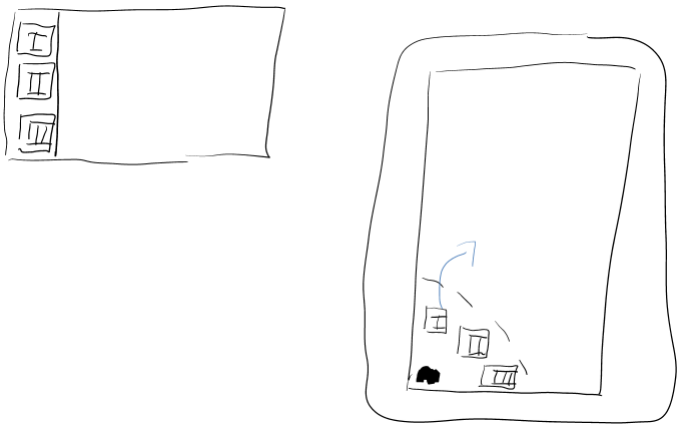


Abbildung 30: Vorbereitung der Auswertungssitzung
Der Moderator fügt die Aufgaben zur Analyseumgebung hinzu.

Er stellt seinen Laptop oder legt eine Id-Karte auf den Tabletop. Der Laptop zu der Id-Karte muss sich im gleichen Netz befinden wie die Analyseumgebung. Über ein Tag an der Unterseite, in dem IP-Adresse, Port und Identifikation kodiert sind, wird der Laptop erkannt und in die Analyseumgebung eingebunden. Ein Feld wird auf dem Tabletop angezeigt. In dem Feld befinden sich die drei vorher ausgewählten Aufgaben. Mit einer Drag&Drop-Interaktion schiebt der Moderator die Aufgaben auf eine freie Fläche des Tabletops. Daraufhin werden die Aufgaben im Datenexplorer aller Geräte innerhalb der Analyseumgebung angezeigt.

Durchführung der Besprechung

Nachdem sich alle Teilnehmer der Besprechung im Besprechungsraum eingefunden haben, beginnt der Moderator die Analysesitzung. Er zieht mit einer weiteren Drag&Drop-Interaktion die erste Aufgabe aus dem Datenexplorer auf eine freie Fläche des Tabletops. Anschließend wird an dieser Stelle das Bild der Aufgabenstellung angezeigt. Gleichzeitig wird es an der entsprechenden Stelle der Powerwall angezeigt. Mit einer Geste justiert ein Teilnehmer die Größe des Bildes, so dass alle anderen Teilnehmer eine einwandfreie Sicht auf das Bild haben. Anhand dieses Bildes erklärt der Moderator den Teilnehmern die Aufgabenstellung für die Probanden und seine These für die Auswertung.

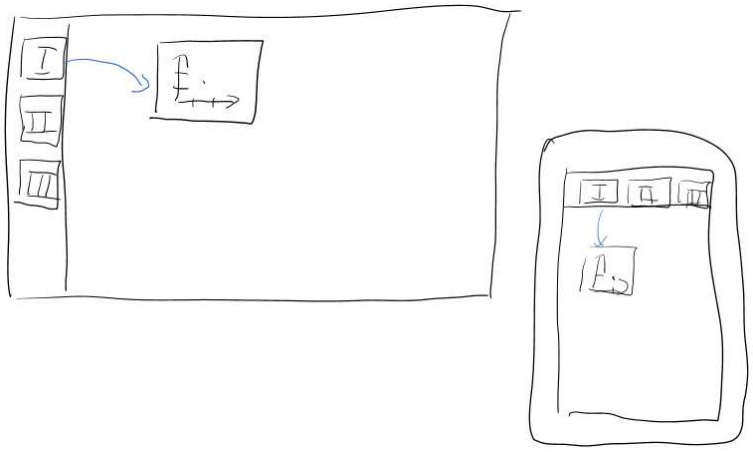


Abbildung 31: Start der Sitzung
Der Moderator erzeugt die erste Visualisierung.

Damit während der gesamten Analyse das Bild der Aufgabenstellung sichtbar bleibt, dupliziert der Moderator mit einer Interaktion diese Ansicht. Nun werden zwei identische Aufgabenbilder dargestellt.

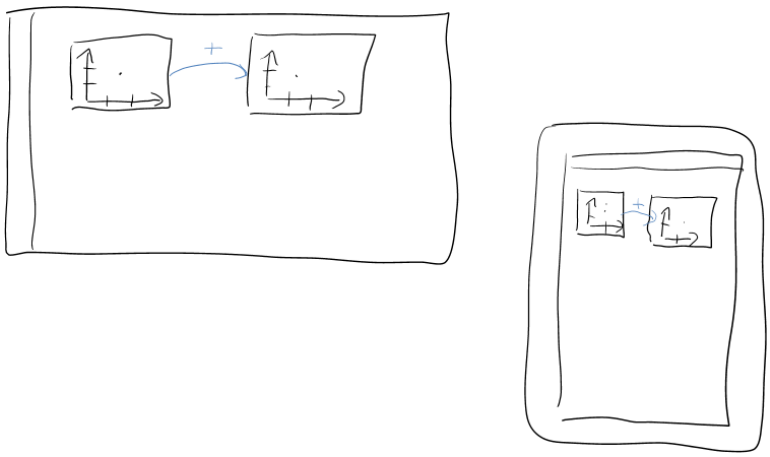


Abbildung 32: Duplikation einer Visualisierung

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

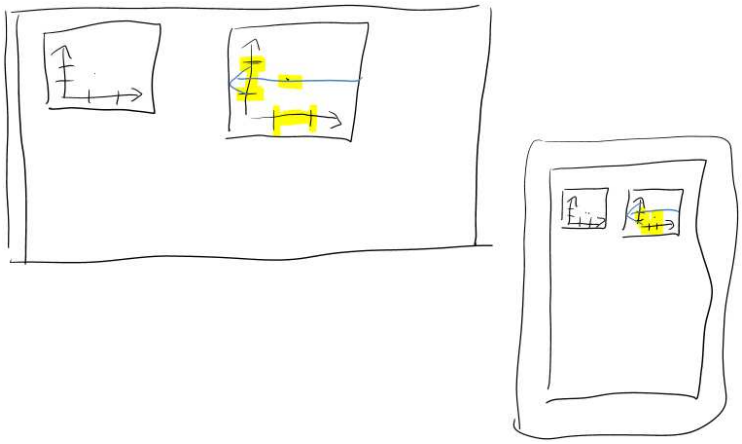


Abbildung 33: Ändern des Visualisierungstyps
Ein Nutzer wechselt die Visualisierung mit einer Wischbewegung.

Auf Basis der HeatMap-Visualisierung diskutieren die Teilnehmer, welche Bereiche des Bildes wahrscheinlich für die Probanden interessant waren und nun näher untersucht werden sollen. Ist ein Proband ausgewählt, werden dessen Blickpunkte in der HeatMap hervorgehoben. So können die Bereiche betrachtet werden, die von einzelnen Probanden angesehen wurden. Darauf werden von den Teilnehmern mehrere AOIs durch eine Interaktion, die eine Form bildet, definiert. Die AOIs werden sowohl in der HeatMap-Visualisierung, als auch in der Ansicht mit dem Bild der Aufgabenstellung angezeigt.

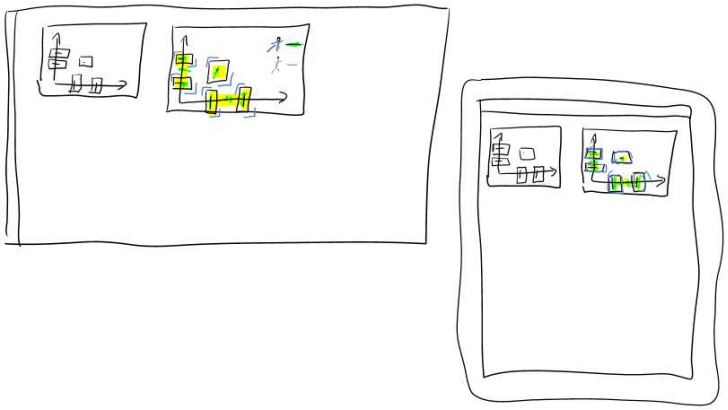


Abbildung 34: AOIs erstellen

Mit einer Wischbewegung nach links über die HeatMap-Ansicht wechselt diese zu einer Gaze-Duration-Visualisierung, in dem alle Probanden angezeigt werden. Der Moderator arrangiert daraufhin die Reihenfolge der vertikalen Koordinatenachsen wie er sie benötigt.

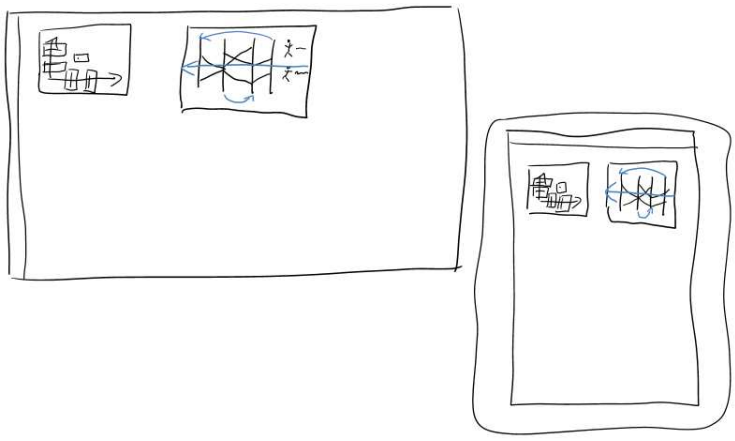


Abbildung 35: Verändern des Inhalts einer Visualisierung
Ein Nutzer ändert die Reihenfolge der Achsen in einer Parallel-Scan-Path-Visualisierung.

Nach einiger Diskussion der Teilnehmer fällt auf dem Gaze-Duration-Diagramm ein Proband auf, der offensichtlich nicht dem Verhalten der anderen Teilnehmer entspricht. Mit einer Interaktion markiert ein Teilnehmer diesen Probanden und entfernt ihn mit einer Geste aus der Visualisierung. Alternativ öffnet ein Teilnehmer die Filterdarstellung auf dem Tabletop und entfernt das dem Probanden zugeordnete Objekt.

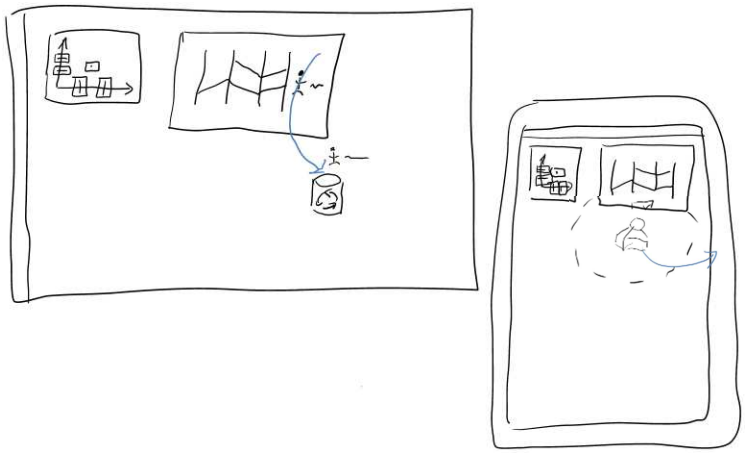


Abbildung 36: Entfernen eines Probanden aus einer Visualisierung

Im weiteren Verlauf fällt auf, dass sich die Probanden auf der Gaze-Duration-Visualisierung in zwei verschiedene Gruppen aufteilen. Der Moderator wählt einen der Probanden der ersten Gruppe aus und zieht ihn mit einer Drag&Drop-Interaktion auf eine freie Fläche des Tabletops. Dabei wird eine neue Gaze-Duration-Visualisierung an dieser Stelle erstellt, die ausschließlich die Darstellung des Probanden, der soeben aus dem ersten Diagramm herausgezogen wurde enthält. Daraufhin öffnet ein Teilnehmer die Filterdarstellung beider Visualisierungen und verschiebt alle zu der Gruppe zugehörigen Probanden mit ihren Tangibles in die Filterdarstellung der neuen Visualisierung.

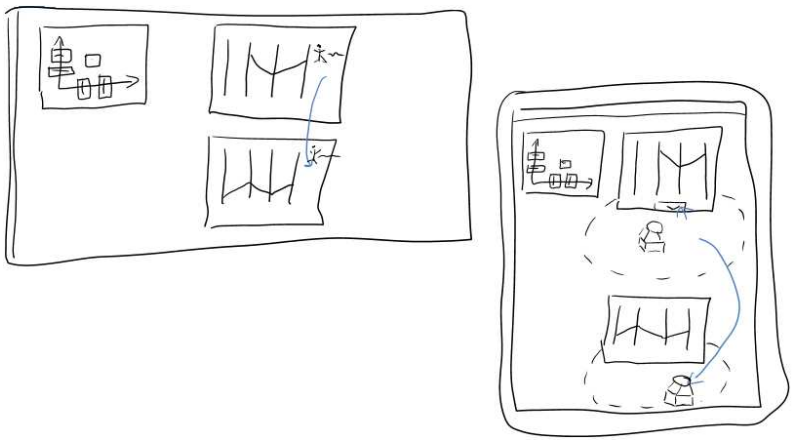


Abbildung 37: Filtern der Daten
Ein Nutzer teilt die Probanden auf zwei Visualisierungen auf.

Der Moderator möchte dieses Ergebnis in Form eines Screenshots festhalten, um es später weiterverwenden zu können. Mit einer Geste erstellt er, aus den beiden Gaze-Duration-Visualisierungen, ein Screenshot und überträgt diesen mit der Id-Karte auf sein Notebook.

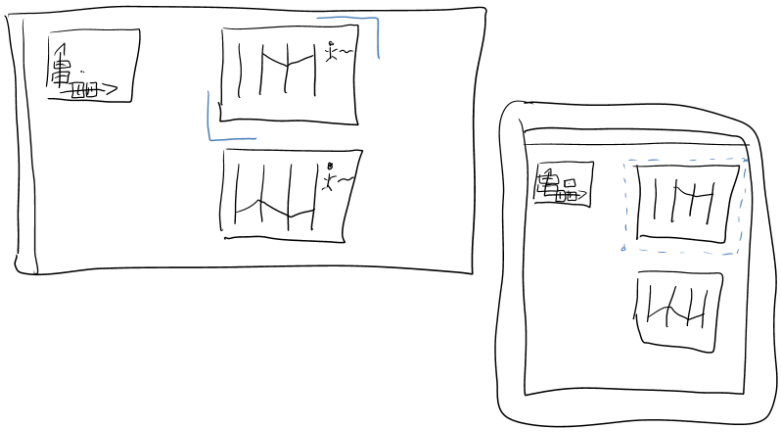


Abbildung 38: Erzeugen eines Screenshots von einer Visualisierung

Zuletzt soll überprüft werden, ob sich die Probanden der beiden Gruppen in einer anderen Aufgabe vergleichbar verhalten haben. Der Moderator führt auf eine der beiden Gaze-Duration-Visualisierungen eine Wischgeste nach unten aus. Dabei wechseln sich die Aufgaben der Visualisierungen und damit ihre zugrundeliegenden Daten.

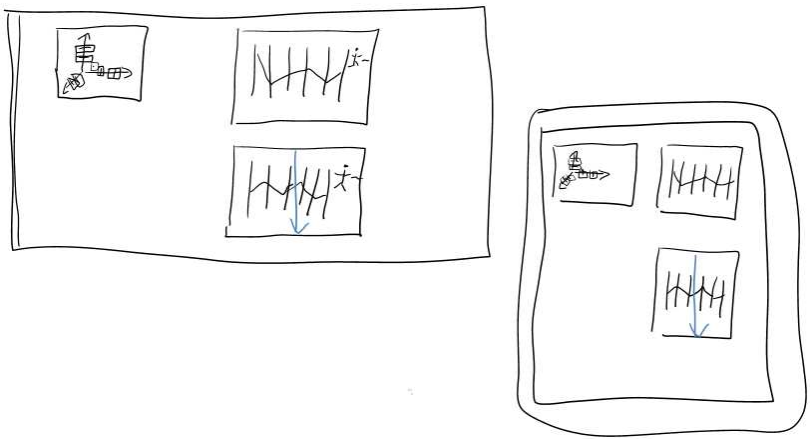


Abbildung 39: Wechseln der Aufgabe in den Visualisierungen

Beendigung oder Unterbrechung der Auswertungssitzung
Die Besprechung ist beendet oder soll unterbrochen werden. Um die Analyse zu einem späteren Zeitpunkt fortzusetzen, oder die abschließenden Ergebnisse bei Beendigung der Analyse festzuhalten, speichert der Moderator den Zustand der Analyseumgebung.

5.3 MegaMol-Szenario

Das zweite Szenario bezieht sich auf die Visualisierung von Simulationsdaten aus Molekülmodellen und ist damit aus dem Bereich der wissenschaftlichen Visualisierung. Das Szenario baut auf dem Programm MegaMol [3] auf und wurde auf Grundlage der Funktionen des Programms erstellt.

Mit der Visualisierungssoftware MegaMol [3] können Moleküle aus punktbasierte Datensets visualisiert werden. Bei dem Szenario möchte ein Wissenschaftler seine Erkenntnisse bei der Erforschung von Proteinen seinen Kollegen weitergeben. Dazu verwendet er die Analyseumgebung um Visualisierungen von MegaMol zu präsentieren.

Alle Interaktionen in dem Szenario beziehen sich auf den Tabletop. Die Ansicht der Powerwall entspricht in dem Szenario immer der des Tabletops. Alle Skizzen zeigen links die Powerwall und rechts den Tabletop.

Offline-Vorbereitung
Um seine Präsentation vorzubereiten, wählt der wissenschaftliche Mitarbeiter drei Proteine aus, mit denen er seine Erkenntnisse vorstellen möchte. Mit seinem Laptop importiert der Mitarbeiter die Simulationsdaten für die Proteine in die Analyseumgebung. Für die importierten Proteine werden Vorschaubilder einer Visualisierung im Datenexplorer des Laptops angezeigt.

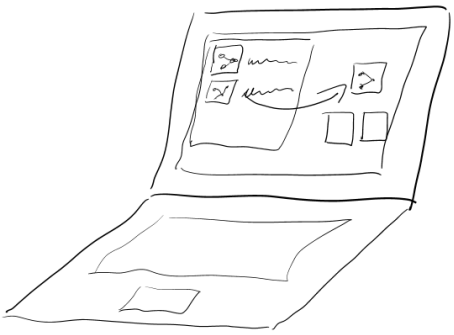


Abbildung 40: Offline Vorbereitung
Der wissenschaftliche Mitarbeiter importiert Proteindaten ins System.

Kurz vor Beginn der Präsentation
Der wissenschaftliche Mitarbeiter begibt sich in den Raum mit der Powerwall und dem Tabletop. Daraufhin legt er eine ID-Karte für den Laptop oder den Laptop auf den Tabletop. Der Laptop wird von der Analyseumgebung erkannt und in der Analyseumgebung registriert. Auf dem Tabletop erscheint ein Dialog, der den privaten Bereich des Laptops anzeigt. Innerhalb des Dialogs werden die Proteine mit ihren Vorschaubildern angezeigt. Der wissenschaftliche Mitarbeiter zieht die Proteine mit einer Drag&Drop-Interaktion auf einen Bereich des Tabletops.

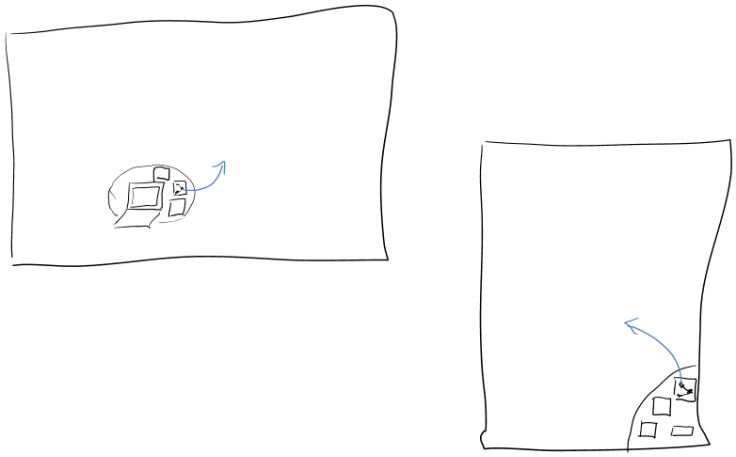


Abbildung 41: Vorbereitung der Präsentation
Der wissenschaftliche Mitarbeiter fügt Proteine zur Analyseumgebung hinzu.

Anschließend werden die Protein-Daten auf die Geräte der multimodalen Analyseumgebung kopiert und in den Daten-Explorern der Geräte angezeigt.

Während der Präsentation
Während seiner Präsentation möchte der Mitarbeiter nun seine Forschungsergebnisse anhand eines Proteins seinen Kollegen präsentieren. Mit einer Drag&Drop-Interaktion zieht der Mitarbeiter das Vorschaubild des entsprechenden Proteins aus dem Daten-Explorer eine freie Fläche des Tabletops. Daraufhin wird dort eine Visualisierung von MegaMol angezeigt, die das Protein darstellt. Gleichzeitig wird diese Visualisierung auch auf der Powerwall angezeigt.

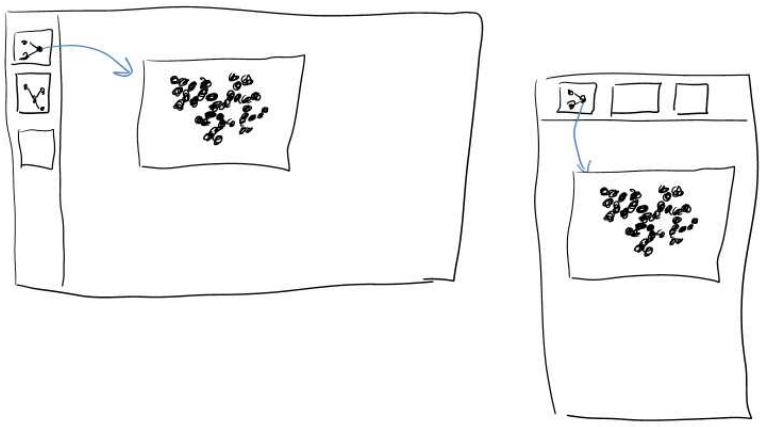


Abbildung 42: Start der Sitzung
Der wissenschaftliche Mitarbeiter erzeugt die MegaMol-Visualisierung.

Der wissenschaftliche Mitarbeiter möchte während seiner Präsentation die Visualisierung des Proteins öfter ändern. Daher benötigt er eine Ausgangsdarstellung, die es seinen Kollegen erlaubt den Überblick zu behalten. Dazu dupliziert er die Visualisierung des Proteins mit einer Interaktion. Daraufhin werden zwei gleiche Visualisierungen des Proteins angezeigt.

Wenn die Powerwall nicht groß genug ist oder nur der Tabletop genutzt wird, verbraucht die Ausgangsdarstellung des Proteins zu viel Platz. Daher schiebt der Vortragende diese Darstellung auf die linke obere Ecke und verkleinert sie mit einer Interaktion. Die andere Darstellung, mit der weitergearbeitet wird, schiebt er in die Mitte und vergrößert sie mit einer Interaktion auf die restliche Fläche der Powerwall bzw. des Tabletops. Auf diese Weise

können die Zuschauer alle Details sehen. Die vergrößerte Visualisierung kann der wissenschaftliche Mitarbeiter nun nutzen, um seine Erkenntnisse vorzustellen.

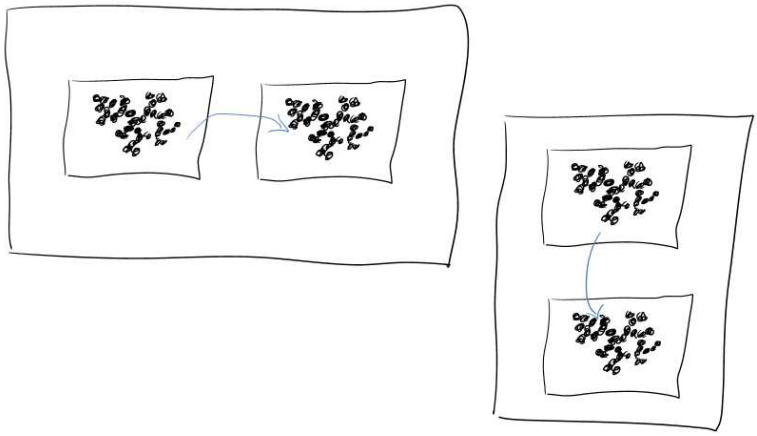


Abbildung 43: Duplizierung der Visualisierung

Um bestimmte Strukturen des Proteins hervorzuheben, markiert der wissenschaftliche Mitarbeiter mit einer Auswahlgeste diese Strukturen und ändert mit einer Interaktion das Mapping dieser Atome auf eine andere Farbe. Auf eine ähnliche Weise ändert er das Mapping der Form für einige Atome.

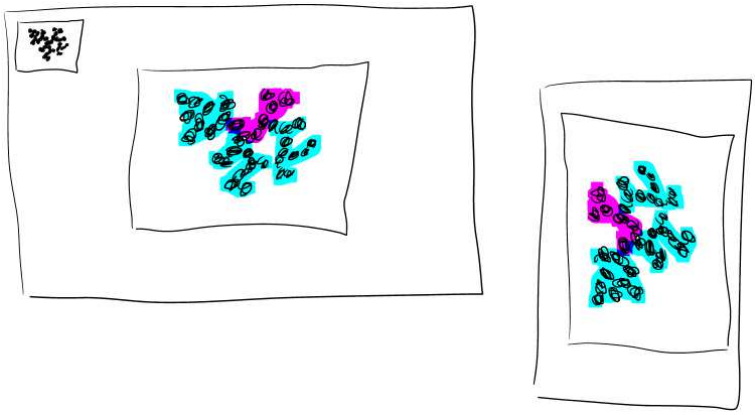


Abbildung 44: Änderung des Mappings der Visualisierung.

Um Ergebnisse zu präsentieren, die in der aktuellen Darstellung nicht sichtbar sind ändert der Vortragende die Art der Darstellung. Dafür ändert er die MegaMol-Visualisierung in der Hauptansicht mit einer Wischgeste. Nun kann er seine Erkenntnisse in der Darstellung der neuen Visualisierung vorstellen.

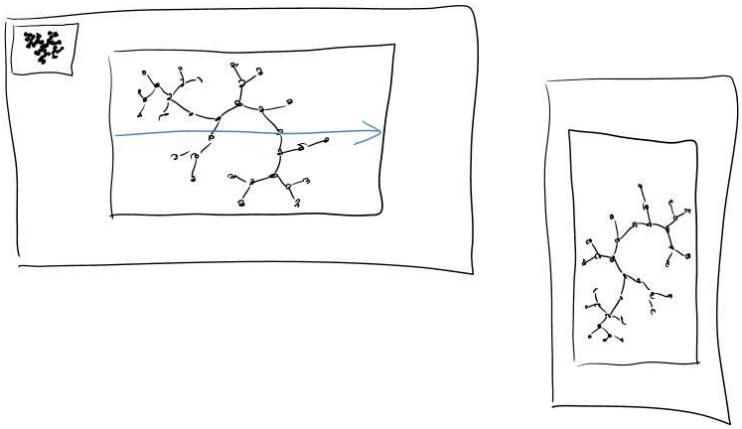


Abbildung 45: Ändern des Visualisierungstyps
Der Vortragende wechselt die Visualisierung mit einer Wischbewegung.

Nun möchte der vortragende Mitarbeiter die Ansicht auf das Molekül innerhalb der MegaMol-Visualisierung ändern. Dazu führt er eine Pan-Geste auf der Visualisierung aus, die die Kamera in der 3D-Ansicht um das Molekül bewegt. Außerdem bewegt er mit einer Geste aus zwei Fingern die Kamera an das Molekül heran. Damit kann er den Zuschauern bestimmte Details besser erläutern.

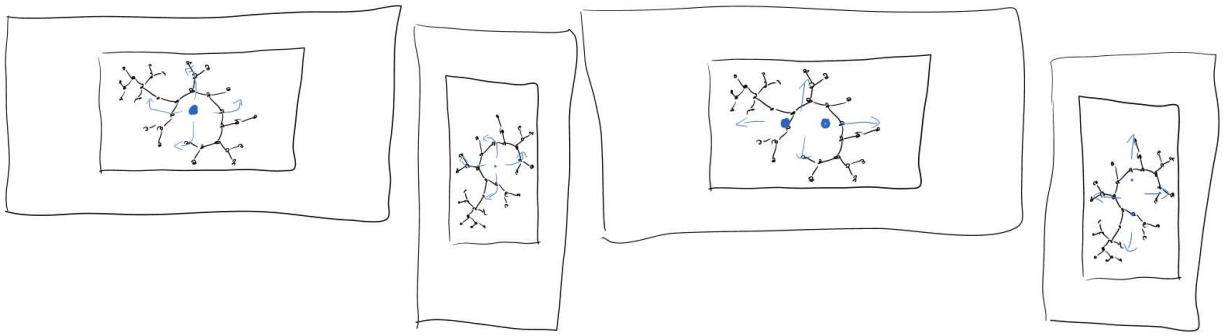


Abbildung 46: Manipulation der 3D-Ansicht mit einer Pan-Geste links und einer Pinch-Geste rechts

Nachdem der wissenschaftliche Mitarbeiter seinen Vortrag beendet hat, entsteht eine Diskussion zwischen ihm und seinen Kollegen. Dabei wird die Ansicht des Proteins noch einige Male geändert. Bei einer Ansicht entdecken die wissenschaftlichen Mitarbeiter etwas, das ihre These unterstützt. Einer der Mitarbeiter markiert darauf einen Bereich der MegaMol-Visualisierung mit einer Interaktion und erstellt dadurch einen Screenshot. Den Screenshot zieht der Mitarbeiter mit einer Drag&Drop-Interaktion wieder auf den Dialog für seinen Laptop. Dieser befindet sich nun auf dem Laptop und kann von dem Mitarbeiter zu einem späteren Zeitpunkt verwendet werden.

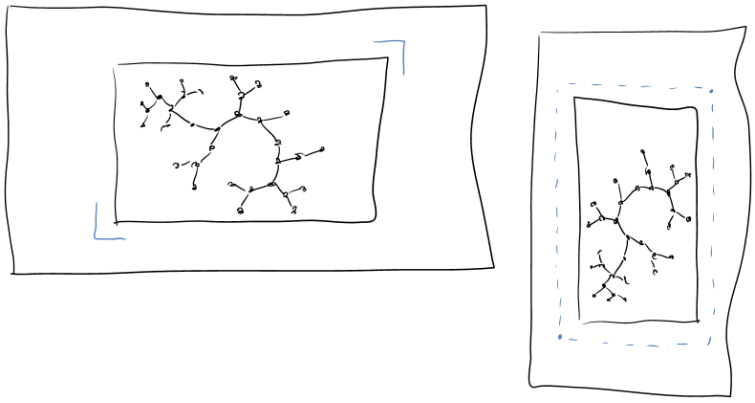


Abbildung 47: Erzeugen eines Screenshots

5.4 Anforderungen

Dieser Abschnitt beschreibt, welche Anforderungen von der Analyseumgebung und einer Tabletop-Steuerung umgesetzt werden müssen. Die Anforderungen ergeben sich zum einen aus den Problemen, die bei der Analyse der Auswertungssitzung (Kapitel 5.1) identifiziert wurden und mit einer solchen Umgebung nicht mehr auftreten sollen. Außerdem wird beschrieben welche Anforderungen sich aus den Szenarien (Kapitel 5.2 und 5.3) für die Analyseumgebung ergeben.

5.4.1 Anforderungen an die Analyseumgebung

Zunächst werden die Anforderungen an die Analyseumgebung beschrieben. Dazu gehören unter anderem welche Anforderungen an das System gelegt werden, das die Kommunikation zwischen der Powerwall und dem Tabletop oder anderen Geräten verwaltet. Außerdem welche Anforderungen an die Benutzbarkeit und die Oberfläche der Geräte innerhalb der Analyseumgebung gestellt werden. Und welche Anforderungen speziell an die Powerwallanzeige gestellt werden.

- *Die öffentlichen Bereiche aller Geräte sollen die gleichen Elemente anzeigen.*
  Der Bereich, der bei allen Geräten auf die Powerwall synchronisiert werden soll, muss immer die gleichen Elemente an entsprechender Position anzeigen. Die Umrechnung der Perspektiven zwischen den einzelnen öffentlichen Bereichen sollte nachvollziehbar sein. Außerdem sollen zugehörige Ansichten auf den verschiedenen Geräten die gleichen Daten nutzen.

- *Die Synchronisation der Geräte soll in angemessener Zeit passieren.*
  Für die Synchronisation der Geräte soll eine Technik verwendet werden, die es erlaubt in einem lokalen Netzwerk die Geräte angemessen schnell zu synchronisieren. Der Benutzer soll daher kaum eine Verzögerung wahrnehmen.

- *Das System soll eine einheitliche Datenhaltung zur Verfügung stellen.*
  Das System der Analyseumgebung sollte eine Datenhaltung haben, die verhindert, dass die einzelnen Geräte asynchrone Daten nutzen.

- *Es sollen verschiedene Geräte eingesetzt werden können.*
  Die Interaktion mit der Analyseumgebung soll mit verschiedenen Geräten möglich sein. Das schließt die Fernsteuerung der Powerwall mit verschiedenen Geräten, wie dem Tabletop, und eine Möglichkeit der direkten Steuerung, wie in der Arbeit [46] beschreiben, ein.

- *Es muss möglich sein, eine Sitzung vorzubereiten.*
  Es soll möglich sein die Daten für eine Analyse in das System zu importieren, so dass sie später innerhalb des Systems genutzt werden können.

- *Verschiedene Visualisierungsarten sollen angezeigt werden können.*
  Das System soll erweiterbar sein, damit verschiedene Arten von Visualisierungen angezeigt werden können.

- *Es sollen viele Visualisierungen gleichzeitig betrachtet werden können.*
  Die Oberfläche und Visualisierungen sollten so gehalten sein, dass möglichst viele Visualisierungen angezeigt werden können.

- *Die Visualisierungen sollen in einer Gruppe analysiert werden können.*
  Die Teilnehmer einer Analyse sollen sich in kleinen Gruppen zusammentun können, um einzelne Aspekte einer größeren Aufgabe zu analysieren. Später sollen die einzelnen Gruppen ihre Ergebnisse präsentieren können, damit in der großen Gruppe ein Konsens entstehen kann.

5.4.2 Anforderungen an Tabletop-Oberfläche und Interaktionen

Dieses Kapitel beschreibt die Anforderungen an die Tabletop-Steuerung der Analyseumgebung. Dazu gehören Anforderungen an die Interaktionen, die Oberfläche der Tabletop-Steuerung und die allgemeine Bedienbarkeit.

- *Die Tabletop-Steuerung soll einen privaten und einen öffentlichen Bereich haben.*
  Damit die Benutzer sowohl auf der Powerwall als auch für sich selbst arbeiten können, soll auf dem Tabletop ein öffentlicher Bereich und ein privater Bereich existieren. Der öffentliche Bereich soll auf allen Geräten und der Powerwall angezeigt werden. Der private Bereich soll nur auf dem Tabletop angezeigt werden, so dass die Benutzer allein oder in kleinen Gruppen arbeiten können.

- *Auf dem Tabletop soll von mehreren Nutzern gleichzeitig gearbeitet werden können.*
  Da der Tabletop für kollaboratives Arbeiten als Mehrbenutzersystem ausgelegt ist, soll die Tabletop-Steuerung der Analyseumgebung eine gleichzeitige Nutzung ihrer Benutzeroberfläche ermöglichen.

- *Der Analysefluss soll nur minimal gestört werden.*
  Wenn eine Interaktion mit der Powerwall nötig wird, soll diese Interaktion auf dem Tabletop so gelöst werden können, dass sie den Analysefluss nicht stört. Sie sollte daher schnell und einfach durchzuführen sein und möglichst auch während der Benutzer etwas erklärt.

- *Die Datenrepräsentation und Filterung soll mit Tangibles möglich sein.*
  Laut der Aufgabenstellung ist ein Filterungssystem erwünscht, das es ermöglicht die Datenmenge mit Tangibles zu beeinflussen. Beispielsweise sollte mit einem Tangible für einen Probanden die Visualisierung nach diesem Probanden gefiltert werden können.

- *Die Interaktionen sollen möglichst intuitiv sein.*
  Natural User Interfaces sollen es dem Benutzer ermöglichen, als Anfänger die Bedienung eines Systems schnell zu erlernen. Daher sollten die einzelnen Interaktionen intuitiv sein und den Vorstellungen der Benutzer entsprechen.

6 Konzept

Dieses Kapitel beschreibt grundsätzliche Konzepte, um die Anforderungen aus Kapitel 5 umzusetzen. Dabei wird zuerst das Konzept für das benötigte System vorgestellt. Weiter wird die grundsätzliche Benutzeroberfläche auf Tabletops für das geforderte System erläutert. Danach werden einzelne Interaktionsmodalitäten aufgezeigt, die entsprechend der Anforderungen bei den Funktionen eingesetzt werden können. Anschließend wird das Konzept für das lokale Datenmanagement des Tabletops erklärt, das benötigt wird um das Konzept der Benutzeroberfläche mit dem des Systems in Einklang zu bringen. Das Zusammenspiel der einzelnen grundsätzlichen Konzepte, die in diesem Kapitel vorgestellt werden, beschreiben anschließend Kapitel 7, 8 und 9.

6.1 System

Wie in Kapitel 5 erläutert soll ein System entwickelt werden, das eine Analyse oder Präsentation von Daten mit Visualisierungen an der Powerwall ermöglicht. Die Umsetzung des hier vorgestellten Konzepts für das System der Analyseumgebung wird in Kapitel 8 vorgestellt.

Der Benutzer soll die Möglichkeit haben mit dem System schnell und unkompliziert zu interagieren. Weiterhin soll das System die Möglichkeit bieten eine Analyse in einer Gruppe durchzuführen. Daher soll das System auch von mehreren Personen bedient werden können. Darauf aufbauend soll das System unterstützen, Analysen im Privaten durchzuführen und die Ergebnisse der Gruppe vorzustellen. Diesen Erkenntnissen entsprechend, ergeben sich für die Hardware zur Bedienung des Systems verschiedene Optionen.

- **System zur Freiraum-Gestenerkennung**
  Freiraum-Gesten ermöglichen es direkt mit der Powerwall zu interagieren. Dies ermöglicht direkte Manipulation der Elemente auf der Powerwall. Mögliche Hardware sind Controller wie die Wiimote, Laserpointer oder für Freihand-Gesten die Kinect von Microsoft. Damit sind keine oder nur eingeschränkte private Umgebungen umzusetzen.

- **Tabletop-PC**
  Ein Tabletop ist wie in Kapitel 2.3 beschrieben ein PC in Form eines Tisches mit Multitouch-Eingabe. Tische sind bei Besprechungen oder Konferenzen üblich und daher fügen sich Tabletops mit ihrer Berührungseingabe natürlicher in die Besprechung mit ein als das bei einem normalen PC der Fall ist. Leider ist mit den Tabletops nur indirekte Interaktion mit der Powerwall möglich. Dafür bietet der Tabletop eigene Interaktionsmodalitäten wie Tangibles an, die neue Interaktion ermöglichen und gleichzeitig die Distanz zwischen Nutzer und Benutzerschnittstelle verringern. Tabletops sind außerdem für Mehrbenutzersysteme optimiert. So können auch mehrere Personen privat an einem Tabletop arbeiten

- **Tablets**
  Tablets sind kleine, berührungssensitive PCs mit begrenzter Multitouch-Unterstützung. Dabei sind insbesondere „Slates“ von Interesse. Diese Art von Tablets ist flach und hat zur Bedienung nur einen berührungsempfindlichen Bildschirm. Ein Beispiel ist das Apple iPad. Sie sind sehr mobil und lassen sich während einer Besprechung komfortabel nutzen. Sie können ähnlich einem Notizblock im Stehen oder im Sitzen genutzt werden. Mit ihnen lassen sich einfach private Bereiche

umsetzen. Um allerdings Mehrbenutzerbetrieb zu gewährleisten sind mehrere Tablets nötig.

- **Smartphones**
  Smartphones können analog zu Tablets eingesetzt werden. Der hauptsächliche Unterschied ist mehr Mobilität zu Kosten von Bildschirmgröße und Bedienkomfort.

Da alle Bedienungskonzepte und zugehörige Hardware ihre eigenen Stärken aufweisen, ist es für das System am sinnvollsten eine offene Umgebung zu schaffen, die eine Integration aller dieser Konzepte zulässt. Diese Umgebung soll durch ein Kommunikationsframework zwischen den einzelnen Geräten bereitgestellt werden.

Das Kommunikationsframework stellt eine Abstraktion jeglicher Kommunikation, sowohl im Bereich von Interaktionen als auch für den Datenaustausch, zur Verfügung. Das Kommunikationsframework arbeitet mit Nachrichten. Das Framework bietet eine Grundmenge an Nachrichtentypen an, welche aber erweitert werden kann. Weiterhin bietet es eine grundsätzliche Abstraktion der Daten und Visualisierungen an, um eine Erweiterung der Datenhaltung und des Nachrichtensystems offen zu halten. So können jegliche Arten von Daten mit einer freien Auswahl an Visualisierungen untersucht werden. Durch die Abstraktion können die Visualisierungen ebenfalls auf unterschiedlichen Systemen implementiert werden. Jedoch sollte die Darstellungsweise ähnlich sein. Das Framework ist ein Client-Server-System. Jedes Gerät erhält dabei einen Client. Der Server registriert die Clients und verwaltet die Kommunikation. Weiterhin sollte der Server auch die Datenhaltung übernehmen, so dass alle Clients auf die gleiche Datenquelle zugreifen können. Um dabei eine gute Flexibilität zu erreichen, wird hier eine Schnittstelle für Datenquellen eingeführt.

Auf Seiten der Benutzeroberfläche werden die Daten innerhalb von Visualisierungen angezeigt. Um eine Fernsteuerung der Powerwall mit Geräten, die keine direkte Manipulation der Powerwall erlauben zu ermöglichen, muss die Darstellung und die grundlegenden Interaktionen der Visualisierungen kompatibel sein. Dazu wird ein Visualisierungscontainer genutzt, der jede Art von Visualisierung auf Basis der Abstraktion des Frameworks aufnehmen kann. Dabei wird auf das bewährte Prinzip der Fenster zurückgegriffen. Die Oberfläche wird abgesehen von der Powerwall in einen privaten und einen öffentlichen Bereich eingeteilt. Der öffentliche Bereich stellt die Anzeige der Powerwall auf dem Gerät dar und ermöglicht so die Fernsteuerung. Die Visualisierungsfenster können zwischen privaten und öffentlichen Raum verschoben werden, um auf der Powerwall angezeigt zu werden. Die Datengrundlage der Visualisierungen wird in einem Datenexplorer angezeigt und dient gleichzeitig zum Anlegen neuer Visualisierungen.

Diese Arbeit konzentriert sich auf den Teil des Tabletop-Clients. Die Integration eines Tabletops in das System ist folgendermaßen: In dem Besprechungsraum mit der Powerwall wird ein Tabletop beliebig vor der Powerwall positioniert. Sowohl auf dem Tabletop, als auch auf dem Steuerrechner der Powerwall läuft ein Client des Systems. Der Client der Powerwall kann zusätzlich auch direkte Interaktion mit den vorherig genannten Methoden unterstützen. Das wird in der Arbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ [46] entwickelt. Auf einem beliebigen Rechner läuft der Server des Systems.

Im Weiteren werden die grundlegenden Konzepte in Bezug auf Benutzeroberfläche und Interaktion des Tabletops dargestellt.

6.2 Tabletop-Oberfläche

Das Kapitel beschreibt die Grundlagen für den Lösungsansatz der Bedienoberfläche auf dem Tabletop.

Der Entwurf der Bedienungsoberfläche des Tabletops hält sich an die Überlegungen in Kapitel 6.1. Somit wird die Oberfläche in einen privaten und einen öffentlichen Raum eingeteilt. Jeder dieser Räume zeigt die Visualisierungen in Visualisierungscontainern an. Jeder Container hat eine eigene Datengrundlage, die separat gefiltert werden kann.

Zuerst wird die grundlegende Funktion der Visualisierungscontainer beschrieben. Danach wird auf den öffentlichen Raum und dessen Ausprägung auf einem Tabletop eingegangen. Ebenso wird der private Raum beschrieben. Zum Schluss wird auf zwei verschiedene Ablagen für Visualisierungscontainer eingegangen.

Die konkreten Formen der Benutzeroberfläche und der hier beschriebenen Konzepte wird in Kapitel 7 beschrieben.

Interaktion mit den Visualisierungscontainern
Wie in den Anforderungen erarbeitet gibt es einige Interaktionen, die mit den Containern und ihren zugehörigen Visualisierungen durchgeführt werden können. Die Container können innerhalb der Räume skaliert und verschoben werden.

Wenn ein Benutzer eine identische Ansicht für eine weitere Verarbeitung braucht, um z.B. in eine andere Richtung zu explorieren, kann der Container dupliziert werden. Die Visualisierung des duplizierten Containers hat danach die gleiche Ansicht, wie die des Originals. Der neue Container kann danach unabhängig von seinem Original genutzt werden.

Eine Erweiterung dieser Idee ist die Kopplung. Falls ein Benutzer zwei verschiedene Visualisierungen mit der gleichen Datengrundlage und Filterung anzeigen möchte, dann kann er die zugehörigen Container koppeln. Wenn die Filterung der Datengrundlage eines Containers geändert wird, dann ändert sich diese auch in dem gekoppelten Container. Ein Beispiel dafür wird in dem Eyetracking-Szenario in Kapitel 5.2 vorgestellt. Eine weitere Art von Kopplung, die hauptsächlich für den privaten Raum sinnvoll ist, wirkt sich nicht nur auf die Daten sondern auch auf die Anzeige aus. Dabei wird in Hinblick auf die Visualisierungspipeline, im Gegensatz zur vorherigen Kopplung, das Rendering statt der Filterung gekoppelt. Die Anzeige der gekoppelten Visualisierungen ist also immer identisch. Diese Kopplung kann genutzt werden um in kleineren Gruppen Erkenntnisse zu besprechen und Daten gemeinsam zu explorieren.

Innerhalb des Containers kann sowohl die Visualisierung als auch die Datengrundlage gewechselt werden. Es kann nur auf eine Visualisierung gewechselt werden, die den aktuellen Typ von Daten unterstützt. Analog dazu kann beim Wechsel der Datengrundlage nur innerhalb des aktuellen Datentyps gewechselt werden. Das bewahrt die Übersicht und führt zu keinem verwirrenden Wechseln bei der Visualisierung. Beide Wechsel sollten eine Vorschau besitzen wenn möglich.

Privater Raum
Im privaten Raum des Tabletops können einzelne Benutzer oder kleinere Benutzergruppen die Visualisierungen explorieren, ohne dass die Visualisierungen in die Analyseumgebung übertragen werden. Dies dient zur Vorbereitung von Ansichten, die später der größeren Gruppe vorgestellt werden. Auf ihm soll wie auf einem normalen Tisch gearbeitet werden können.

Im privaten Raum können, wie schon genannt, mehrere Personen gleichzeitig arbeiten. Er nutzt die Mehrbenutzerfähigkeit des Tabletops. Jeder Container kann von einem anderen Benutzer bedient werden. Somit können mehrere Benutzer arbeiten ohne ein eigenes Gerät zu haben. Je nach Größe des Tabletops ist die Bedienung mit vielen Personen und längere Zeit nicht sinnvoll. Allerdings ist das in einer Besprechung auch nicht angebracht. Alle Interaktionen mit und zwischen Containern sind für eine Mehrbenutzerumgebung entworfen. Innerhalb eines Containers ist keine Mehrbenutzerbedienung möglich. Mögliche Anwendungsfälle die dies benötigen sollten allerdings größtenteils durch die Kopplung abgedeckt werden.

Da der Tabletop frei vor der Powerwall positioniert werden kann und die Benutzer überall um den Tisch stehen können, muss die Ausrichtung der Container auf dem privaten Raum geändert werden können. Es gibt grundsätzlich zwei Arten von Methoden wie die Orientierung der Container festgelegt werden kann. Die Methoden sind manuell und automatisch.

- **Manuelle Orientierung**
  Die manuelle Orientierung überlässt es komplett dem Benutzer wie der Container ausgerichtet wird. Sie benötigt Interaktion, um die Container auszurichten. Bei der manuellen Ausrichtung erhält man völlige Flexibilität bei etwas mehr Interaktionsbedarf.

- **Automatische Orientierung**
  Bei der automatischen Orientierung werden die Container entsprechend von Regeln ausgerichtet. Normalerweise wird die Position als Ausgangspunkt genutzt. Dragicevic et al. [49] stellen eine Methode mit Vektorfeldern vor. Bei den Vektorfeldern ist die Ausrichtung des Containers abhängig von der Position auf dem Tisch angeben. Wenn der Container bei automatischer Ausrichtung auf dem Tisch verschoben wird, ändert er entsprechend des Vektorfelds seine Orientierung. Die nötigen Interaktionen für Orientierung fallen bei dieser Methode weg, dies geht auf Kosten der Flexibilität.

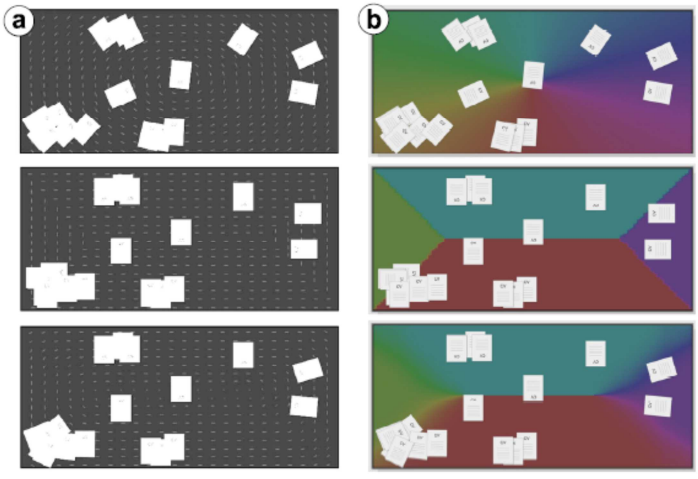


Abbildung 48: Zeigt einige Ausrichtung anhand von Vektorfeldern a) Vektoren als Pfeildarstellung, b) Vektorrichtung als Farbe [49]

In Anbetracht der vielen verschieden Situationen bei einer Besprechung, vor allem wenn Benutzer sich öfters zur Powerwall begeben, stellt die manuelle Ausrichtung in diesem Fall die bessere Wahl dar. Es können sich verschiedene Situationen ergeben z.B. können die Benutzer sich anfangs um den Tabletop stellen und später hinter diesem stehen um Ergebnisse

an der Powerwall zu betrachten. Die manuelle Ausrichtung verträgt sich ebenfalls besser mit der Analogie zu einem normalen Tisch.

Weil sich das „privat“ im privaten Raum auf das Gerät bezieht, muss in Betracht gezogen werden, dass es weitere Räume für die Benutzer innerhalb des privaten Raums gibt. Diese persönlichen Räume grenzen sich von Bereichen, die von der Gruppe genutzt werden, ab. Der persönliche Bereich ist meist direkt vor oder neben dem Benutzer. Scott et al. [50] haben dazu das Verhalten von Personen beim Arbeiten an Tischen untersucht. Da das Szenario wie schon genannt sehr dynamisch ist, werden keine persönlichen Räume im Konzept vorgesehen. Das Prinzip findet sich jedoch zum Teil bei der persönlichen Ablage (siehe weiter unten) wieder. Die persönlichen Räume ergeben sich daher wie bei einem normalen Tisch in Eigendynamik.

Öffentlicher Raum
Der öffentliche Raum dient der Fernsteuerung und damit der Interaktion mit der Powerwall. Seine Anzeige ist direkt mit der der Powerwall verbunden. So können Visualisierungen auf der Powerwall angezeigt werden. Dadurch lassen sich Ergebnisse vorstellen oder eine Analyse in der größeren Gruppe durchzuführen.

Die Fernsteuerung ermöglicht es, die Visualisierungen der Powerwall anzupassen oder deren Container zu verschieben bzw. zu skalieren. Im Gegensatz zum privaten Raum auf dem Tabletop ist eine Rotation hier nicht sinnvoll. Die Visualisierungscontainer können wie im System beschrieben zwischen dem öffentlichen Raum und dem privaten Raum verschoben werden. Hier ist eine möglichst nahtlose Interaktion zu wählen, um eine natürliche Bedienung zu erreichen.

Die Anzeige des öffentlichen Raums auf dem Tabletop hat in den meisten Fällen nicht dieselbe Größe wie die Powerwall und ist daher skaliert. Um eine bessere Bedienung zu ermöglichen, kann die Anzeige des öffentlichen Raums als Viewport frei auf der Powerwall verschoben oder skaliert werden. Daher kann der Benutzer wählen ob er eine verkleinerte Ansicht der Powerwall oder einen nicht skalierten Teil der Powerwall sehen möchte.

Für die Anzeige des öffentlichen Raums sind verschiedene Optionen denkbar. Eine Anzeige, die separat von der Anzeige des privaten Raums positioniert ist. Dabei käme z.B. ein gesplitteter Bildschirm oder Tabs in Frage. Eine andere Option wäre es den öffentlichen Raum als eigener Container im privaten Bereich anzuzeigen. Diese Option vereinfacht die Interaktionen und erlaubt ein freieres Anpassen der öffentlichen Anzeige. Dafür ist die Trennung der beiden schwieriger zu erkennen. Umgekehrt gäbe es die Möglichkeit den privaten Raum innerhalb des öffentlichen anzuzeigen. Diese Option ist allerdings verwirrend und lässt weniger Freiheit bei der Bedienung. So ist es z.B. nicht mehr von überall mehr möglich im privaten Raum zu arbeiten.

Persönliche Ablage
Der Tabletop ist bei vielen Benutzern schnell überfüllt, daher möchten Benutzer unter Umständen Visualisierungen vom Tisch entfernen. Da Elemente vom Tabletop nicht wie von einem normalen Tisch entfernt werden können, muss eine Ablage genutzt werden. Da ein Benutzer bei einem normalen Tisch die Möglichkeit hat seine Dokumente bei sich zu verstauen, liegt es nicht fern dieses Konzept bei Tabletops auch zu nutzen. Dafür benötigt der Benutzer eine Art der Identifizierung um auf seine Ablage zuzugreifen. Eine mögliche Lösung dafür wäre bei Tabletops Tangibles. Diese lassen sich im Gegensatz zu einer Eingabe wie eines Benutzernamens oder einer anderen ID schneller nutzen.

Dieses Konzept lässt sich ebenfalls mit persönlichen Geräten verbinden. So ist die persönliche Ablage keine reine Ablage mehr sondern eine Verbindung zu einem privaten Gerät innerhalb des Systems. So können z.B. Visualisierungen auf den privaten Tablet übertragen werden und alleine weiter untersucht werden.

Globale Ablage
Analog zur persönlichen Ablage ist auch eine globale Ablage denkbar. Sie ermöglicht es mehreren Nutzern Elemente gemeinsam abzulegen. Bei richtigem Entwurf kann sie auch von den Benutzern eingesetzt werden um zwischen den Benutzern Elemente auszutauschen.

Globales Menü
Das globale Menü ermöglicht den Zugriff auf globale Funktionen des Systems. So können z.B. globale Elemente wie der Dialog für Einstellungen oder der öffentliche Raum ein- und ausgeblendet werden.

Das globale Menü braucht eine eindeutige Interaktion, mit der es von überall geöffnet werden kann. Dafür würde eine einzigartige Geste oder ein Tangible in Frage kommen.

6.3 Tabletop-Interaktionsmodalitäten

Tabletops haben verschiedenste Interaktionsmodalitäten. Es gibt die klassischen Steuerelemente. Diese sind für Berührungsinteraktion optimiert worden ähnlich wie die auf Smartphones. Weiter gibt es die in Natural User Interfaces und Smartphones beliebten Touchgesten. Am modernsten sind jedoch die Tangibles mit denen Tangible User Interfaces entwickelt werden. Leider sind Tabletops ähnlich wie Tablets für normale Desktopanwendungen nicht geeignet. Daher müssen Anwendungen für Tabletops in Hinblick auf diese Interaktionsmodalitäten entworfen werden. Dieses Kapitel stellt die einzelnen Interaktionsmodalitäten vor und gibt mögliche Anwendungsgebiete innerhalb des Tabletop-Clients an.

Die konkrete Nutzung der einzelnen Interaktionsmodalitäten für die Tabletop-Oberfläche wird in Kapitel 7 beschrieben.

Interaktion mit klassischen Steuerelementen
Klassische Steuerelemente wie Menüs, Werkzeugleisten oder einzelne Buttons haben sich bei Desktops seit langem durchgesetzt und werden bis heute eingesetzt(siehe 2.2.2). Da jedoch für Maus entwickelte Steuerelemente mit Berührung nur sehr schwer bedient werden können, weil z.B. die Finger zu dick für die meist filigranen Elemente sind [23], wurden für Berührungseingabe neue oder optimierte Steuerelemente entwickelt. Diese Entwicklung begann in größerem Maßstab mit den Smartphones, die bei dieser Bedienung eine Vorreiterrolle einnehmen. Dieselben Erkenntnisse wurden dann auch für Tablets angewendet. Bei Tabletops wurde ebenfalls in dieser Richtung geforscht und Frameworks mit optimierten Steuerelementen entwickelt. Die Optimierungen sind ähnlich wie bei Smartphones, Größenanpassung der Elemente für Finger, Einbindung von Pan-, Pinch- und Slide-Gesten.

Die klassischen Steuerelemente haben folgende Vorteile:

- *Sie haben einen starken Aufforderungscharakter.*
  Steuerelemente wie Knöpfe oder Menüs haben meist einen starken Aufforderungscharakter nach den Designkonzepten von D.A. Norman [51]. Das heißt der Nutzer versteht beim Anschauen des Elements dessen Zweck. Dies ergibt sich natürlich auch aus der oft langen Erfahrung von Benutzern mit diesen

Steuerelementen. Dadurch unterstützen die Steuerelemente den ungelernten Benutzer bei dem Verständnis der Funktionen.

- *Sie sind teilweise schnell zu bedienen.*
  Elemente wie Knöpfe oder Schieberegler lassen sich schnell bedienen, denn sie benötigen nur eine Interaktionsform wie Drücken oder Schieben.

Die Nachteile der klassischen Steuerelemente sind dagegen

- *Manche klassischen Steuerelemente sind nur langsam zu bedienen.*
  Steuerelemente wie Menüs oder größere Listen sind nicht so schnell zu bedienen wie andere Steuerelemente. Sie werden daher von erfahrenen Benutzern oft nicht verwendet.
- *Sie verbrauchen viel Platz.*
  Beschriftungen oder Icons, die den Steuerelementen ihren Aufforderungscharakter verleihen wirken sich negativ auf die Größe aus. Dadurch können sie bei Benutzeroberflächen, die wenig Platz zur Verfügung haben nur begrenzt eingesetzt werden. Dieser Effekt verstärkt sich noch bei Elementen für die Berührungsbedienung, da diese meist noch Größer sind.

Für die Anwendung ergibt sich aus diesen Erkenntnissen, dass klassische Steuerelemente wie Knöpfe für häufige und wichtige Aufgaben eingesetzt werden. Steuerelemente wie Menüs dagegen für Aufgaben die seltener genutzt werden müssen oder einen stärken Erklärungsbedarf haben.

Drag&Drop-Interaktion
Drag&Drop ist eine Erweiterung von Icons kombiniert mit Mausinteraktion in Richtung von Gesten. Wie der Name sagt basiert es auf dem Prinzip, dass ein Element aufgenommen, verschoben (drag) wird und an einer anderen Stelle abgelegt (drop). Damit hat es eine Analogie zur realen Welt, was sich meist vorteilhaft auf die Benutzererfahrung auswirkt.

Der Aufforderungscharakter hängt von dem Ausgangspunkt der Interaktion ab. Das Ausgangselement muss anzeigen, dass es gezogen werden kann. Das ist nicht einfach und wird von Benutzern oft erst durch Ausprobieren erkannt. Sobald ein Element aufgenommen ist, kann es durch die Anzeige am Zeiger oder Berührpunkt mögliche Interaktionen mit anderen Elementen darunter anzeigen. Daher hat die Interaktion auch immer ein kontinuierliches Feedback, was eine natürliche Bedienung zulässt. Durch die Interaktion mit der Oberfläche sind kontextabhängige Funktionen möglich.

Dadurch, dass die Herkunft der Drag&Drop-Interaktion bekannt ist und es sich um eine Interaktion mit nur einem einzelnen Berührungspunkt handelt, kann sie sehr gut in Mehrbenutzersystemen angewendet werden. Aus diesem Grund bieten Frameworks für Tabletops einige Grundlagen für Drag&Drop-Interaktionen. Ein Beispiel ist das Surface SDK von Microsoft, dort sind Drag&Drop-Interaktionen fest integriert und gut unterstützt. Außerdem gibt es Steuerelemente, die speziell für Drag&Drop Operationen optimiert sind.

Gesten-Interaktion
Gesteninteraktion ist jede Art von dynamischer Interaktion auf dem Tabletop. Bei Tablets und Smartphones haben sich bereits Gesten für Berührungsinteraktion durchgesetzt. Die meist genutzten sind für Verschieben, Skalieren und Rotieren.

Das größte Problem von Gesten ist ihre Existenz anzuzeigen. Denn Gesten verändern meist direkt die Benutzeroberfläche, wie z.B. beim Skalieren. Daher ist der Aufforderungscharakter von Gesten meistens schwierig, denn woher weiß der Benutzer was er manipulieren kann [52]. Ihre Nutzbarkeit kommt durch ihre natürliche Bedienung mit der Benutzeroberfläche. Die Beispiele dafür sind Verschiebe- und Pan-Gesten, diese verändern die Benutzeroberfläche mit der Bewegung der Geste. Solche Gesten liefern dabei auch wie die Drag&Drop-Interaktion ein kontinuierliches Feedback. Je komplizierter eine Geste ist, desto schwieriger wird es sie in die Benutzeroberfläche zu integrieren. Je komplizierter eine Interaktion ist, desto schwieriger wird es dafür eine natürliche Geste zu finden. Aus diesem Grund wird in von Wobbrock et al. [48] das Verfahren vorgestellt, bei der Benutzer selbst Gesten definieren. Dabei wird eine Studie mit Benutzer durchgeführt, bei der die Benutzer eine Geste für eine bestimmte Aufgabe durchführen sollen. So können intuitive Gesten für bestimmte Aufgaben gefunden werden.

Gesten können überall eingesetzt werden wo Elemente manipuliert werden können. Sie bieten den Vorteil, dass sie keinen Platz durch Steuerelement verbrauchen. Wie schon genannt müssen Gesten aber mit Sorgfalt entworfen werden, sonst sind sie für den Benutzer nicht verständlich. Etwas anders verhält es sich mit bereits etablierten Gesten. Diese sind meist so gebräuchlich, dass die meisten Nutzer deren Existenz als gegeben ansehen. Dazu gehören die schon zu Anfang genannten Gesten: Verschieben, Skalieren und Rotieren.

Tangible-Interaktion
Die Tangible erweitern die Interaktionsmöglichkeiten um eine neue Dimension, indem sie durch fassbare Objekte Interaktionen mit der Benutzerschnittstelle erlauben. Weiterhin geben sie dem Benutzer eine stärkere Nähe zu der Benutzeroberfläche, was vor allem für Nutzer mit weniger Erfahrung mit Computern angenehm sein könnte.

Zwei Nutzungen für Tangible liegen nahe: Einmal als Interaktionstangible, das eine festgelegte Interaktion durchführt. Die zweite Möglichkeit ist etwas komplexer. Dabei ist das Tangible eine Datenrepräsentation, mit der interagiert werden kann, bzw. die mit der Benutzeroberfläche interagiert.

Interaktionstangible
Ein Interaktionstangible hat eine ihm zugeordnete Interaktion. Für diese Interaktion gibt es eine bestimmte Sequenz, die mit dem Tangible durchgeführt wird. In diesem Sinne sind die Interaktionstangibles vergleichbar mit Gesten. Allerdings ist durch die Existenz des Tangibles ein höherer Aufforderungscharakter vorhanden. Im Weiteren müssen die Interaktionen nicht komplett verbunden sein wie bei Gesten, da die Tangibles eindeutig sind. Die Interaktionen sind zwar fest zugeordnet, können aber kontextabhängig unterschiedliche Aufgaben erfüllen. So kann ein Tangible z.B. je nachdem wo es auf der Benutzeroberfläche abgesetzt wird verschiedene Funktionen auslösen.

Diese Art von Tangibles kann ähnlich wie Gesten alle möglichen Aufgaben übernehmen. Da Tangibles aber nur sehr wenig in größerem Maß genutzt worden sind, gibt es noch keine Erfahrungswerte. Daher muss auch hier sorgfältig entworfen werden.

Datenrepräsentationstangible
Diese Tangibles repräsentieren visuelle Elemente oder Datenelemente des Systems. Wenn ein Tangible ein Datenelement zugeordnet wird, kann es als Repräsentant des Datums mit der Benutzeroberfläche interagieren. Im Gegensatz zu einer visuellen Repräsentation des Datums innerhalb der Benutzeroberfläche, kann so viel einfacher mit diesen interagiert werden. Weiterhin können die Tangibles und damit die Datenelemente von dem Tabletop entfernt

werden und abseits der Benutzeroberfläche abgelegt werden. Dadurch erhält die Benutzeroberfläche eine neue Dimension der Interaktion. Außerdem bieten die Tangibles den Benutzern die Möglichkeit mit anfassbaren Objekten zu interagieren, das bringt den Benutzer näher an die Benutzeroberfläche.

Analoges gilt für Repräsentationen von visuellen Elementen der Benutzeroberfläche, wie z.B. Containern. Auch diese können von Tangibles repräsentiert werden. Das ermöglicht es visuelle Elemente auf der Benutzeroberfläche mit Tangibles zu manipulieren oder auszublenden.

Mit dieser Art von Tangible können, je nach Art des repräsentierten Elements, Ausblendeaktionen, Positionierungsaktionen, Zuordnungsaktionen und Filteraktionen durchgeführt werden.

6.4 Tabletop-Datenmanagement

Das Framework übernimmt wie in Kapitel 6.1 beschrieben einige Teile des Datenmanagement durch Anbieten der Datenquelle und Abstraktion der Daten. Allerdings benötigt der Tabletop-Client mit seinem privaten Raum ein eigenes Datenmanagement. Dieses Datenmanagement verwaltet die Daten und Visualisierungen auf dem Tabletop. Die konkrete Umsetzung der Datenhaltung wird in Kapitel 9 vorgestellt.

Die Verwaltung beinhaltet die Datenhaltung der Visualisierungen im privaten Raum. Weiter verwaltet das Datenmanagement die Wechsel von Visualisierungscontainern zwischen privaten und öffentlichen Raum. Wenn eine Visualisierung aus dem öffentlichen Raum genommen wird, muss sie zwar in der Datenhaltung bleiben aber aus der Analyseumgebung entfernt werden. Umgekehrt muss eine Visualisierung, die vom privaten Raum in den öffentlichen Raum überführt wird, in die Analyseumgebung aufgenommen werden.

Ebenso muss das Datenmanagement die Verwaltung der Kopplungen übernehmen. Es speichert die Kopplungen zwischen den Visualisierungen und hält Daten von Visualisierungen vor, die mit welchen aus dem öffentlichen Raum gekoppelt sind. Denn diese Daten müssen gesondert behandelt werden, da sie sich in der Analyseumgebung befinden und gleichzeitig von Visualisierung im privaten Raum genutzt werden.

7 Lösungsansatz

Der Lösungsansatz beschreibt Interaktionen und Benutzeroberflächenelemente, für die im Lösungskonzept (Kapitel 6) beschriebenen Konzepte unter Einbeziehung der Anforderungen, die in der Anforderungsanalyse (Kapitel 5) identifiziert wurden.

In 7.1 werden die Vorüberlegungen zu der Benutzeroberfläche und den Interaktionen beschrieben. Diese wurden auf Basis der Anforderungen, Beobachtungen aus anderen Projekten (siehe Kapitel 3), Konzepten aus dem Surface SDK und eigener Ideen entworfen. Die Funktionen, die die Interaktionen ermöglichen sollen, wurden dafür in Kapitel 5 identifiziert. Nach den Vorüberlegungen wird die Vorstudie in 7.2 beschrieben, mit der die Interaktionen und die Oberfläche aus 7.1 evaluiert und mit Ideen der Probanden ausgebaut wird. Am Ende wird in 7.3 der Lösungsansatz aus den Vorüberlegungen mit den Ergebnissen der Vorstudie noch einmal überarbeitet.

7.1 Vorüberlegungen

Die Vorüberlegungen zeichnen einen vorläufigen Lösungsansatz für die Benutzeroberfläche und die Interaktionen.

Zuerst wird ein Überblick über die Benutzeroberfläche gegeben. Danach werden die Teile der Benutzeroberfläche, wie die Powerwallanzeige und die Visualisierungscontainer, beschrieben. Im Anschluss daran werden einige Interaktionen bzw. Funktionen dieser Teile näher beschrieben. Zum Schluss wird das Filterkonzept für die Daten der Visualisierungen beschrieben und die persönliche Ablage für Visualisierungen.

7.1.1 Überblick über die Benutzeroberfläche

Die Benutzeroberfläche ist eine freie Fläche auf dem Tabletop auf der die einzelnen Elemente ungeordnet liegen (siehe Abbildung 49). Dieses Prinzip ist durch das Surface SDK [32] inspiriert. Ist aber auch ähnlich im ZOIL Projekt [39] vorhanden.

Die gesamte Benutzeroberfläche kann für den privaten Raum genutzt werden. Daher liegen die Visualisierungen des privaten Raums direkt auf der Benutzeroberfläche. Da die Visualisierungen aber als Fenster funktionieren, liegen sie in Visualisierungscontainern. Die Visualisierungscontainer können frei auf der Oberfläche verschoben und rotiert werden. Ebenfalls auf der Benutzeroberfläche befinden sich die Powerwallansicht, der Datenexplorer und die Oberflächenelemente, die auf Grund der Tangibleinteraktion auf der Benutzeroberfläche erzeugt werden. Der Datenexplorer enthält Drag&Drop-Elemente für Daten, die in der Analyseumgebung genutzt werden können. Diese können entnommen werden und erzeugen neue Visualisierungen. Elemente wie die Powerwallansicht, der Datenexplorer oder mögliche weitere Systemsteuerelemente können ebenfalls frei auf der Oberfläche verschoben und rotiert werden. Die Elemente der Tangibles sind abhängig von diesen. Tangibles, die Elemente auf der Oberfläche erzeugen sind die Tangibles für die Filterung und die private Ablage. Die Tangibles für das Filtern erzeugen einen Kreis unter diesem, in dem dann die Filterinteraktion stattfinden kann. Das Tangible für die private Ablage erstellt einen Dialog als Ablagefläche für Visualisierungscontainer.

Die Kopplung von Visualisierungen innerhalb des privaten Raums wird als Linie zwischen den Visualisierungscontainern angezeigt. Ebenso wird die Beziehung zwischen einer Visualisierung und ihres Filtertangibles mit einer Linie dargestellt. Die gleiche Darstellung kommt daher, dass die Kopplung und die Filterung die Elemente auf Datenebene auf ähnliche Weise verbinden. Um eine Kopplung wieder zu entfernen, befinden sich an den Linien

Knöpfe zum Entfernen mit einem entsprechenden Icon. Damit beide Benutzer einer Visualisierung eine Kopplung aufheben können befindet sich an beiden Enden ein Knopf.

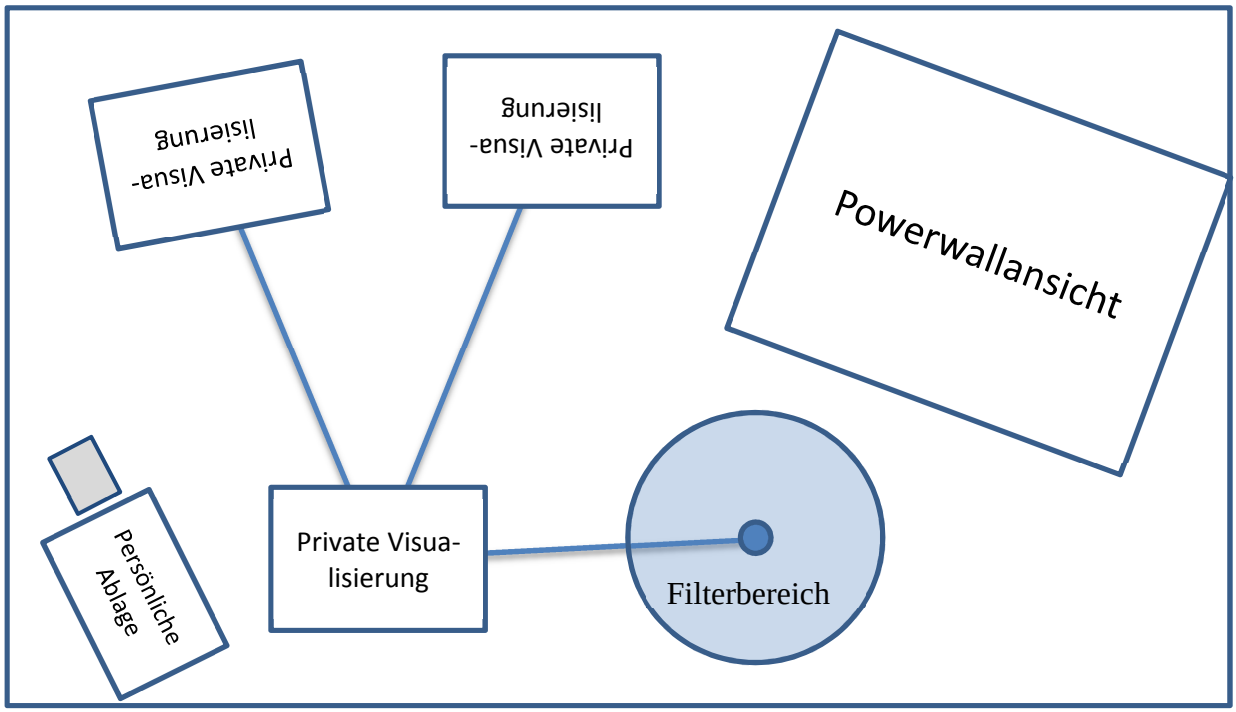


Abbildung 49: Skizze der Benutzeroberfläche

7.1.2 Powerwallansicht

Die Powerwallansicht (siehe Abbildung 50) repräsentiert den öffentlichen Raum auf der Benutzeroberfläche des Tabletops. Sie ermöglicht es dem Benutzer Visualisierungen auf die Powerwall zu übertragen und diese zu steuern. Die Powerwallansicht zeigt die Visualisierungscontainer mit ihren Visualisierungen an, so wie sie auf der Powerwall angezeigt werden. Die Visualisierungscontainer in der Powerwallansicht haben die gleichen Funktionen wie jene im privaten Raum. Wenn zwei Visualisierungscontainer gekoppelt sind, von denen sich einer im öffentlichen Raum befindet und der andere im privaten Raum, wird die Kopplung über gleiche Rahmenfarbe angezeigt. Denn eine Linie über den Rahmen der Powerwallansicht ist nicht sinnvoll darstellbar.

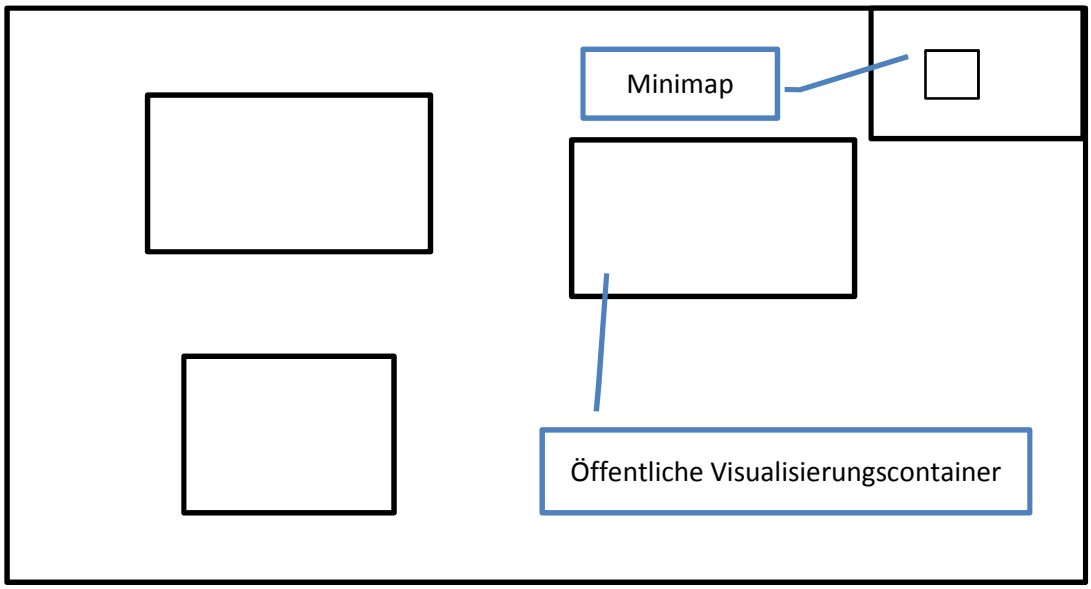


Abbildung 50: Skizze der Powerwallansicht mit drei Visualisierungscontainern

Die Visualisierungscontainer können frei auf der Powerwallansicht verschoben werden aber sie können nicht rotiert werden wie im privaten Raum. Das würde auf der Powerwall keinen Sinn ergeben.

Da die Powerwallansicht meist kleiner ist als die Powerwall und damit meist auch eine niedrigere Auflösung hat, müssen die Visualisierungscontainer auf der Powerwallansicht skaliert werden. Dies hat zur Folge, dass entweder die komplette Powerwall gesehen werden kann oder dass die Visualisierungen im gleichen Detail gesehen werden können wie auf der Powerwall. Daher zeigt die Powerwallansicht einen Bereich der Powerwall an und ist damit eine Art Fenster auf einen Teil der Powerwall, ein „Viewport“. Damit dieser Viewport frei auf der Powerwall skaliert und verschoben werden kann, unterstützt die Powerwallansicht „Pinch“- und „Pan“-Gesten. Das bedeutet mit einem Finger kann die Powerwall virtuell unter dem Viewport verschoben werden. Das nennt sich „Pan“-Geste. Mit zwei Fingern kann die Powerwall virtuell innerhalb des Viewports gestreckt und gestaucht werden. Das nennt sich „Pinch“-Geste.

Um zu sehen wo sich der Viewport der Powerwallansicht auf der Powerwall befindet, hat die Powerwallansicht an der rechten oberen Ecke eine „Minimap“. Die Minimap zeigt die gesamte Powerwall stark verkleinert an. Innerhalb der Minimap wird der Viewport als Rechteck angezeigt. So weiß der Benutzer immer wo er sich auf der Powerwall befindet ohne die Ansicht mit der Powerwall zu vergleichen.

7.1.3 Visualisierungscontainer

Die Visualisierungscontainer (siehe Abbildung 51) enthalten, wie im Konzept beschrieben, beliebige Visualisierungen. Um die Container zu manipulieren werden Gesten, die schon bei Smartphones und Tablets in den normalen Gebrauch übergegangen sind, genutzt. Für Verschieben wird eine Ziehgeste mit einem Finger genutzt. Skalieren wird als Pinch-Geste mit zwei Fingern umgesetzt und Rotieren mit Drehen um eine Achse. Die Gesten können fließend ineinander übergehen, denn wenn die zwei Finger parallel bewegt werden, findet ebenfalls ein Verschieben statt. Die Gesten wurden in Varianten auch von Wobbrock et al. [48] bestätigt.

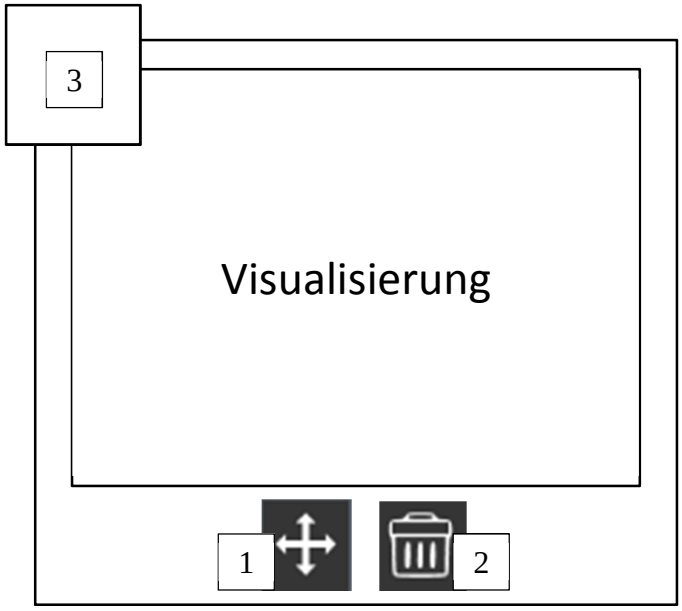


Abbildung 51: Visualisierungscontainer
1) Interaktionsmodus, 2) Löschenknopf, 3) Drag&Drop-Interaktionselement

Um mit der Umgebung interagieren zu können, hat der Visualisierungscontainer an der oberen linken Ecke ein Drag&Drop-Interaktionselement. Dieses Interaktionselement stellt eine Miniaturdarstellung des Containers dar und ist somit eine Repräsentation des Containers für Interaktionen mit anderen Elementen. Mit diesem Element werden die Aktionen Verschieben auf die Powerwall, Duplizieren und Koppeln durchgeführt.

Am unteren Rand des Visualisierungscontainers ist ein Bereich für Steuerelemente. Dieser ist hauptsächlich für Knöpfe mit häufig benötigten Funktionen reserviert. Man kann es als lokale Werkzeugleiste sehen. Die Werkzeugleiste hat momentan zwei Funktionen mit jeweils einem Knopf.

Der rechte Button dient dem Entfernen des Visualisierungscontainers und somit auch seiner Visualisierung von der Benutzeroberfläche. Damit er nicht versehentlich gedrückt werden kann, wird seine Funktion erst bei längerem Halten ausgelöst. Dazu wird eine visuelle Rückmeldung gegeben, wie sich füllender Balken oder Rahmen um den Knopf. Wenn sich der Visualisiercontainer im öffentlichen Raum befindet, wird er laut Konzept auch aus der Analyseumgebung entfernt und daher auch von der Powerwall.

Da Interaktionen des Visualisierungscontainers und der Visualisierung in Konflikt geraten können, muss einer der beiden Priorität gegeben werden. Dazu gibt es den linken Knopf auf dem Visualisierungscontainer. Dieser Knopf dient als Umschalter des „Interaktionmodus“ des Visualisieungscontainers. Der Interaktionsmodus hat zwei Zustände, den „Verschiebemodus“ und den „Bearbeitungsmodus“(siehe Abbildung 52). Der Verschiebemodus gibt dem Visualisierungscontainer die Priorität über die Interaktionen. Das bedeutet der Benutzer kann den Visualisierungscontainer mit seiner gesamten Fläche manipulieren, auch dort wo sich die Visualisierung befindet. Der Bearbeitungsmodus gibt dagegen der Visualisierung die Priorität über die Interaktion auf ihrer Fläche. Daher kann der Visualisierungscontainer nur noch an den Rändern manipuliert werden und Interaktionen innerhalb der Visualisierung werden an diese weitergegeben.

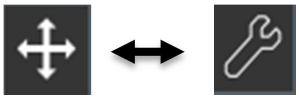


**Abbildung 52: Zeigt die beiden Modi des Knopf für den Interaktionsmodus.
Beim Drücken wird zwischen diesen umgeschaltet.
Der linke Teil zeigt den Knopf im Verschiebemodus, der rechte zeigt ihn im Bearbeitungsmodus.**

Die Interaktionen der Visualisierung werden von der Visualisierung selbst bestimmt. Daher können sie je nach Visualisierung variieren.

Das Wechseln von Visualisierungen wird ebenfalls vom Visualisierungscontainer übernommen. Sie geschieht im Verschiebemodus damit es keine Einschränkungen für die Visualisierung bezogen auf die Interaktionen gibt. Der Wechsel geschieht über längere Berührung der Visualisierung gefolgt von einer Wischgeste. Für die verschiedenen Visualisierungen gibt es eine Vorschau über ein Karussell wie es aus Webanwendungen bekannt ist.

7.1.4 Drag&Drop-Interaktionselement

Das Drag&Drop-Interaktionselement (siehe Abbildung 51) dient der Interaktion zwischen verschiedenen Visualisierungscontainern. Das Element soll als eine Repräsentation des Visualisierungscontainer angesehen werden und kann so an Stelle des Containers selbst mit anderen Elementen der Benutzeroberfläche interagieren. Als Drag&Drop-Element kann das Interaktionselement unabhängig von dem Visualisierungscontainer entnommen und abgelegt werden. Die Hauptfunktionen, die das Interaktionselement ermöglicht, sind die Duplizierung, das Verschieben auf die Powerwallansicht und die Kopplung von Visualisierungscontainern.

Die Duplizierung eines Containers erfolgt durch Verschieben des Interaktionselements in einen freien Bereich im privaten Raum oder auf der Powerwallansicht (siehe Abbildung 53 unten). Falls das Interaktionselement auf die Powerwallansicht verschoben wird, muss der Benutzer über ein Menü entscheiden ob ein Verschieben oder eine Duplizierung stattfinden soll.

Die Kopplung geschieht durch das Ziehen des Interaktionselements auf einen anderen Visualisierungscontainer (siehe Abbildung 53 oben).

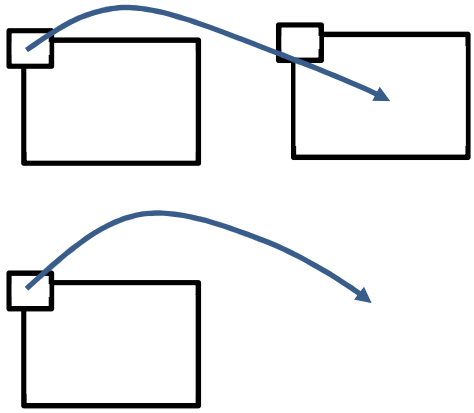


**Abbildung 53: Interaktionen mit dem Drag&Drop-Interaktionselement
oben: Koppeln zweier Visualisierungscontainer
unten: Duplizierung eines Visualisierungscontainers**

Das Drag&Drop-Interaktionselement ist somit kontextabhängig und erlaubt es mehrere Funktionen in sich zu vereinen. Durch klar verständliche Unterschiede beim Kontext der

Interaktionen kann der Benutzer die verschiedenen Funktionen dennoch leicht auseinanderhalten.

7.1.5 Gesteninteraktion

Neben den Standardgesten aus 7.1.3 können auch für andere Interaktionen mit den Visualisierungscontainern Gesten angewendet werden, so auch für die Duplizierung und Kopplung.

Eine mögliche Geste für die Duplizierung wäre ein Auseinanderziehen eines Visualisierungscontainers. Diese Art von Geste, mit jeweils einem Finger ausgeführt, würde aber in Konflikt mit der Skalierungsgeste kommen. Darum wird eine Geste mit jeweils zwei Fingern verwendet, die sich auf dem Visualisierungscontainer voneinander weg bewegen. Ab einem bestimmten Abstand entstehen zwei gleiche Container. Eines der Fingerpaare behält den alten Container und das andere erhält den duplizierten Container. (siehe Abbildung 54 oben)

Das Koppeln der Visualisierungscontainer wird durch eine Variante der vorherigen Geste gelöst. Mit zwei Fingern wird der Visualisierungscontainer fixiert und einem weiteren Finger wird von dem Container eine Ziehgeste auf einen anderen durchgeführt. Nach dem Loslassen der Finger ist die Geste beendet. (siehe Abbildung 54 unten)

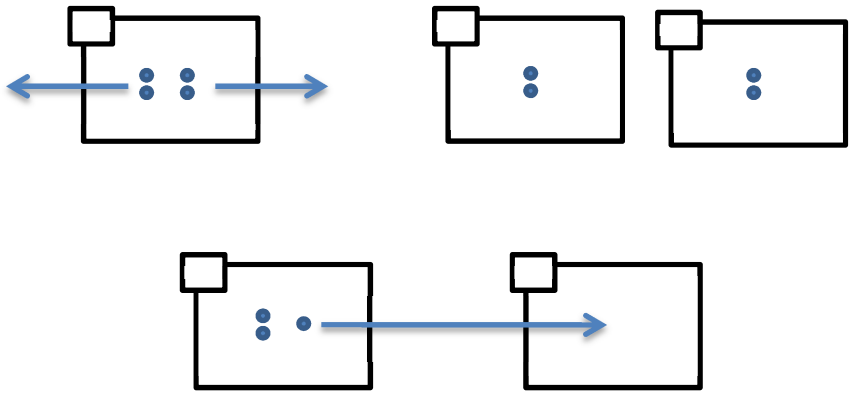


Abbildung 54: Gesteninteraktion mit dem Visualisierungscontainer
oben: Duplizieren mit Geste
unten: Koppeln mit Geste

7.1.6 Tangibleinteraktion

Mit den aus dem Konzept bekannten Interaktionstangibles ergibt sich für den Tabletop eine weitere Variante der Interaktion mit den Visualisierungscontainern.

Um nicht zu viele verschiedene Tangibles zu benötigen wird auch hier eine kontextabhängige Variante entworfen. Dadurch werden die Interaktionstangibles auf zwei minimiert.

Das erste ist das Duplizierungtangible. Es fungiert als eine Art Stempel. Als erstes wird durch Stellen des Tangibles auf einen Visualisierungscontainer dieser registriert. Danach kann der Benutzer durch Stellen desselben Tangibles auf den privaten Raum oder die Powerwallanzeige den Visualisierungscontainer duplizieren. Wie bei einem Stempel kann dies beliebig oft wiederholt werden. Eine Alternative mit zwei verbundenen Tangibles wäre ein Kopieren- und ein Einfügentangible.

Das andere Tangible vereinigt mehrere Funktionen in sich. Die erste Unterscheidung ist beim Registrieren des zugehörigen Visualisierungscontainers.

Wird das Tangible nur kurz auf einen Visualisierungscontainer gestellt, wird der Container für die Dateninteraktion registriert. Das bedeutet, die weitere Interaktion des Tangible hat Auswirkungen auf die Daten des Containers bzw. seiner Visualisierung. Wird das Tangible dann auf einen Visualisierungscontainer gelegt, wird eine Kopplung zwischen dem registrierten und diesem Visualisierungscontainer angelegt. Wenn das Tangible andererseits auf eine freie Region des privaten Raums abgelegt wird, dann wird ein Filterelement für den Visualisierungscontainer angelegt. Damit können die Daten der Visualisierung dieses Containers gefiltert werden (siehe 7.1.7).

Wird das Tangible länger auf einen Visualisierungscontainer gestellt, dann wird der Container daran gebunden. Damit wird aus dem Tangible ein Repräsentionstangible für den Container. Daher kann der Benutzer den Visualisierungscontainer mit dem Tangible manipulieren. Das Tangible bietet die Möglichkeit den Container zu verschieben und zu rotieren. Eine Skalierung ist mit einem einzelnen Tangible nicht möglich und wird daher mit der normalen Geste, wie in 7.1.3 beschrieben, durchgeführt. Zudem kann das Tangible von der Benutzeroberfläche genommen werden, wodurch sich der Visualisierungscontainer ausblendet. Dadurch wird der reale Raum um den Tabletop selbst zur globalen Ablage für Visualisierungscontainer.

Um kurzes von längerem Auflegen des Tangible unterscheiden zu können, benötigt der Benutzer eine visuelle Rückmeldung. Dies kann durch die Animation eines sich schließenden Kreissegments gelöst werden. Nachdem der Kreis vollständig ist, gilt es als langes Auflegen.

7.1.7 Filtern mit Tangibles

Laut Anforderungen und Konzept und entsprechend der Visualisierungspipeline sollen die Daten der Visualisierungen gefiltert werden können. Eine Lösung mit Tangibles erschien attraktiv. Darum wurde ein Filterinteraktionskonzept mit Tangibles erstellt. Das Filterinteraktionskonzept sollte einfach zu verstehen und zu bedienen sein aber gleichzeitig auch möglichst mächtig sein.

Eine Möglichkeit ist die Nutzung von Graphen, deren Knoten Mengen und Kanten Filter darstellen, zu sehen in der Arbeit von Bosch et al. [17] .Diese Methode ist sehr mächtig und die Darstellung ist einfach zu verstehen. Leider ist die Bedienung nicht ganz so einfach. Außerdem verträgt es sich nur bedingt mit den Tangibles als Datenrepräsentation, wie es gewünscht ist. In diesem Fall wären die Tangibles eher Manipulatoren des Graphen.

Daher wurde eine andere Vorgehensweise gewählt. Inspiriert wird das Filterkonzept von den Darstellungen der Mengenlehre. Doch auch diese haben ihre Grenzen z.B. beim Schnitt und eine Umsetzung wäre schwierig. Darum wurde es weiter vereinfacht. Die grundlegenden Operationen bei dem Filterkonzept sind Vereinigung und Verfeinerung. Die eigentlichen Filter werden durch Tangibles erzeugt. Die Tangibles sind dabei Datenrepräsentationstangible. Jedes Tangible repräsentiert ein bestimmtes Datum. Dieses Datum filtert die Ausgangsmenge der gesamten Filteroperation anhand dieses Datums. Als Beispiel, wenn die Ausgangsmenge alle Fixationen sind und das Datum ein Proband, dann entsteht eine Menge mit nur den Fixationen des Probanden. Diese gefilterten Mengen werden schlussendlich über die Grundoperationen zur Ergebnismenge kombiniert. Die Verfeinerung ist dabei im Prinzip nichts anderes als ein Schnitt. Dennoch ist „Verfeinerung“ der bessere Begriff, da die Darstellung und Bedienung des Filterkonzepts im Grunde als Baum funktionieren. Die Vereinigungen sind Nachbarknoten des Baums und Verfeinerungen Kindknoten.

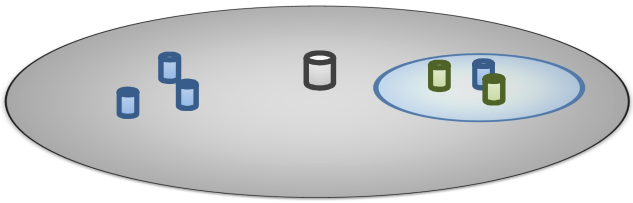


Abbildung 55: Filterkonzept mit Tangibles
Die Zylinder sind die Tangibles. Das graue Tangible stellt den Filterbereich her.

Der Baum schlägt sich auch in der Darstellung nieder. Der Wurzelknoten ist dabei das Bedienungselement auf der die Filterinteraktion stattfindet. Er wird als Kreis dargestellt, zu sehen in Kapitel 7.1.1, und erstellt durch ein Tangible wie in Kapitel 7.1.6 beschrieben. Durch Ablegen eines Tangibles auf dem Kreis wird die Ergebnismenge auf den, durch das Tangible beschriebenen Filter, reduziert. Werden nun weitere Tangibles auf den Kreis gelegt, werden die Filtermengen dieses Tangibles zur Ergebnismenge hinzugefügt. So können Vereinigungen genutzt werden. Dafür nochmal ein Beispiel für Eyetracking, wenn ein Benutzer mehrere Tangibles für Probanden ablegt, dann werden die Fixationen aller dieser Probanden in die Ergebnismenge übernommen. Jeder der Tangibles erzeugt innerhalb des Kreises einen eigenen Kreis, der skaliert werden kann und in den wieder Tangibles abgelegt werden können. Tangibles, die in den Kreis eines anderen Tangible gelegt werden, verfeinern die Filtermenge dieses Tangibles. Das bedeutet in der Filtermenge des übergeordneten Tangibles bleiben nur die Datenelemente, die auch in der Filtermenge des untergeordneten Tangibles vorkommen. Werden mehrere Tangibles innerhalb des Kreises des Tangibles abgelegt, werden diese Filtermengen vorher vereinigt, bevor sie die Filtermenge des übergeordneten Tangibles verfeinern. Das entspricht dem allgemeinen Vereinigen. Das Verfeinern kann beliebig oft wiederholt werden solange der Platz nicht ausgeht. Werden keine Tangibles auf dem Kreis des Wurzelknotens abgelegt findet keine Filterung statt.

Sollen Daten zur Filterung verwendet werden, können sie durch einen Dialog einem Datenrepräsentationstangible zugeordnet werden.

Das Filterkonzept ist relativ simpel und kann mit den Tangibles schnell bedient werden. Der Benutzer kann sehr schnell in die Bedienung und Darstellung eingeführt werden. Die Mächtigkeit des Filterkonzeptes wurde nicht näher untersucht, doch beim Ausprobieren mit einigen Beispielen scheinen die meisten Möglichkeiten abgedeckt zu sein.

7.1.8 Persönliche Ablage

Eine persönliche Ablage wie sie im Konzept beschrieben wird, soll ebenfalls umgesetzt werden. Auch für sie können Tangibles eingesetzt werden. Die Idee ist, dass jeder Benutzer ein Tangible in Form einer Id-Karte erhält. Wenn die Karte auf einen freien Bereich des privaten Raums gelegt wird, öffnet sich eine Ablage für Visualisierungscontainer. Die Ablage

ist eine Drag&Drop-Ablage. Das bedeutet, dass Visualisierungscontainer mit dem Drag&Drop-Interaktionselement in die Ablage überführt werden können.

Die Erweiterung dieser Idee mit persönlichen Geräten wie z.B. Tablets funktioniert ganz ähnlich. Die Id-Karte öffnet in diesem Fall keinen einfachen Container, sondern zeigt den privaten Raum des persönlichen Geräts an. Voraussetzung ist natürlich, dass das Gerät in die Analyseumgebung eingebunden ist. Auf diese Weise können Visualisierungscontainer direkt in den privaten Raum des persönlichen Gerätes abgelegt werden. Somit können die Container sowohl abgelegt werden, als auch zur weiteren Untersuchung auf das persönliche Gerät übertragen werden.

7.2 Vorstudie

Dieses Kapitel beschreibt die Vorstudie und deren Ergebnisse. Als erstes wird die Durchführung der Studie und deren Aufgaben beschrieben. Danach wird das Verhalten der Probanden beschrieben und die dabei gemachten Beobachtungen gekürzt wiedergegeben. Die Ideen und Aussagen aber nicht bewertet oder zensiert. Die Studie wurde mit 5 Probanden durchgeführt, die größtenteils auf dem Gebiet der Informatik arbeiten. Zum Schluss werden die Beobachtungen zusammengefasst und analysiert.

7.2.1 Beschreibung

Die Studie nutzt „Paperprototyping“ [47] um das Konzept der Benutzeroberfläche und der Interaktionen den Probanden vorzustellen. Beim Paperprototyping werden abstrahierte Elemente der Benutzeroberfläche in Papierform hergestellt. Der Proband kann dann Interaktionen anhand der Papierelemente ausführen und bewerten. Des Weiteren wird in Anlehnung an das „User Defined Design“ [48] versucht durch Anregungen der Probanden neue Ideen für das Konzept zu ermitteln. Dabei kann der Proband innerhalb des Papierprototyps, Elemente der Benutzeroberfläche zeichnen und neue Interaktionen beschreiben. Die Ergebnisse der Studie sind rein qualitativ.

Der Aufbau besteht aus einem ca. DIN A 1 großes weißes Blatt Papier, das auf einen Tisch oder einer ähnliche Erhöhung wie z.B. Kartons gelegt wird. Dieses Papier stellt den Tabletop dar. Auf dem Papier liegt ein brauner Papprahmen, der innen ungefähr die Größe des weißen Papiers hat. Er stellt den Rand des Tabletop-Displays dar. Am Rand des Tischs wird eine Kamera aufgestellt, um die Studie zu dokumentieren. Die Powerwall wird durch ein Whiteboard oder eine Wand repräsentiert. (zu sehen in Abbildung 56)

Abbildung 56: Links: Studienszenario mit Position der Kamera
Rechts: Sicht der Kamera zur Dokumentation der Studie

Durchgeführt wurde die Studie mit fünf Probanden. Vor Beginn der Studie wurden demografische Aspekte sowie Vorkenntnisse anhand einer Umfrage (siehe Anhang A: Fragebogen) ermittelt.

Tabelle 1: Teilnehmer der Studie

	Geschlecht	Alter	Touch-Geräte	Nutzungsdauer pro Woche (h)	Erfahrung mit Tabletops	Erfahrung mit Tangibles
Proband 1	Männlich	24	Ja	>14	Nein	Nein
Proband 2	Männlich	25	Nein	0	Nein	Nein
Proband 3	Männlich	26	Ja	15	Nein	Nein
Proband 4	Männlich	24	Ja	10	Ja	Nein
Proband 5	Weiblich	23	Ja	48	Nein	Nein

In der Studie wird für jede Funktionalität den Probanden anhand eines Beispiels eine Aufgabe gestellt, für die sie mit den zur Verfügung stehenden Mitteln eine Interaktion entwickeln sollen. Als Zweites oder wenn der Proband keine Idee für eine Interaktion hat, wird eine eigene Interaktion vorgestellt, die der Proband dann objektiv und subjektiv bewerten kann. Zusätzlich werden Probanden die Ideen von vorherigen Probanden vorgestellt die ihren eigenen ähneln, so dass der Proband seinen Vorschlag noch verbessern kann. Der Proband wird anfangs darauf hingewiesen, dass er durch lautes Nachdenken seinen Gedankengang verdeutlichen soll.

Um das Prinzip abstrakt zu halten, wurden die Fenster der Visualisierungen im Lösungsansatz als „Visualisierungscontainer“ bezeichnet. In der Studie wird dagegen der Begriff „Fenster“ verwendet, um die Probanden nicht zu verwirren.

Die Aufgaben in der Vorstudie wurden anhand der Vorüberlungen des Lösungskonzepts entwickelt. Eyetracking-spezifische Funktionalitäten wurden verallgemeinert oder weggelassen. Mögliche Interaktionen für folgende Aufgaben und zugehörige Funktionalität wurden in der Studie untersucht. Die Aufgaben sind: Grundsätzliche Interaktionen mit Fenstern auf berührungsempfindlichen Displays, Schieben eines Fensters auf die Powerwall, Verschieben von Fenstern auf der Powerwall, Duplizieren eines Visualisierungsfensters, Löschen eines Visualisierungsfensters, Koppeln von zwei Visualisierungsfenstern, Entkoppeln von Visualisierungsfenstern, Nutzung einer private Ablage für Visualisierungen, Nutzung einer globalen Ablage, ein Filterkonzept mit Tangibles erstellen, einen Filterbereich für Tangibles erzeugen und Wechseln der Visualisierung im Fenster.

1. **Grundsätzlich Interaktion mit Fenstern in berührungsempfindlichen Displays**
   Um einen Überblick über die grundsätzliche Fähigkeiten des Probanden mit berührungsempfindliche Hardware zu bekommen und einen Vergleich mit existenten Systemen zu ermöglichen, wird dem Proband die Aufgabe gegeben einfache Interaktionen durchzuführen. Der Aufbau für diese Aufgabe ist eine freie Fläche mit einem Papierelement, das ein Fenster darstellt. Der Aufbau wird dem Probanden erklärt und er wird aufgefordert das Fenster zu verschieben, zu skalieren und zu rotieren.

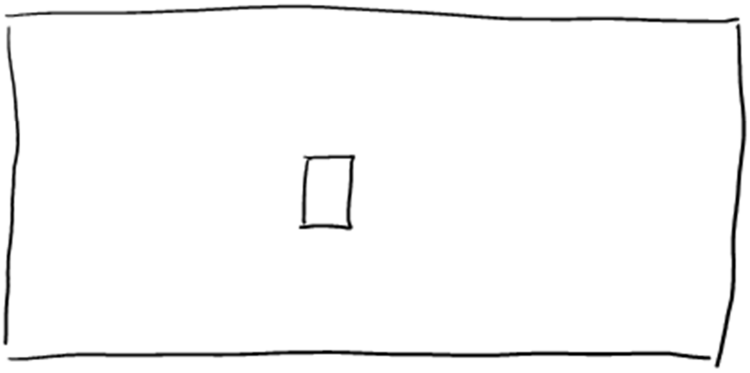


Abbildung 57: Ausgangssituation bei Aufgabe 1

2. **Schieben eines Fenster auf die Powerwall**
   Der Proband wird gebeten das Fenster auf die Powerwall zu schieben. Dabei wird darauf geachtet, dass dem Probanden nicht vorgegeben wird ob das Fenster kopiert oder verschoben wird. Der Aufbau zu Beginn der Aufgabe ist der gleiche wie in Aufgabe 1. Mit dem Unterschied, dass eine Tafel oder ähnliches dem Proband als Powerwall vorgestellt wird. Nachdem der Proband seine Interaktionsidee vorgestellt hat, wird er gefragt wie der Endzustand aussieht.

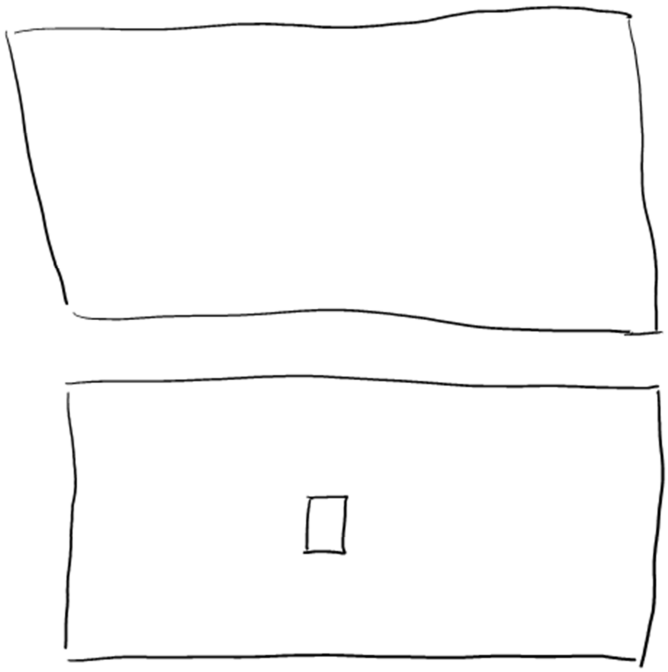


Abbildung 58: Ausgangssituation von Aufgabe 2
Oben ist die Powerwall und unten der Tabletop zu sehen.

3. **Verschieben des Fensters auf der Powerwall**
   Der Startaufbau dieser Aufgabe geht direkt aus der vorherigen hervor. Der Proband wird gefragt wie er das Fenster auf der Powerwall verschieben würde.

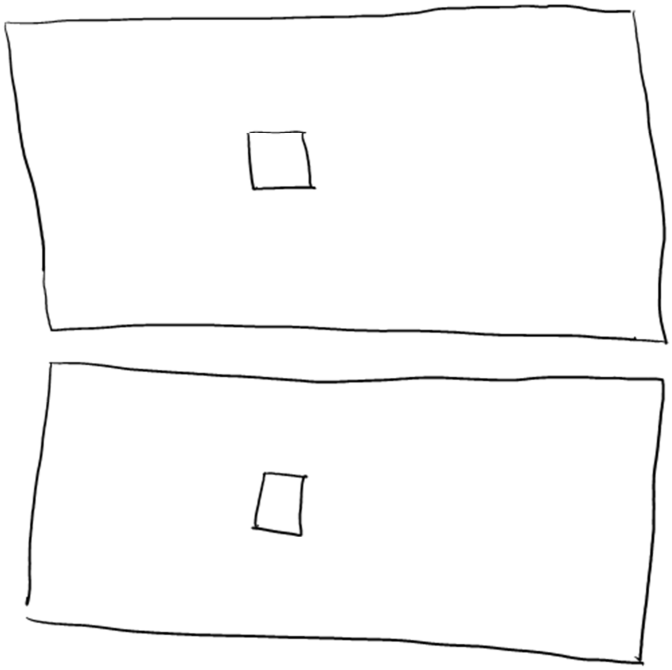


Abbildung 59: Ausgangssituation von Aufgabe 3, falls der Proband das Fenster kopiert.

Dabei sollte der Proband die Erkenntnis haben, dass keine Interaktion mehr möglich ist, falls das Fenster auf die Powerwall verschoben wurde (siehe Abbildung 60). Daraufhin soll der Proband eine Lösung für das Problem finden.

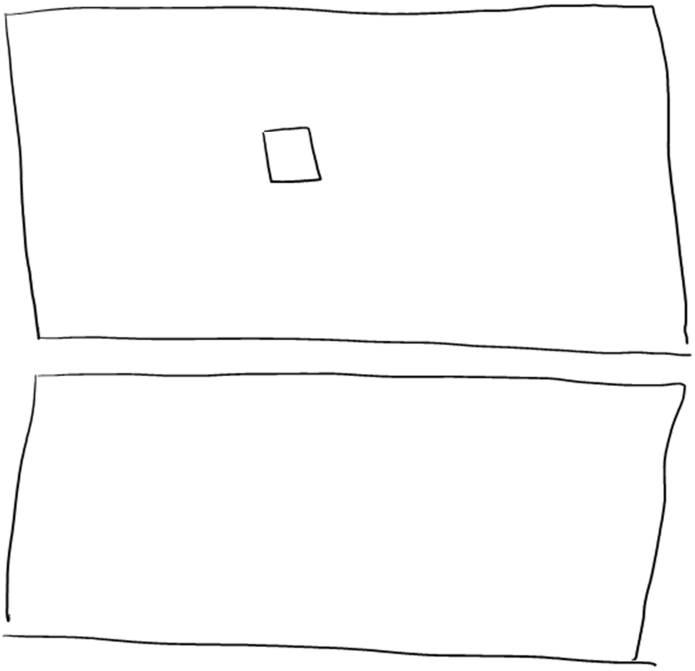


Abbildung 60: Ausgangssituation von Aufgabe 3, falls der Proband das Fenster verschiebt.

Wenn keine Lösung gefunden wird oder eine andere als die Powerwallansicht wird diese dem Proband vorgestellt. Die Powerwallansicht ist ein frei verschiebbares Fenster, das einen skalierten Bereich der Powerwall anzeigt. Der Proband wird nach seiner Meinung gefragt und es wird nochmals die Aufgabe gestellt, das Fenster auf die Powerwall zu schieben. Im Anschluss wird er aufgefordert, das Fenster mit Hilfe der Powerwallanzeige auf der Powerwall zu verschieben. In Hinblick darauf soll der

Proband festlegen wann er auf die Powerwall schauen würde und wann auf den Tabletop.

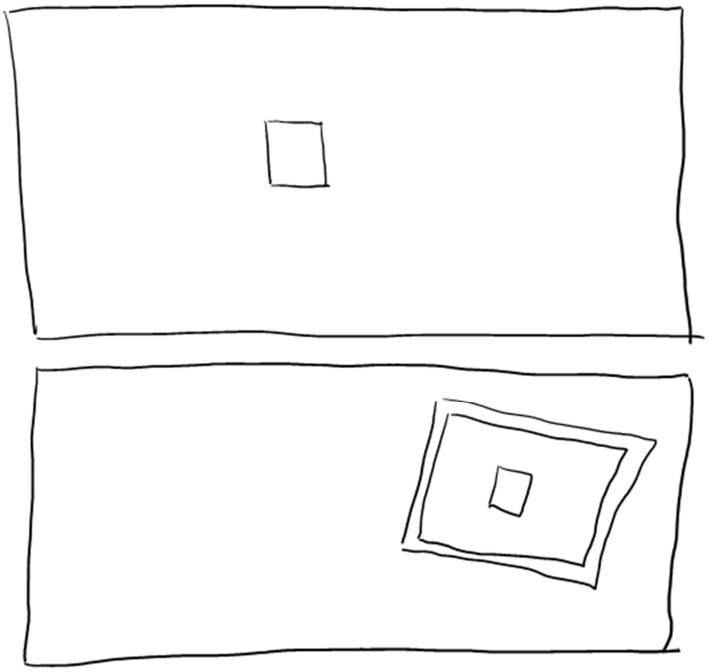


Abbildung 61: Tabletop mit Powerwallansicht und einem Fenster auf der Powerwall

4. **Duplizieren eines Visualisierungsfensters**
   Ausgangsaufbau bei dieser Aufgabe ist ein Fenster auf dem Tabletop (siehe). Der Proband wird aufgefordert das Fenster zu duplizieren. Davor wird der Proband darauf hingewiesen, dass er alles einsetzen kann was ihm einfällt, zum Beispiel: Steuerelemente, Gesten oder Tangibles. Wenn der Proband eine Geste vorschlägt, können mögliche Konflikte mit z.B. der Skalierengeste besprochen werden. Nachdem der Proband alle seine Ideen vorgestellt hat, wird dem Fenster eine Minidarstellung an der linken oberen Ecke hinzugefügt. Danach wird der Proband gefragt wie dieses Element zu verstehen sein könnte. Wenn der Proband keine Idee hat, wird das Stichwort Drag&Drop eingeworfen. Falls der Proband den Begriff nicht kennt, wird das Verfahren beschrieben. Im Falle, dass der Proband das Steuerelement versteht, soll er die Interaktion mit diesem andeuten. Falls nicht, wird beschrieben, dass das Steuerelement eine Miniaturrepräsentation des Fenster darstellt und an eine neue Position verschoben werden kann, um das Fenster zu duplizieren. Wenn der Proband bis zu diesem Punkt noch keine Interaktion mit Tangibles vorgeschlagen hat, wird er aufgefordert dies noch zu tun.

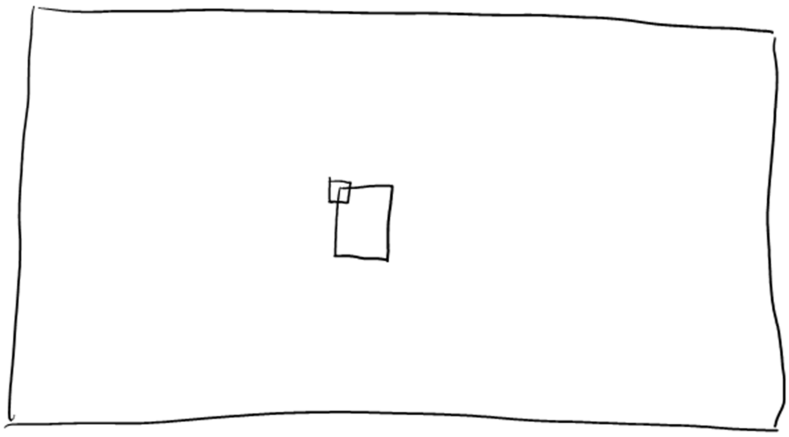


Abbildung 62: Fenster mit Miniaturansicht(Drag&Drop-Interaktionselement) auf dem Tabletop

5. **Löschen eines Visualisierungsfensters**
   Bei dieser Aufgabe soll ein Fenster mit einer Visualisierung gelöscht werden. In der Ausgangssituation gibt es wieder ein Fenster (siehe Abbildung 63). Der Proband wird gefragt wie er das Fenster entfernen würde. Die angegebenen Lösungen können daraufhin im Hinblick auf Mehrbenutzerbetrieb diskutiert werden. Als nächstes wird dem Probanden die Variante mit einem Knopf am Fenster angeboten. Der Knopf hat einen Mülleimer als Abbildung.

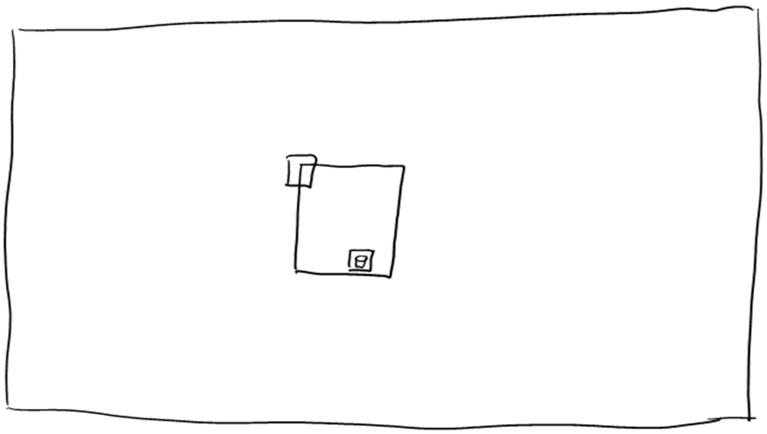


Abbildung 63: Fenster mit Löschenknopf

6. **Koppeln von zwei Visualisierungsfenstern**
   Bei dieser Aufgabe soll eine Interaktion für das Koppeln von zwei Visualisierungsfenstern gefunden werden. Der Proband bekommt als erstes eine kurze Erklärung für welche Anwendungsfälle das Koppeln benötigt wird. Die Ausgangsituation sind diesmal zwei Fenster mit den entsprechenden Drag&Drop-Interaktionselementen. Nachdem der Proband seine Idee vorgeschlagen hat, wird wieder die Interaktion mit dem Drag&Drop- Interaktionselement vorgestellt, falls der Proband die Interaktion nicht schon gezeigt hat. Das Element kann auf das andere Fenster geschoben werden, um eine Kopplung zu erzeugen. Falls der Proband noch keine Interaktion mit tangibles vorgeschlagen hat, wird er an dieser Stelle aufgefordert eine vorzuschlagen. Nachdem alle Interaktionen besprochen sind soll der Proband zeigen wie er eine Kopplung darstellen würde.

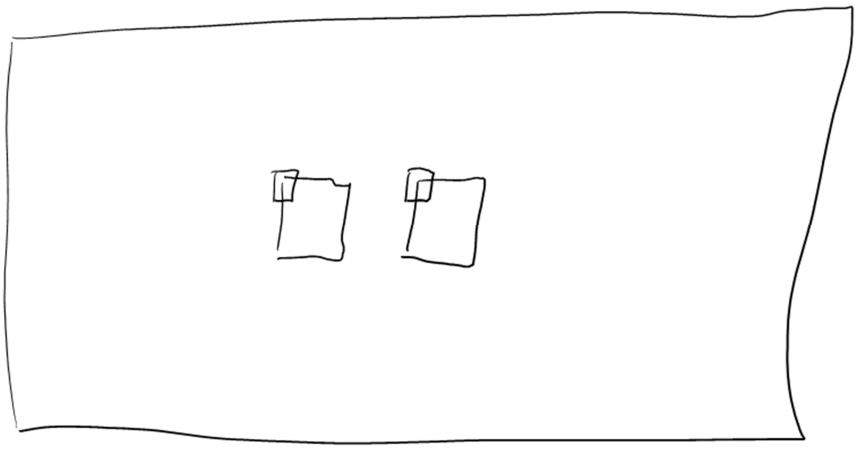


Abbildung 64: Zwei Fenster mit Drag&Drop-Interaktionselementen

7. **Entkoppeln von Visualisierungsfenstern**
   Nachdem der Proband vorgeschlagen hat wie die Kopplung auszusehen hat, soll er als nächstes eine Interaktion angeben, die die Kopplung aufhebt. Er kann dabei von seiner eigenen Darstellung der Kopplung ausgehen. Wenn der Proband mit seinen Vorschlägen fertig ist, wird ihm eine Variante mit Linien als Kopplungsdarstellung und Knöpfen, um die Kopplung aufzuheben, vorgestellt (siehe Abbildung 65).

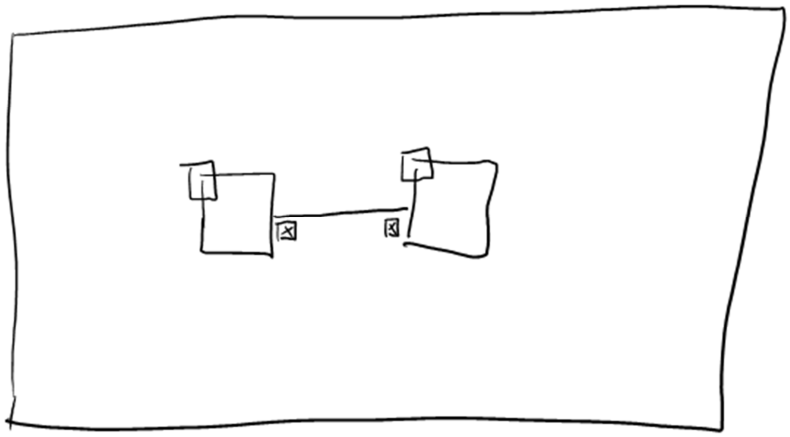


Abbildung 65: Gekoppelte Fenster mit Linie und Knöpfen

8. **Private Ablage für Visualisierungen**
   Die private Ablage ist eine Interaktion, die direkt aus dem Interaktionskonzept mit Tangibles hervorgeht. Ein Tangible erzeugt hier einen Ablagecontainer in den Visualisierungsfenster und optional andere Elemente abgelegt werden können. Das Tangible ist als Identifikationskarte gestaltet, um dem Benutzer die private Bedeutung zu suggerieren. Der Ursprungszustand bei dieser Aufgabe ist ein Fenster auf dem Tabletop. Der Proband bekommt die Id-Karte und ihm wird erklärt, dass es sich dabei um ein Tangible handelt. Der Proband kann die Karte auf den Tabletop legen, worauf die Ablage erscheint (siehe Abbildung 66). Dem Proband wird erklärt, dass es sich dabei um eine Ablage für Elemente des Systems, insbesondere Visualisierungsfenster, handelt. Der Proband soll daraufhin ein Fenster ablegen.

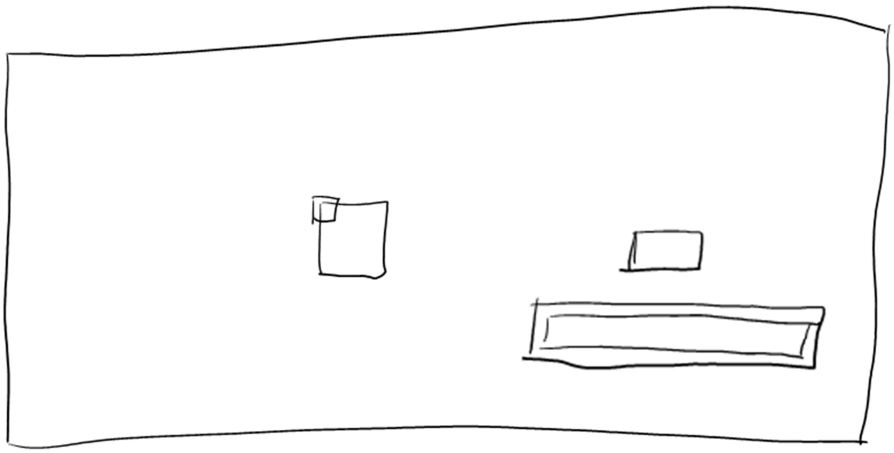


Abbildung 66: Id-Karte erzeugt private Ablage für Fenster(rechts)

9. **Globale Ablage für Visualisierungen**
   Um eine einfachere Ablage zu haben, die auch gemeinsam genutzt werden kann, soll zusätzlich eine globale Ablage entworfen werden. Darum wird der Proband gefragt wie er sich in Anlehnung an die private, eine globale Ablage vorstellen könnte.
   In Anlehnung daran wird danach das Prinzip vorgestellt ein Tangible direkt an ein Visualisierungsfenster zu binden. Das bedeutet das Fenster wird danach direkt durch das Tangible positioniert und rotiert. Außerdem ist das Fenster nur sichtbar, wenn das Tangible auf dem Tabletop liegt. Daher kann es ebenfalls als Ablage genutzt werden.

10. **Filterkonzept mit Tangibles**
    Für die Analyseumgebung sollte eine Möglichkeit gefunden werden mit Tangibles Daten einfach und intuitiv zu filtern. Das Filtersystem sollte daher einen möglichst guten Kompromiss aus Mächtigkeit und einfacher Bedienung beinhalten.
    Ausgangsituation bei dieser Aufgabe ist ein runder Bereich auf dem Tabletop (siehe Abbildung 67), der als Filterinteraktionsbereich für die Tangibles dienen soll und einer Visualisierung zugeordnet ist. Vorgegeben sind weiterhin einige Tangibles, denen Datenelementen aus der Gesamtdatenmenge zugeordnet sind. Die Datenelemente sind dann die Filterkriterien. Mit diesen Vorgaben und einem Datenbeispiel, aus dem Umfeld des Probanden, soll der Proband ein Filterinteraktionskonzept entwickeln. Wenn der Proband keine eigenen Ideen mehr hat, wird das eigene Konzept vorgestellt.

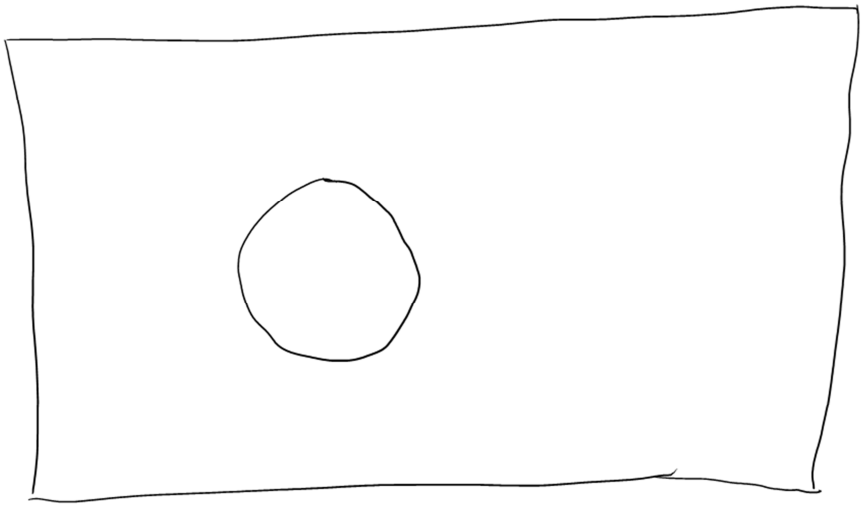


Abbildung 67: Kreis als Filterbereich

11. **Filterbereich für Tangibles aufrufen**
    In Anschluss an Aufgabe 10 soll bei dieser Aufgabe, nachdem der Proband von dem Filterkonzept gehört hat, die Interaktion gefunden werden, um einen Filterinteraktionsbereich für eine Visualisierung bzw. dessen Fenster zu erstellen. Ausgangsituation ist wieder ein einzelnes Visualisierungsfenster. Der Proband wird gefragt wie er einen Filterbereich erstellen würde. Wenn der Proband fertig ist, wird angeboten den Filterbereich mit einem Tangible zu öffnen.

12. **Fenster-/Visualisierungs-Interaktionsmodus**
    Es gibt zwei Interaktionsmodi für die Fenster. Im Verschiebemodus kann nur mit dem Fenster interagiert werden, im anderen auch mit der Visualisierung. Auf diese Weise kann der Benutzer die ganze Fensterfläche nutzen, um das Fenster zu verschieben oder zu skalieren. Die Ausgangssituation ist ein Fenster mit einem Knopf. Der Proband wird als erstes nochmal gefragt wie er verschieben oder skalieren würde. Wenn der Proband die Visualisierungsfläche nutzt, wird er darauf angesprochen, dass die Visualisierung möglicherweise auch Interaktionen zulässt. Danach wird der Proband auf den Knopf aufmerksam gemacht. Der Knopf hat ein Steuerkreuz abgebildet, wird er gedrückt dann kommt wird es ein Schraubenschlüssel. Der Knopf zeigt den Status des Interaktionsmodus an. Falls der Proband die Bedeutung nicht versteht, wird das Konzept erklärt. Der Steuerkreuzknopf zeigt den Verschiebenmodus an und der Schraubenschlüsselknopf zeigt den Bearbeitenmodus an. Beim Drücken dient der Knopf als Umschalter.

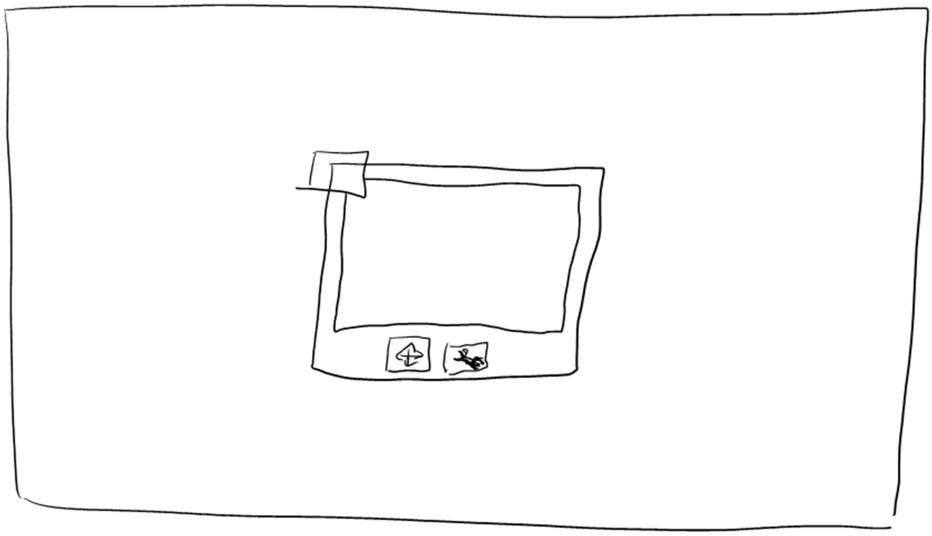


Abbildung 68: Fenster mit den beiden Modi des Knopfs

13. **Wechseln der Visualisierung im Visualisierungsfenster**
    Eine weitere nötige Interaktion ist das Wechseln der Visualisierung im Fenster. Dazu ist wieder ein Fenster Ausgangspunkt dieser Aufgabe. Der Proband wird gefragt, wie er die Visualisierung in diesem Fenster wechseln würde. Wenn er keine Geste nennt, wird explizit nachgefragt.

Es wurde angestrebt die Aufgaben der Studie in der angegebenen Reihenfolge durchzuführen. Allerdings sind sie manchmal ineinander übergegangen oder der Proband hat sie zu einem früheren Zeitpunkt schon beantwortet.

7.2.2 Proband 1

Proband 1 ist männlich und 24 Jahre alt. Er studiert Informatik und hat Erfahrung mit Convertible Notebooks und Smartphones.

Der Proband nutzt für die Interaktionen Verschieben, Skalieren und Rotieren die Standardgesten wie in Kapitel 7.1.3 beschrieben.

Für das Steuern der Powerwall legt der Proband einen extra Bereich auf dem Tabletop fest. Um ein Fenster auf der Powerwall anzuzeigen, würde der Proband das Fenster auf diesen Bereich schieben. Dabei erwartet der Proband nicht, dass diese den Bereich der Powerwall überlappen können, sondern dass sie direkt übertragen werden. Die vorgestellte Powerwallansicht findet der Proband sehr gut, da sie sich mit seiner Idee deckt. Beim Steuern der Powerwall über die Powerwallansicht würde er hauptsächlich auf diese schauen.

Der Proband schlägt für das Duplizieren eines Fensters eine Geste vor. Die Geste entspricht der Standardgeste für Skalieren. Das ergibt einen Konflikt der beiden Gesten. Der Proband weiß keine Lösung, um diesen Konflikt zu umgehen. Der Vorschlag zwei statt einem Finger einer Hand zu nutzen wird angenommen. Tangibles würde der Proband in dieser Situation auf die gleiche Weise nutzen wie die Geste. Eine andere Idee dazu hat er nicht. Das Drag&Drop-Interaktionselement versteht der Proband auch nach einiger Erklärung nicht wirklich. Auch ist er von der Idee nicht sehr überzeugt.

Um ein Fenster zu löschen, würde der Proband es auf den Rand ziehen. Den Knopf zum Löschen findet er nicht gut.

Für die Kopplung von Fenstern nutzt der Proband ebenfalls eine Geste. Dabei würde er den Container mit zwei Fingern festhalten und mit einem Finger eine Linie zu dem anderen Fenster ziehen. Die Kopplung würde der Proband als Linie zwischen den beiden Fenstern darstellen. Das Entfernen der Kopplung würde der Proband als Geste lösen, bei der bildlich mit einem Finger die Linie durchtrennt wird (siehe Abbildung 69). Ein Knopf ist auch hier unerwünscht.

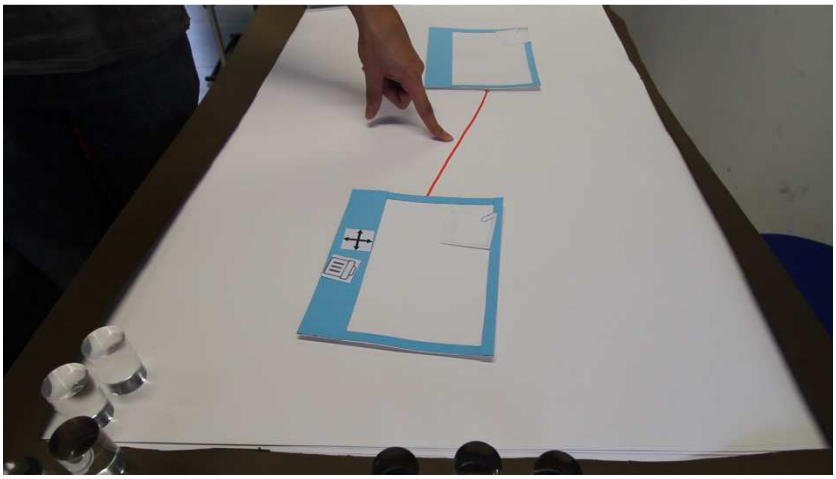


Abbildung 69: Entkoppeln zweier Fenster durch Durchstreichen der Linie

Die persönliche Ablage wurde bei kurzer Erklärung und der Simulation mit Papier vom Probanden nicht verstanden. Nachdem das Prinzip ausführlich erklärt und vorgeführt wird, versteht der Proband die Funktion. Der Proband ist allerdings von der Idee nicht überzeugt und würde eher eine globale Ablage nutzen.

Die globale Ablage würde der Proband als Randleiste umsetzen. Diese kann mit einer Geste aus dem Rand herausgezogen werden. Das Ablegen einer Visualisierung funktioniert analog zum Übertragen auf die Powerwallansicht. Der Vorschlag ein Tangible an ein Fenster zu binden um das Fenster dann ablegen zu können wurde anfangs nicht begriffen. Nach einer ausführlichen Erklärung wird die Idee aber als gut bewertet.

Als Filterinteraktionskonzept schlägt der Proband boolesche Ausdrücke vor. Diese können mit speziellen Tangibles gebaut werden. Jedes der Tangibles ist entweder eine Menge oder ein Operator. Der Vorschlag des Filterkonzepts aus dieser Arbeit wurde schnell verstanden und sehr positiv bewertet.

Der Interaktionsmodus bei den Fenstern wurde zwar nicht auf Anhieb verstanden, aber nach kurzer Erklärung ist der Proband von dessen Funktionalität überzeugt.

7.2.3 Proband 2

Proband 2 ist männlich und 25 Jahre alt. Er studiert Softwaretechnik mit der Vertiefung in Visualisierung und Interaktive Systeme. Er hat kaum Erfahrung mit berührungsempfindlichen Geräten.

Dieser Proband nützt die Standardgesten für Verschieben, Skalieren und Rotieren der Fenster.

Um die Powerwall zu steuern, schlägt der Proband eine „Bild in Bild“-Umschaltung vor, die es dem Benutzer ermöglicht zwischen der lokalen Ansicht und der Powerwall umzuschalten, ähnlich wie es bei Fernsehern der Fall ist. Verschieben würde er das Fenster indem er es in Richtung der Powerwall schiebt. Der Vorschlag der Powerwallansicht aus dieser Arbeit gefällt dem Probanden auch sehr gut. Hier würde er das Fenster direkt auf die Ansicht verschieben (siehe Abbildung 70). Dass sich ein Fenster mit der Ansicht überlappen kann, hält er für unnötig. Der Proband erkennt, dass der Übergang zwischen dem privaten Raum und der Powerwallansicht schwierig ist und ein wohlüberlegtes Feedback benötigt. Daraufhin hat er einen neuen Einfall. Die Berührungspunkte beim Schieben des Fensters können als Bezugspunkte genommen werden. Sobald sich diese auf der Powerwallansicht befinden, wird das Fenster auf die Powerwallansicht verschoben, statt im privaten Raum zu sein. Bei der Steuerung der Powerwall mit der Ansicht würde der Proband in den meisten Fällen auf die Powerwallansicht schauen. Auf Nachfrage bestätigt der Proband, dass er auf die Powerwall schauen würde, wenn er etwas vorführe.

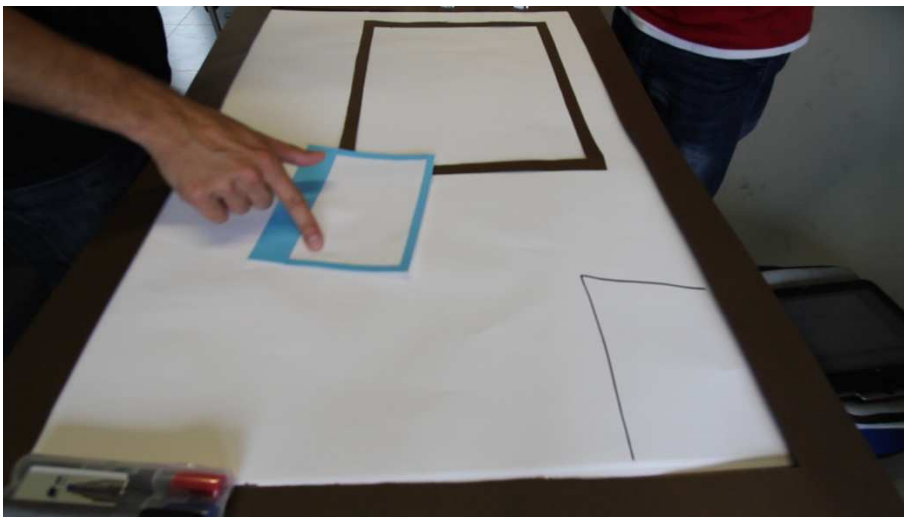


Abbildung 70: Schieben eines Fensters auf die Powerwallansicht

Das Duplizieren von einem Fenster könnte sich der Proband mit einer Geste vorstellen. Die Geste wird mit beiden Händen ausgeführt und zieht das Fenster auseinander. Um die Geste von der Skalierung zu unterscheiden schlägt der Proband die Nutzung von drei Fingern vor. Besser findet der Proband aber das Duplizieren über einen Knopf. Dieser erzeugt an einer beliebigen Stelle ein Duplikat des Fensters. Das Drag&Drop-Interaktionselement wird, nachdem es erklärt wurden, vom Probanden als nicht intuitiv bezeichnet. Daraufhin schlägt der Proband vor, dass der Knopf für die Duplizierung eine Drag&Drop-Interaktion auslöst. Diese erzeugt eine Vorschau des duplizierten Fensters, das frei auf dem Tisch positioniert werden kann. Für die Duplizieren-Interaktion mit Tangibles schlägt der Proband dieselbe Abfolge wie bei der Geste vor nur, dass statt den Händen ein Tangible genutzt wird. Die Duplizierung per Tangible durch Stempel-Interaktion wird akzeptiert.

Um ein Fenster zu entfernen schlägt der Proband vor, das Fenster auf einen Mülleimer zu ziehen oder einen Knopf zum Löschen anzubringen. Er bevorzuge aber den Knopf, da das Ziehen des Fensters umständlich ist.

Das Koppeln zweier Fenster würde der Proband analog zur Duplizierung ebenfalls mit einem Knopf lösen, der eine Drag&Drop-Operation auslöst. Mit der Drag&Drop-Operation kann das zweite Fenster ausgewählt werden. Für die gleiche Funktion mit Tangibles hat der Nutzer keine Idee. Das kombinierte Tangible mit der Duplizierung gefällt dem Benutzer nicht so gut. Die Kopplung zweier Fenster würde der Proband über Farbcodierung anzeigen. Zwei gekoppelte Fenster haben z.B. die gleiche Rahmenfarbe. Die Kopplung über eine Verbindungslinie findet der Proband nicht ideal, weil diese verdeckt werden könnte. Das Entkoppeln würde der Proband über einen Knopf umsetzen.

Die persönliche Ablage wird vom Probanden nach kurzer Erklärung verstanden. Der Proband findet sie gut. Um Fenster in die Ablage zu verschieben, würde der Proband gleich vorgehen wie beim Verschieben in die Powerwallansicht.

Das Binden eines Tangibles an einen Visualisierungscontainer findet der Proband gut. Allerdings merkt er an, dass bei den Tangibles mit kombinierten Funktionen dann die Duplizierung und Kopplung unklar wird.

Für die Filterung mit Tangibles hat der Proband keine Idee. Nach kurzer Überlegung, schlägt der Proband eine Interaktion analog zur Mengenlehre vor. Er kann aber keinen nützlichen Umgang mit den Tangibles finden. Das Filterkonzept aus den Vorüberlegungen gefällt dem Probanden sehr gut. Allerdings merkt er an, dass diese Bedienung für Wissenschaftler möglicherweise nicht mächtig genug ist.

Nach kurzer Einweisung ist dem Proband der Konflikt zwischen Interaktion mit dem Fenster und der Visualisierung innerhalb klar. Der Interaktionsmodus wird erst nach Erklärung verstanden. Dabei ist der Proband anfangs skeptisch, nach einigen Beispielen ist er aber überzeugt.

7.2.4 Proband 3

Der 3. Proband ist männlich und 26 Jahre alt. Er studiert Softwaretechnik und hat längere Erfahrung mit Smartphones.

Der Proband nutzt Standardgesten für das Rotieren, Verschieben und Skalieren.

Für das Verschieben eines Fensters auf die Powerwall schiebt der Proband das Fenster in Richtung der Powerwall aus dem Rand. Das Fenster ist danach nicht mehr auf dem Tabletop sondern nur noch auf der Powerwall zu sehen. Auf Anfrage, wie der Proband nun das Fenster

auf der Powerwall manipulieren würde, erkennt er das Problem. Der Proband schlägt keine Powerwallansicht vor und hat auch sonst keine Idee wie man die Fenster komplett steuern könnte. Von der Powerwallansicht ist der Proband begeistert. Das Schieben eines Fensters auf die Powerwall würde der Proband durch Verschieben des Fensters in die Powerwallansicht lösen. Die Idee von Proband 2 den Berührpunkt des Benutzers als Bezugspunkt zu verwenden, findet der Proband gut.

Für das Duplizieren von Fenstern schlägt der Proband zwei Gesten vor. Die erste ist das Auseinanderziehen von Fenstern wie es auch schon Proband 1 und Proband 2 vorgeschlagen haben. Dabei schlägt er jeweils zwei Finger pro Hand vor, damit es keinen Konflikt zu anderen Gesten gibt. Die zweite Geste ist ein symbolisches Durchschneiden des Fensters mit einem Finger. Der Proband führt es vor indem er einen Finger vom oberen Rand zum unter Rand des Fensters zieht. Außer den Gesten schlägt der Proband Copy&Paste mit Menüs vor. Dabei könne er sich eine Art Kontextmenü vorstellen, das auf den Fenstern und dem freien Bereich des Tabletops geöffnet werden kann. Die Funktion des Drag&Drop-Interaktionselements versteht der Proband nach kurzer Erklärung. Er empfindet das Element allerdings als störend. Die Idee Drag&Drop-Elemente als Repräsentation des Fensters zu nutzen findet er allerdings sehr gut. Er schlägt vor ein Drag&Drop-Element durch einen Button zu erstellen, das dann im Fenster erscheint. Dieses Element kann dann wie das Drag&Drop-Interaktionselement verwendet werden(siehe 7.1.4).

Beim Löschen bevorzugt der Proband das Ziehen des Fensters auf einen Papierkorb oder den Rand. Ein Löschbutton könnte versehentlich gedrückt werden.

Zum Koppeln könnte sich der Proband eine Zusammenziehgeste mit Drag&Drop-Elementen oder das Ziehen eines Drag&Drop-Elements auf ein anderes Fenster vorstellen. Auch das Verbinden von Konnektorpunkten (siehe Abbildung 71) an den Fenstern ist möglich. Ebenso ist die Lösung durch ein Koppeltangible vorstellbar. Er hält jedoch die Tangible-Lösungen nicht für intuitiv.

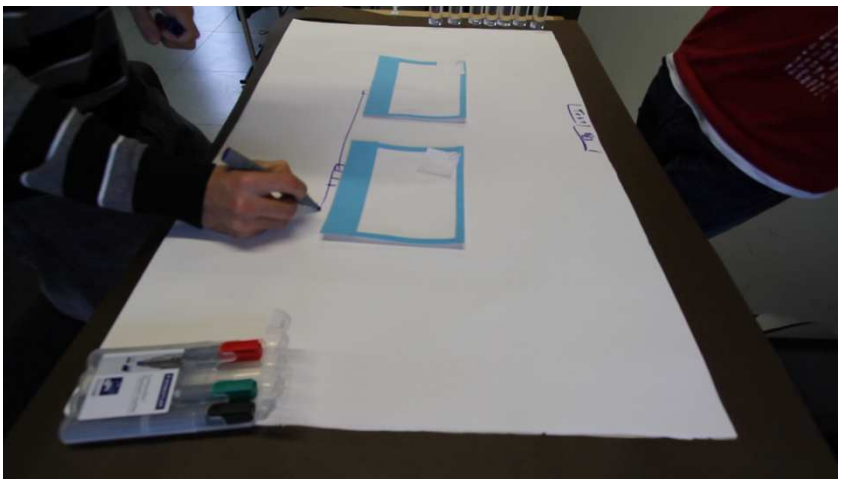


Abbildung 71: Erklärung von Koppeln mit Konnektorpunkten

Der Proband bevorzugt das Verbinden von bestimmten Fensterpunkten(Koppel- oder Ankerpunkte). Entkoppelt wird durch das Durchtrennen der Verbindungslinie. Es könnte aber auch einen Entkoppelknopf an der Linie oder am Fenster geben.

Für die Kopplungsdarstellung wird eine Farbkodierung vorgeschlagen. Die Linien sollen nur bei aktiven Fenstern angezeigt werden.

Die private Ablage wird von Proband als gut empfunden.

Auch das gebundene Tangible wird als gute Idee angesehen, musste aber erklärt werden.

Zum Filtern schlägt der Proband eine Graphstruktur vor. Er versteht die vorgeschlagene Filteridee und findet sie gut.

Ebenso wird die Notwendigkeit des Verschiebe- und Bearbeitungsmodus verstanden. Der Proband hat aber ein Problem mit dem Togglebutton und schlägt eine Art von Radiobutton vor.

Für den Visualisierungswechsel im Fenster könnte er sich einen Button oder eine Wischgeste vorstellen. Dabei sollte aber der aktuelle Interaktionsmodus beachtet werden und man sollte eine Vorschau einbauen. Bei vielen Visualisierungen wäre ein Dropdown-Menü gut.

7.2.5 Proband 4

Der 4. Proband ist männlich und 24 Jahre alt. Er studiert Softwaretechnik und hat Erfahrung mit Smartphones. Er hat schon einen Tabletop-PC benützt.

Der Proband hat keine Idee für den Einsatz von Tangibles.

Für das Verschieben und Skalieren von Fenstern schlägt er die Standardgesten vor. Für die Rotieren-Operation einen Kreis um das Element, der überall angefasst werden kann.

Durch das Schieben des Fensters aus dem Rand in Richtung der Powerwall, wird auf die Powerwall verschoben. Das Interaktionsproblem wird über ein Klonen der Powerwall gelöst. Wenn auch ein Bereich auf dem Tisch ohne Interaktion mit der Powerwall existieren soll, könnte man durch „Bild in Bild“ zwischen beiden wechseln. Die vorgestellte Powerwallansicht wird positiv aufgenommen. Fenster werden einfach auf die Ansicht geschoben oder mit einer speziellen Geste, sofern eine Überlappung mit der Powerwallansicht möglich sein soll. Der Proband würde auf die Powerwall schauen, wenn er etwas zeigen will und auf den Tisch, wenn er für sich selbst arbeitet.

Duplizieren will er über eine längere Berührung mit dem ein Menü aufgerufen wird. Die Geste, die er vorschlägt kollidiert mit der Skalierungsgeste. Darauf hingewiesen, entschließt er sich zum Auseinanderziehen mit vier Fingern, findet es aber nicht sehr intuitiv. Das Drag&Drop-Interaktionselement wird erst nach Erklärung verstanden und für sinnvoll erachtet. Als Alternative wäre ein verständlicher Ankerpunkt oder ein Seitenmenü am Fenster möglich. Bei der Benutzung von Tangibles könnte er sich eine Variante vorstellen, bei der ein Menü aus dem Fenster kommt oder die Stempel-Aktion.

Gelöscht wird mit einem Button oder dem üblichen X. Möglich ist auch das Ziehen in einen Mülleimer oder über den Rand.

Für das Koppeln schlägt der Proband vor, beide Fenster anzutippen. Nach dem Hinweis auf den Konflikt durch mehrere User hat er nach mehreren unvollständigen Ansätzen keine Idee mehr. Die Benutzung eines Koppelbutton und nachfolgendes Ziehen findet er gut. Stempeln mit einem Koppeltangible findet er sinnig, aber er zieht den Einsatz eines Koppeltangiblepaars vor. Damit werden die Fenster kurz angetippt.

Entkoppelt wird über ein Menü oder über ein Tangiblepaar entsprechend zum Koppeln.

Die private Ablage wird verstanden und für das Ablegen wird Drag&Drop empfohlen. Für die globale Ablage wird ein Rand-Container vorgeschlagen.

Der Proband schlägt selbst vor, ein Fenster durch ein Tangible aufzunehmen. Er würde das Fenster aber nicht fest binden. Nach der Erklärung wird das Binden als nicht schlecht empfunden.

Die vom Proband vorgeschlagenen Filterlösungen haben nur eingeschränkte Möglichkeiten, da Schnitt und Vereinigung vom Filtertyp abhängig sind. Die angebotene Filterlösung wird verstanden und als gut empfunden (siehe Abbildung 72).

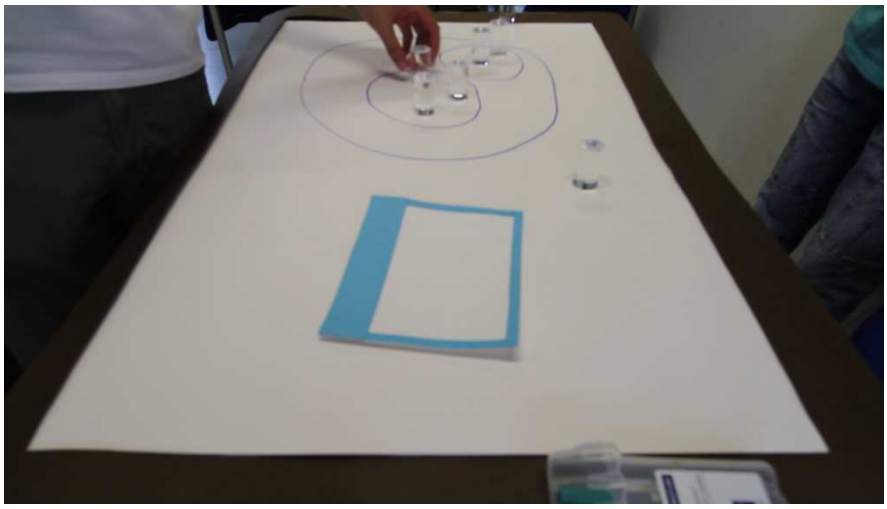


Abbildung 72: Proband beim Ausprobieren des vorgeschlagenen Filterkonzepts

Der Konflikt der Interaktionen innerhalb des Fensters und mit dem Fenster selbst wird erkannt. Der Proband schlägt einen Modus vor der nur eine einmalige Operation zulässt. Der vorgeschlagene Interaktionsmodus-Umschalter wird nicht als gut empfunden, da danach wieder zurückgeschaltet werden muss.

Bei mehreren Visualisierungen innerhalb eines Fensters schlägt der Proband vor, wie bei Webseiten, mit Hilfe eines Karussells durch die Visualisierungen zu navigieren. Alternativ kann ein Radialmenü benützt werden. Als Geste wird Wischen vorgeschlagen.

Der Proband merkt an, dass er generell Menüs bevorzuge, da man dabei direkt die Bedeutung und Möglichkeiten sähe. Das ist Vorteilhaft wenn man nicht täglich mit einer Oberfläche arbeitet.

7.2.6 Proband 5

Der 5. Proband ist weiblich und 23 Jahre alt. Sie studiert Forschung und Entwicklung in der Erziehungswissenschaft im Masterstudiengang. Sie hat Erfahrung im Umgang mit Smartphones und einem iPad.

Die Probandin hat keine spontanen Ideen für Tangibles. Sie könnte sich vorstellen damit zu Malen. Der Hinweis, dass jedes Tangible individuell erkannt wird, führt zu ihrer Idee, mit Tangibles Elemente zu gruppieren und zu kennzeichnen. Die Standardinteraktionen führt sie etwas ungewöhnlich aus. Sie rotiert die Fenster mit vier Fingern und verschiebt sie mit drei Fingern. Das Skalieren erfolgt in üblicher Weise.

Ein Fenster würde sie in Richtung der Powerwall „schubsen“, um es auf der Powerwall anzuzeigen. Sie erkennt das Interaktionsproblem mit der Powerwall und möchte, dass das Fenster nicht vom Tisch verschwindet. Ob ein Fenster nur auf dem Tisch existiert oder auch auf der Powerwall, soll farblich unterschieden werden. Für alleiniges Arbeiten würde der Tisch genutzt, um jemand etwas zu zeigen würde die Probandin auf die Powerwall schauen. Eine Powerwallansicht auf dem Tisch findet sie sehr sinnvoll. Sie würde jetzt die Fenster auf die Powerwallansicht schieben. Die Idee von Proband 2, einen möglichen Skalierungsunterschied beim Überwinden des Berührungspunkts auf die Powerwallansicht stattfinden zu lassen, findet sie gut. Eine Überlappung von Fenstern auf der Powerwall findet sie eher störend und keinesfalls nötig.

Fenster duplizieren würde die Probandin mit einem Button. Das duplizierte Fenster erscheint leicht versetzt und kann dann verschoben werden. Mehrfach dupliziert wird durch wiederholtes Drücken. Die Drag&Drop-Variante wird gleich verstanden und auch als gute Lösung empfunden. Sie könnte sich auch eine Duplizierung durch Teilen mit einer Geste vorstellen. Vier Finger ziehen das Fenster auseinander. Auf den Einsatz von Tangibles hingewiesen kommt die Probandin schnell auf ein Dupliziertangible mit dem das Stempeln nachgebildet wird. Den Einsatz von einem Tangiblepaar findet sie zu aufwendig.

Zum Löschen könnte es, nach Meinung der Probandin, entsprechend auch ein Löschtangible geben. Bei einem Löschbutton wird die Gefahr eines unbeabsichtigten Löschens als zu groß empfunden, eine ständige Nachfrage als zu lästig. An das Schieben erinnert, sieht die Probandin sofort das „vom Tisch schieben“(siehe Abbildung 73) als gute Lösung.

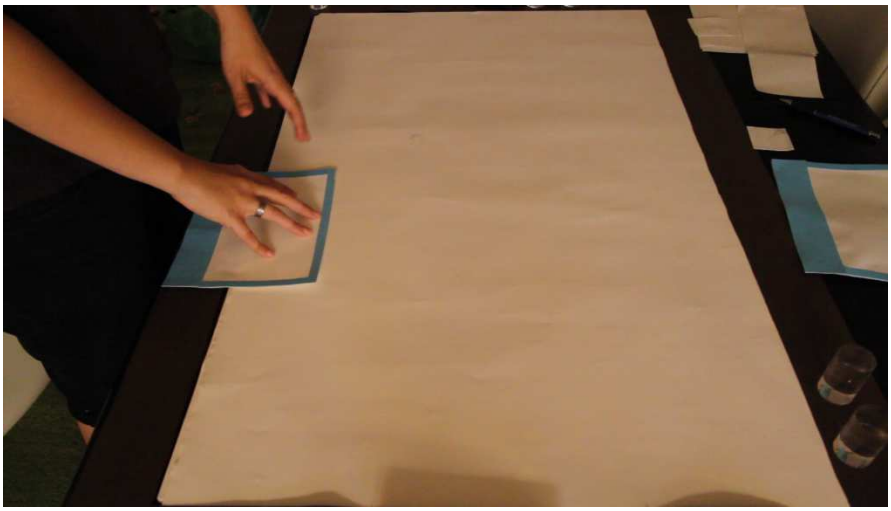


Abbildung 73: Die Probandin schiebt ein Fenster vom Tisch

Bei der ersten Idee zum Fenster koppeln werden die Fenster aneinandergeschoben. Danach könnte sich die Probandin ein Koppeltangible oder Koppeltangiblepaar (siehe Abbildung 74) vorstellen. Die Kopplung könnte durch Linien oder gleiche Zahlen in der linken oberen Ecke angezeigt werden. Den Vorschlag gleiche Farben zu nehmen findet sie nicht so gut. Die Entkopplung wird durch ein Entkoppeltangible vorgenommen. Eine Unterbrechung der Koppellinie findet sie auch gut.

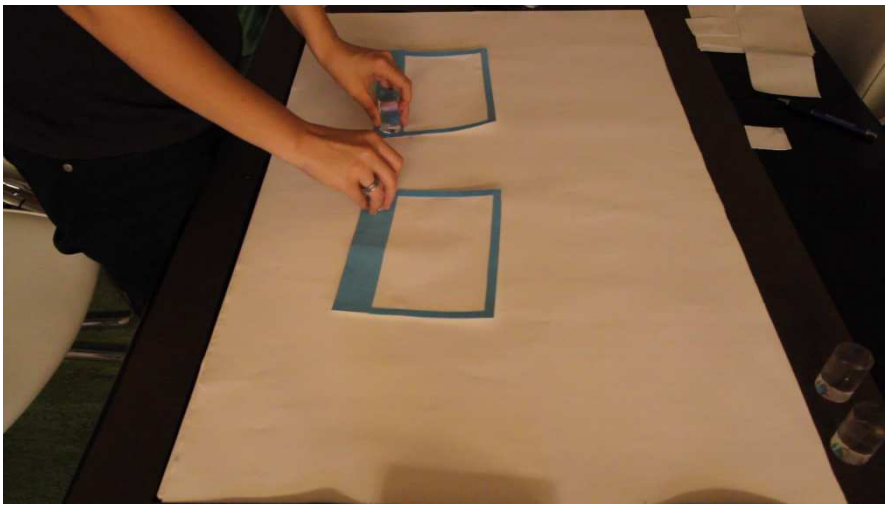


Abbildung 74: Das Koppeln mit einem Tangiblepaar wird angedeutet

Eine persönliche Ablage, die an eine Id-Karte gekoppelt ist, wird für gut befunden. Die Fenster werden hineingeschoben und erscheinen als Mini-Fenster. Die globale Ablage hätte die Probandin gerne als Weltkugel. Da es aber keine sinnvolle Position dafür gibt, wird der Vorschlag die Ablage in eine „Randschublade“ zu integrieren akzeptiert.

Das gebundene Tangible wird verstanden und als sinnvoll empfunden.

Zur Filterung hatte die Probandin keine Ideen, die vorgestellten Lösungsansätze fand sie aber gut.

Als Unterscheidung ob mit oder im Fenster gearbeitet wird, wird ein Button mit Zustandsanzeige des Interaktionsmodus vorgeschlagen.

Bei mehreren Visualisierungen im Fenster könnte sich die Probandin ein Dropdown-Menü vorstellen. Ebenso aber auch Blättern mit Hilfe einer Wischgeste.

7.2.7 Fazit

Bei der Studie mit fünf Probanden konnte keine grundlegende Präferenz für die Bedienung festgestellt werden. Es gab zwar Übereinstimmungen unter den Probanden für verschiedene Einzelinteraktionen, allerdings konnte kein klarer Trend zu einem benutzerübergreifenden Interaktionsprinzip gefunden werden. Die Vorlieben der Probanden können folgendermaßen zusammengefasst werden:

- **Proband 1**
  Der Proband bevorzugte Gesten bei den meisten Interaktionen. Er möchte keine klassischen Steuerelemente benutzen solange es nicht nötig wird. Mit Tangibles konnte er nur wenig anfangen.

- **Proband 2**
  Der Proband hätte gern schnell erreichbare Buttons für wichtige Funktionen. Denn damit könne er mit einem Klick Funktionen auslösen. Gesten würde er nutzen, wenn es sinnvoll ist, aber in einigen Fällen sind sie ihm zu aufwendig. Mit Tangibles hatte der Proband keine Ideen. Er würde sie wie die Gesten einsetzen.

- **Proband 3**
  Der Proband bevorzugt eine Umgebung mit klassischen Steuerelementen und Gesten.

Er würde Kontextmenüs an Stellen, die das ermöglichen, nutzen. Ansonsten sei er seiner Meinung nach von Smartphones beeinflusst. Daher würde er Gesten, die von Smartphones bekannt sind, nutzen. Tangibles würde er ähnlich wie die Gesten einsetzen.

- **Proband 4**
  Der Proband benötigt nach eigener Meinung eine selbsterklärende Benutzeroberfläche und möchte daher gerne Menüs. Gesten könne er sich nicht merken und er wolle sich nicht jedes Mal in ein Programm einarbeiten. Er scheint auch stark von Desktopsystemen beeinflusst zu sein. Die Tangibles findet er interessant, hat aber ebenfalls Schwierigkeiten mit ihnen.

- **Proband 5**
  Der Proband findet die Tangibles sehr interessant und hätte gerne eine Benutzerschnittstelle mit viel Tangible-Interaktion. Außerdem sei er stark von Tablets beeinflusst und wünscht sich auch viele Interaktionen mit Gesten. Der Proband wünscht sich auch verschiedene Interaktionsmöglichkeiten, um sich die angenehmste Interaktion auszusuchen.

Da die verschiedenen Vorlieben der Benutzer stark variieren, ist es schwierig ein klares Bild für das Interaktionsprinzip zu bekommen. Durch Abdeckung mehrerer Interaktionskonzepten also z.B. Gesten, Tangibles und klassischer Bedienung mit Buttons können aber die Vorlieben vieler Benutzer getroffen werden.

Im Verlauf der Studie konnten auch einige Beobachtungen von Wobbrock et al [48] bestätigt oder ähnlich beobachtet werden:

- *Die Benutzer werden stark von existierenden Systemen beeinflusst.*
  Bei einigen Probanden der Studie konnte beobachtet werden, dass sie von Desktopsystemen beeinflusst sind. So wurden z.B. Positionen von Steuerelementen vorgeschlagen, die ähnlich zu denen aus Desktop-Betriebssystemen sind. Als Beispiel kann der Schließen-Knopf eines Fensters an der rechten oberen Ecke genannt werden.

- *Bei Gesten achten die Nutzer nicht auf die Anzahl der Finger, die sie nutzen.*
  Bei den meisten Gesten der Probanden ist aufgefallen, dass sie anfangs nicht unbedingt darauf achten wie viele Finger sie benutzen. Allerdings waren sich fast alle Probanden im Klaren, wenn zwei Gesten in Konflikt kommen. Daraufhin wurde meistens eine Unterscheidung mit einer verschiedenen Anzahl der Finger vorgeschlagen.

Im Folgenden werden die Ergebnisse der Vorstudie, zu den verschiedenen Aspekten der Benutzeroberfläche der Vorstudie, zusammengefasst.

Powerwall-Steuerung/-Ansicht
Auf die Frage wie eine Visualisierung auf die Powerwall übertragen werden kann, haben alle Probanden bis auf Proband 1 vorgeschlagen sie in Richtung der Powerwall vom Tisch zu schieben. Das führte meist zu dem Problem, wie die Visualisierungen auf der Powerwall zu steuern sind. Auf dieses Problem hatten nur Proband 1, 2 und 4 eine vollständige Lösung. Ein Klonen der Powerwall auf dem Gerät. Die Ideen dazu waren eine fester Bereich der Oberfläche für das Klonen und eine Bild in Bild-Funktion, wobei Proband 1 nebenbei auch eine frei verschiebbare Powerwallansicht vorgeschlagen hat. (siehe Tabelle 2)

Tabelle 2: Vorschläge bezüglich der Powerwallsteuerung (X: der Proband erwähnt die Funktionalität)

Funktionalität	Proband 1	Proband 2	Proband 3	Proband 4	Proband 5
Schieben in Richtung der Powerwall	_	X	X	X	X
Duplizierung der Powerwall	X	X	_	X	_
Verschiebbare Powerwallansicht	X	_	_	_	_

Die Präsentation der Lösung, mit dem frei verschiebbaren Powerwallansichtsfenster auf der Benutzeroberfläche, wurde von allen Probanden positiv aufgenommen und als beste Lösung akzeptiert. Alle Probanden würden die bei der Steuerung der Powerwall eher auf die Powerwallansicht schauen als auf die Powerwall selbst. Ausnahme dabei ist, wenn der Benutzer etwas an der Powerwall vorstellt.

Interaktionen mit den Visualisierungscontainern (für die Probanden das Fenster)
Bei den Interaktionen für die Manipulationen haben fast alle Probanden die Standardgesten (siehe 7.1.3) genannt. Nur Proband 5 hat für das Verschieben drei Finger angegeben und für das Skalieren und Rotieren vier Finger. Proband 4 hat außerdem bei der Rotation einen Kreis um den Visualisierungscontainer vorgeschlagen.

Wenn ein Visualisierungscontainer auf die Powerwall verschoben wird, dann soll direkt der Visualisierungscontainer verwendet werden. Das wird von allen Probanden bestätigt.

Das Drag&Drop-Interaktionselement wurde von allen Probanden nicht verstanden und die meisten Probanden waren nach längerer Erklärung nicht überzeugt. Drag&Drop selbst wurde dagegen von allen Probanden als nützliches Prinzip gesehen. Als Ersatz des Drag&Drop-Interaktionselements hat sich der Vorschlag von Proband 2 durchgesetzt. Dieser hatte einen Knopf vorgeschlagen, der eine Drag&Drop-Operation auslöst. Für Duplizieren und Koppeln soll dabei ein Knopf vorhanden sein.

Der Interaktionsmodus für den Visualisierungscontainer, der zwischen Interaktion mit der Visualisierung und Interaktion mit dem Container umschaltet, wurde größtenteils gut aufgenommen. Nur Proband 4 würde ihn nicht verwenden, da er nicht gerne umschaltet.

Gesteninteraktion
Es werden kurz die Ideen für die Gesten beschrieben und zusammengefasst (siehe Tabelle 3).

Tabelle 3: Vorschläge für Gesten
(-- : keine Aussage oder wurde nicht gefragt)

Funktion	Proband 1	Proband 2	Proband 3	Proband 4	Proband 5
Duplizieren	Auseinander-ziehen mit jeweils zwei Fingern	Auseinander-ziehen mit jeweils drei Fingern	Auseinander-ziehen mit jeweils zwei Fingern oder mit einem Finger eine Teilungslinie ziehen	Auseinander-ziehen mit jeweils zwei Fingern	Auseinander-ziehen mit jeweils zwei Fingern
Löschen	Fenster über den Rand ziehen	Fenster auf einen Mülleiner ziehen	Fenster auf einen Mülleiner oder über den Rand ziehen	Fenster auf einen Mülleiner oder über den Rand ziehen	Fenster über den Rand ziehen
Koppeln	Fenster mit zwei Fingern halten, mit einem Finger eine Linie ziehen	--	Zusammenziehen von Interaktions-elementen oder Verbinden von Konnektor-punkten	--	Fenster aneinander-schieben
Entkoppeln	Linie mit einem Finger trennen	--	Linie mit einem Finger trennen	--	Linie mit einem Finger trennen
Visualisierungs-wechsel	--	--	Wischen	Wischen	Wischen

Das Duplizieren wurde einheitlich mit einer Geste gemacht, die das Fenster auseinanderzieht. Der Konflikt mit Skalieren wird über mehrere Finger gelöst. (siehe auch 7.1.5)

Beim Löschen ergab sich in Anbetracht der Mehrbenutzerfähigkeit die Geste, bei der das Fenster aus dem Rand gezogen wird. Alle Probanden schlagen aber auch das Ziehen auf einen Mülleimer vor.

Das Koppeln hat keine einheitliche Geste, jedoch deckt sich die Geste von Proband 1 mit der aus den Vorüberlegungen (siehe 7.1.5).

Für Entkoppeln wurde als Geste das Ziehen mit dem Finger durch die Linie vorgeschlagen, falls der Proband Linien nutzen würde.

Das Wechseln der Visualisierung wurde einheitlich mit einer Wischgeste durchgeführt. Proband 1 und Proband 2 wurden nicht befragt.

Tangibleinteraktion
Es werden kurz die Ideen für die Tangibleinteraktion beschrieben und zusammengefasst (siehe Tabelle 4).

Tabelle 4: Vorschläge für Tangibleinteraktion
(-- : keine Aussage)

Funktion	Proband 1	Proband 2	Proband 3	Proband 4	Proband 5
Duplizieren	Ziehen	Stempeln	Stempeln	Stempeln	Stempeln
Löschen	--	--	--	--	Stempeln
Koppeln	--	--	Stempeln	mit Paar	mit Paar
Entkoppeln	--	--	--	mit Paar	mit Paar

Allgemein waren die Probanden bei den Tangibles eher ratlos. Nur Proband 5 war bei dieser Art von Interaktion kreativ. Das könnte sein, weil Proband 5 weiblich ist. Das kann aber aus der Studie nicht bestätigt werden. Ein wahrscheinlicherer Ansatz ist, dass Proband 1 – 4 aus dem Bereich der Informatik stammen und Proband 5 nicht.

Bei den Tangibles hat sich für das Duplizieren fast einheitlich eine Stempel-Interaktion ergeben. Bei dieser berührt der Proband erst das Fenster und dann einen freien Bereich auf der Benutzeroberfläche.

Für das Koppeln hat sich eine Interaktion mit zwei gepaarten Tangibles herauskristallisiert. Dabei wird auf jedes Fenster eines der Tangible gestellt. Entkoppeln funktioniert dann analog. Proband 4 und Proband 5 haben sich auch explizit gegen eine weitere Stempel-Interaktion ausgesprochen.

Das Binden eines Tangibles an einen Visualisierungscontainer bzw. ein Fenster wurde von allen Probanden als gute Idee empfunden.

Abschließende Bemerkungen
Das Ergebnis der Studie ist leider nicht repräsentativ worauf auch die inhomogenen Ergebnisse hindeuten. Allerdings kann für eine größere Studie davon ausgegangen werden, dass auch dort die Probanden große Schwankungen in ihren Vorlieben haben. Dies lässt sich aus der Tatsache begründen, dass bereits die kleine Studie mit Probanden ähnlichen Alters und Profession große Unterschiede bei ihren Vorlieben aufzeigten.

Die Probanden selbst waren auf Nachfrage sehr begeistert von der Art der Studie. Die meisten versicherten auch, dass sie Spaß hatten mitzumachen auch wenn sie nicht potentielle Anwender des Systems sind. Allerdings hatten sie teilweise Hemmungen selbst Ideen auf das Papier aufzuzeichnen.

7.3 Lösungsansatz-Erweiterung

In diesem Kapitel werden Erkenntnisse aus der Vorstudie auf den Lösungsansatz angewandt und unintuitive Ideen verworfen. Im Weiteren werden die speziellen Interaktionen vorgestellt, die durch die Nutzer in der Vorstudie vorgeschlagen wurden und sich mit dem Lösungskonzept decken.

Powerwall-Steuerung/-Ansicht
Die frei verschiebbare Powerwallansicht wurde von den Probanden der Studie als gut empfunden. Darum wird am Lösungsansatz der Powerwallansicht nichts verändert.

Das Verschieben eines Visualisierungscontainers auf die Powerwall wird folgendermaßen geändert: Die Visualisierungscontainer können direkt auf die Powerwallansicht verschoben werden. Dabei wird der Berührungspunkt als Bezugspunkt verwendet, um zu entscheiden ob

sich ein Visualisierungscontainer bereits auf der Powerwallansicht befindet oder nicht. Sobald der Bezugspunkt über der Ansicht ist, wird der Visualisierungscontainer in den öffentlichen Raum verschoben und kann dann sofort auf der Powerwallansicht weiterverschoben werden.

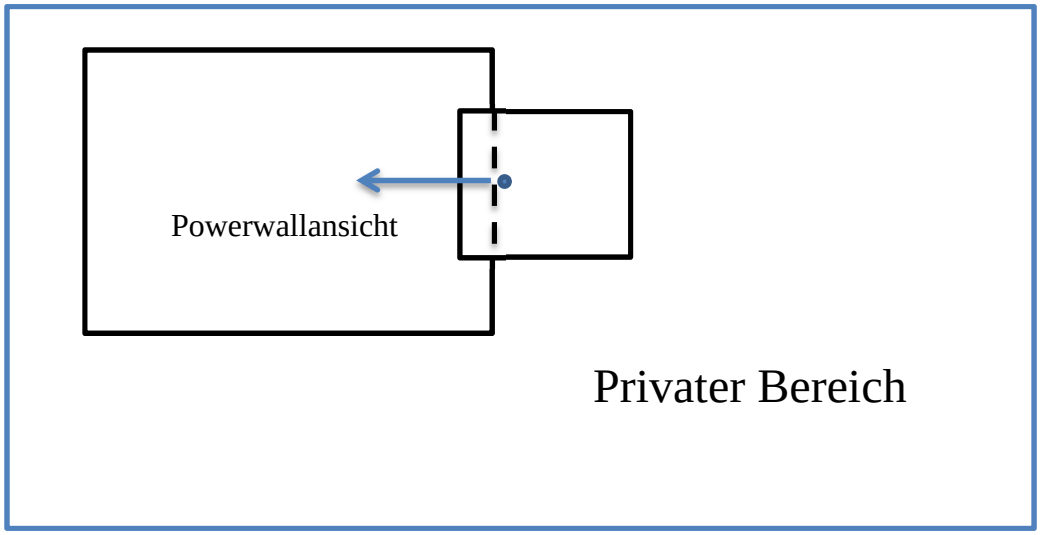


Abbildung 75: Änderung beim Verschieben auf die Powerwallansicht

Interaktion mit den Visualisierungscontainern
Die Interaktionen Verschieben, Skalieren und Rotieren wurden in der Studie bestätigt und werden daher nicht geändert.

Das Drag&Drop-Interaktionselement wurde von den Probanden nicht positiv angenommen. Daher wird dieses Interaktionselement verworfen. Drag&Drop als Interaktion war aber durchaus gefragt. Darum wurde für die jetzt fehlenden Interaktionen Duplizieren und Koppeln Knöpfe eingeführt, die beim Drücken eine Drag&Drop-Operation auslösen. Die Lösung wurde ausgewählt, weil sie von Proband 2 vorgeschlagen wurde und die folgenden Probanden diese Lösung ebenfalls akzeptierten. Der Knopf für die Duplizierung erzeugt eine Drag&Drop-Vorschau und erzeugt beim Loslassen auf dem privaten Bereich oder der Powerwallansicht ein Duplikat des Visualisierungscontainers. Der Knopf für die Kopplung erzeugt einen Drag&Drop-Cursor. Dieser kann auf einen anderen Visualisierungscontainer gezogen werden. Beim Loslassen werden die beiden Visualisierungscontainer gekoppelt.

Gesteninteraktion
Die vorgestellten Gesten werden nicht verändert, da sie entweder von den Probanden bestätigt oder sogar vorgeschlagen wurden.

Für das Entfernen von Visualisierungscontainern wird allerdings eine neue Geste eingeführt. Alle Probanden haben vorgeschlagen den Visualisierungscontainer entweder auf einen Mülleimer oder den Rand zu verschieben. Da jedoch ein Mülleimer für mehrere Benutzer nicht sinnvoll positioniert werden kann, wird der Rand des Tabletops verwendet. Jeder Benutzer hat mindestens einen Rand in seiner Nähe. Damit waren alle Probanden einverstanden. Wenn der Visualisierungscontainer auf den Rand gezogen wird, erscheint ab einen gewissen Abstand ein visuelles Feedback in Form eines Mülleimers, das das Löschen des Containers andeutet. Wird der Container vom Benutzer losgelassen, dann wird er daraufhin entfernt. Als Bezugspunkt der Operation gilt hier wiederum der Berührpunkt des Benutzers.

Tangibleinteraktion
Die Vereinigung von verschiedenen Interaktionen in ein Tangible ist bei den Probanden nicht so gut angekommen. Daher werden diese getrennt und für jede Interaktion ein separater Tangible-Typ vorgesehen.

Das Tangible für Duplizieren funktioniert wie ein Stempel. Alle Probanden waren mit einer Stempelgeste für das Duplizierentangible einverstanden. Das Tangible muss als erstes einen Visualisierungscontainer als Quelle zugeordnet bekommen. Daher wird es als erstes auf einem Container abgelegt. Ein visuelles Feedback, wie z.B. das Aufblinken des Fensters zeigt den Erfolg an. Danach kann das Tangible durch Stempeln an beliebigen freien Stellen auf dem privaten Raum und der Powerwallansicht Duplikate des ursprünglichen Visualisierungscontainers anlegen. Wenn das Tangible nicht sofort vom Tabletop genommen wird nachdem es ein Duplikat angelegt hat, kann mit dem Tangible das Duplikat verschoben werden.

Das Tangible für die Kopplung könnte ebenfalls als Stempel funktionieren. Allerdings gab es einige Probanden, die eine Unterscheidung bei den Interaktionsabläufen wünschten. Daher funktioniert die Kopplung nun mit zwei fest zugeordneten Tangibles. Sobald beide Tangibles auf jeweils einem Visualisierungscontainer stehen, werden diese gekoppelt. Falls die beiden Container schon gekoppelt waren, werden sie entkoppelt. Da die Handhabung von zwei Tangibles gleichzeitig etwas komplizierter ist, haben sie eine kurze Zeitverzögerung bevor sie zwei Visualisierungscontainer koppeln bzw. entkoppeln. Die Zeitverzögerung wird durch ein sich schließendes Kreissegment angezeigt. Damit sieht der Benutzer, dass seine Interaktion ausgeführt wird und wann sie abgeschlossen ist. Gleichzeitig ist die Bedienung für zwei Benutzer, die jeweils eines der Kopplungstangibles haben, einfacher.

Das Tangible für die Filteranzeige funktioniert gleich wie in den Vorüberlegungen. Der einzige Unterschied ist, dass es jetzt einen eigenen Tangible-Typ hat.

Die vorher genannten Tangibles können als Interaktionstangibles bezeichnet werden, da sie einer bestimmten Interaktion zugeordnet sind. Die zweite Klasse von Tangibles, die Datenrepräsentationstangibles, werden nun in einen Tangible-Typ vereinigt, der verschiedene Arten von Daten aufnehmen kann. Dieser Tangible-Typ wird für die Filter des Filterkonzepts und das Binden von Visualisierungscontainern genutzt.

Private Ablage
Bei der privaten Ablage selbst werden keine Änderungen vorgenommen. Die Art, wie Visualisierungscontainer abgelegt werden, funktioniert jetzt wie das Verschieben auf die Powerwallansicht.

Globale Ablage
Die Probanden wollten neben der Ablage, mit an Tangibles gebundenen Visualisierungs-containern, auch eine globale Ablage auf der Benutzeroberfläche. Der Konsens war eine Art Schublade am Rand des Tabletops, die durch eine Geste geöffnet werden kann. Die Visualisierungscontainer können wie das Schieben auf die Powerwall abgelegt werden. Die globale Ablage kann von allen Rändern aus geöffnet werden, enthält aber überall die gleichen Elemente. Dadurch lassen sich auch Visualisierungscontainer am Tabletop austauschen.

8 Framework der Analyseumgebung

Das folgende Kapitel beschreibt die Umsetzung des Frameworks der multimodalen Analyseumgebung. Hierbei werden die Architektur und die Entwurfsentscheidungen beschrieben, sowie beispielhaft einige Abläufe dargestellt. Das Framework wurde in Zusammenarbeit mit der Diplomarbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ [46] implementiert. Die Umsetzung basiert auf den, in Kapitel 6, vorgestellten Konzepten für multimodale Analyseumgebungen.

Das Framework bietet ein asynchrones Nachrichtensystem mit dem die einzelnen Teile des Systems, im weiteren Client genannt, synchronisiert werden können. Außerdem werden grundsätzliche Funktionen, die jeder Client im System benötigt, zur Verfügung gestellt. Um eine leichte Erweiterbarkeit und hohe Flexibilität beim Einsatz des Systems zu gewährleisten, werden Schnittstellen für die Entwicklung eigener Datenbestände, Visualisierungen und ganzer Clients bereitgestellt. Sämtliche bestehenden Implementierungen sind auf .NET-Basis. Durch die Nutzung von SOAP-Webservices zur Kommunikation ist es jedoch möglich, Clients auf Basis andere Sprachen zu implementieren.

Im Folgenden werden zuerst die eingesetzten Architekturmuster beschrieben. Darauf folgend wird die tatsächliche Architektur vorgestellt und beschrieben wie die Architekturmuster zusammenspielen. Daraufhin werden die einzelnen Komponenten und deren Zusammenhang beschrieben. Zum Schuss werden einige ausgewählte Abläufe dargestellt.

8.1 Architekturmuster

Client-Server-Architektur
Bei der Client-Server-Architektur bietet ein Server (Server-Prozess) einer beliebigen Anzahl von Clients (Client-Prozessen) Softwaredienste an. Grundsätzlich wird die Architektur genutzt, um Aufgaben innerhalb eines Netzwerks zu verteilen. Der Server muss sich dabei immer in Bereitschaft befinden, um Anfragen eines Clients beantworten zu können.

Schichtenarchitektur
Bei einer Schichtenarchitektur sind die Teile des Systems einzelne Schichten zugeordnet. Diese werden in einer festen Hierarchie angeordnet. Die Teile einer Schicht dürfen hierbei nur Aspekte tieferer Schichten verwenden, wobei bei strenger Schichtenarchitektur keine Schichten übersprungen werden dürfen. Dies ist jedoch nicht zwingend vorgegeben, allerdings führt ein Überspringen von Schichten insgesamt zu einer höheren Koppelung.

8.2 Architektur

Dieser Abschnitt beschreibt die konkrete Architektur und wie die Architekturmuster eingesetzt werden. Außerdem werden die einzelnen Teile der Architektur kurz erläutert.

Das Framework basiert auf einer, in einer Client-Server-Architektur eingebetteten, Schichten-Architektur (siehe Abbildung 76). Die wichtigsten Teile der Architektur sind die Clients, der Server und die Kommunikation zwischen diesen.

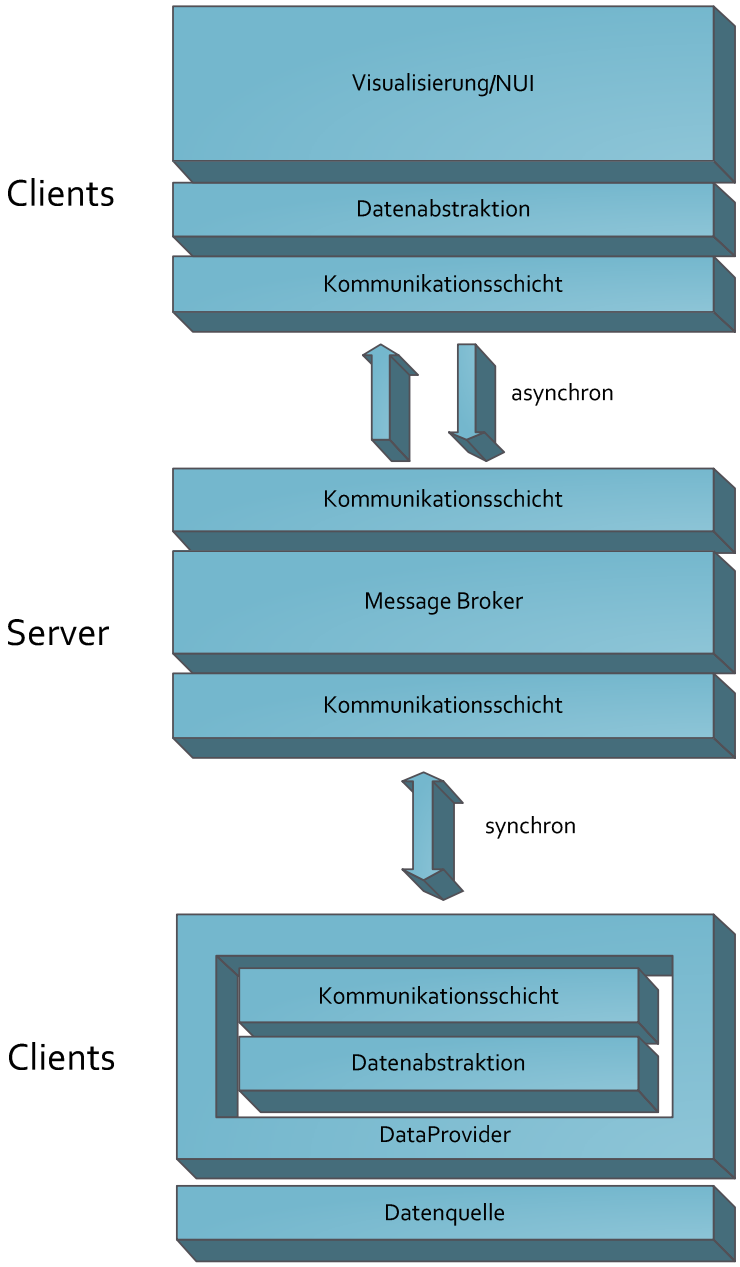


Abbildung 76: Architektur des Systems

Server
Das Framework bietet eine grundsätzliche Serverimplementierung, die einen für die Anwendung spezialisierten „message broker“ enthält. Der message broker nimmt Nachrichten der Clients an und sendet sie an ihren Adressaten oder verteilt sie im System. Der message broker ist über Kommunikationsschichten mit den Clients verbunden. Die Kommunikationsschichten werden vom Framework bereitgestellt.

Clients
Zwischen der eigentlichen Anwendungslogik der Clients und den Kommunikationsschichten findet eine Abstraktion der Daten statt, die garantieren soll, dass beliebige Arten von Daten versendet werden können. Es gibt zwei Arten von Clients: der „Visualisierungsclient“ (siehe Abbildung 76 oben), der die eigentliche Benutzerschnittstelle darstellt und der „Dataprovider-Client“(siehe Abbildung 76 unten), der die Möglichkeit bieten soll jegliche Datenquelle als Grundlage zu verwenden. Beim Visualisierungsclient werden grundsätzliche Funktionen, wie die Kommunikation zum Server und ein Visualisierungspluginmanager, bereits von dem

Framework implementiert. Der Dataprovider ist als Schnittstelle definiert, wobei die Kommunikation mit dem Server bereits implementiert ist.

Kommunikation
Die Kommunikation der Visualisierungsclients zum Server funktioniert asynchron, um dem Benutzer eine möglichst nahtlose Bedienung zu ermöglichen. Die Übertragung findet über Nachrichten statt, welche von einer im Framework enthaltene Bibliothek beschrieben werden.

Der DataProvider kommuniziert synchron über einen Webservice mit dem Server. Da der DataProvider eine ausgelagerte Komponente des Servers darstellt und daher eine engere Beziehung zwischen den beiden Komponenten besteht, ist eine synchrone Kommunikation angemessener.

8.3 Komponentenstruktur

Im Weiteren folgt eine Beschreibung der wichtigsten Komponenten im System . Die folgenden Komponenten sind: die Bibliotheken der Abstraction-Komponente, die Server-Komponente mit dem message broker, die DataProvider-Komponente zur Datenanbindung und die Komponente für die Visualisierungsclients (siehe Abbildung 77).

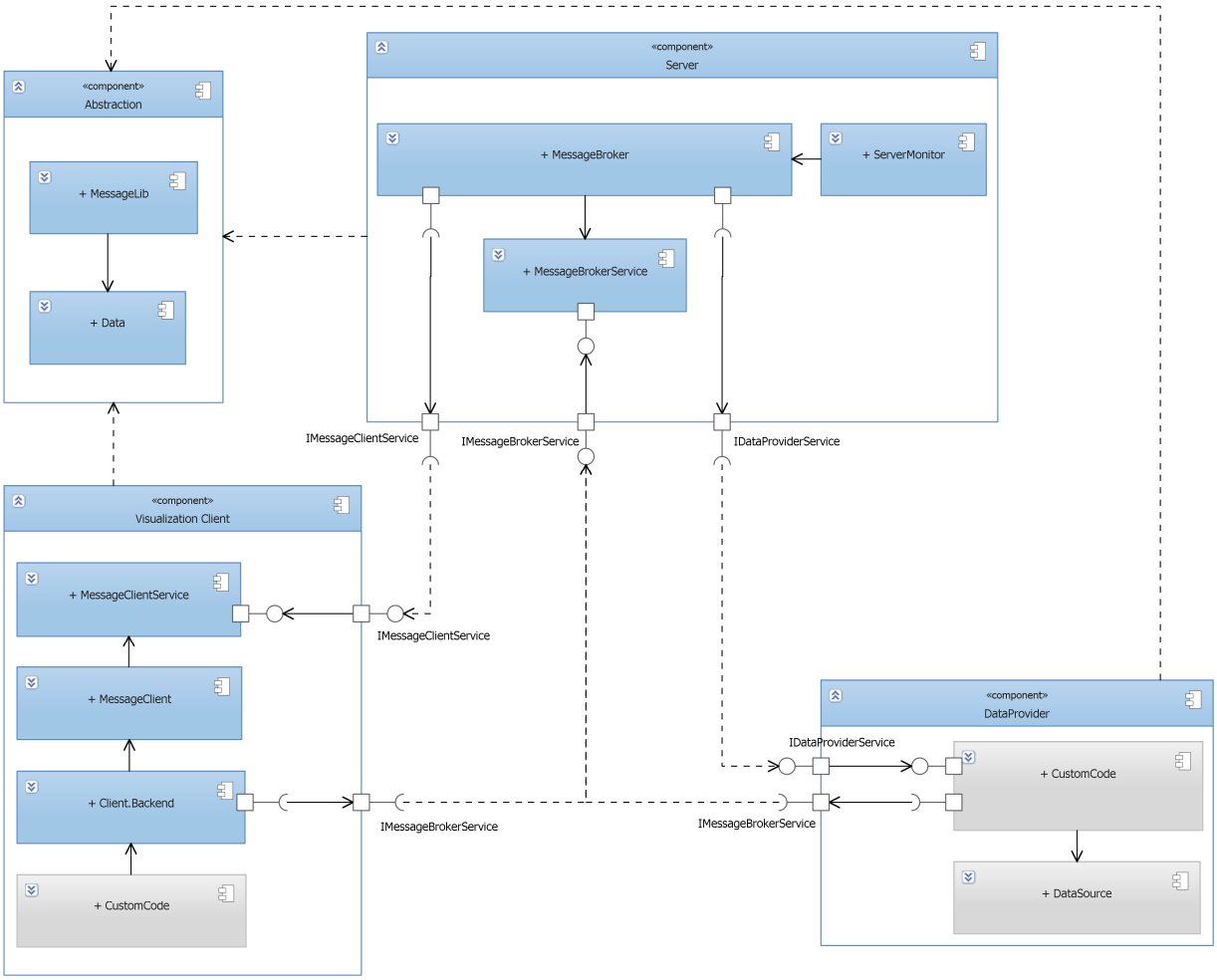


Abbildung 77: Komponentenstruktur des Systems und deren Zusammenhänge

8.3.1 Abstraction

Die Abstraction-Komponente enthält Bibliotheken, die die Abstraktion der Datenhaltung und des Nachrichtensystems ermöglichen (siehe Abbildung 78). Die Funktionalität entspricht der Abstraktionsschicht. Die Abstraktion findet sowohl bei den Nachrichten als auch bei den Daten statt, um sowohl das Nachrichtensystem als auch die Datenhaltung erweitern zu können.

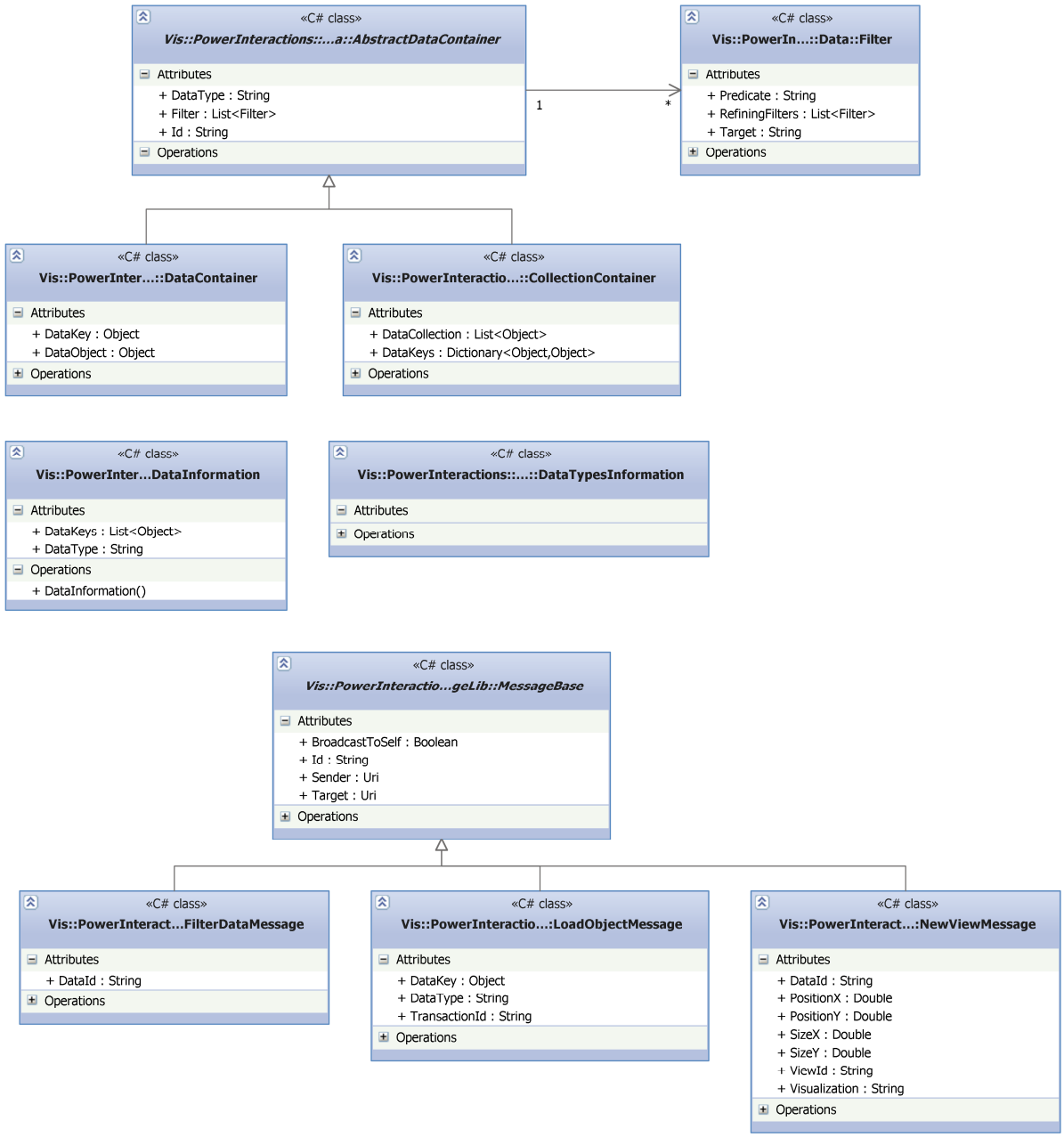


Abbildung 78: Klassenstruktur der Abstraction-Komponente

Die Daten werden über Datencontainer abstrahiert und mit Metainformationen, wie die Id, für die Visualisierungsclients versehen. Diese Zusatzinformationen befinden sich in `AbstractDataContainer`. Hier werden der Datentyp der Daten, die eindeutige Id des Datencontainers und die Filter, die auf den Daten wirken sollen, festgelegt. Die eigentlichen Daten sind in den Unterklassen zu finden, die entweder einzelne Daten aufnehmen oder eine Datenmenge. Zusätzlich werden die Datenschlüssel vermerkt.

Jeder Datencontainer hat eine Menge von Filtern (`Filter`), die als Baumstruktur aufgebaut werden. Jeder Filter repräsentiert eine Teilmenge der Daten, welche den Filterkriterien entsprechen. Die Baumstruktur sagt aus wie die Datenmengen verbunden werden. Wenn die Filter Geschwisterknoten sind werden die Datenmengen vereinigt. Ein Filter als Kindknoten verfeinert die Datenmenge nochmal. Ein Filter wird durch ein Filterprädikat und ein Ziel beschrieben. Das Filterprädikat (`Predicate`) ist ein Ausdruck, der beschreibt wie gefiltert werden soll. Das Ziel (`Target`) ist eine Eigenschaft der Datenmenge, nach der gefiltert werden soll.

Die Nachrichten des Systems werden über die abstrakte Klasse `MessageBase` abstrahiert. Diese enthält alle Informationen, um die Nachricht an den entsprechenden Client weiterzuleiten. So enthält die Nachricht zum Beispiel das Ziel und den Sender der Nachricht. Adressat und Absender werden als Urls von Webservices angegeben. Alle weiteren Eigenschaften der Nachrichten sind frei wählbar.

Sowohl Daten- als auch Nachrichtenabstraktion nutzen Reflection, um die eigentlichen Datentypen zu erkennen, was nötig ist, damit der XML-Serialisierer die Nachrichten und ihre Inhalte versenden kann. Dazu wird die Klasse `KnownTypeHelper` verwendet. Die Klasse liest die in der Anwendungskonfiguration angegebenen Assemblies ein und gibt mit Hilfe des `IKnownTypeHelper`-Interface die gesuchten Datentypen zurück.

8.3.2 Server

Der Server ist die zentrale Komponente des Gesamtsystems und verwaltet die Nachrichtenkommunikation zwischen den Visualisierungsclients. Außerdem verwaltet er die DataProvider und leitet Anfragen der Visualisierungsclients an die Datenhaltung weiter.

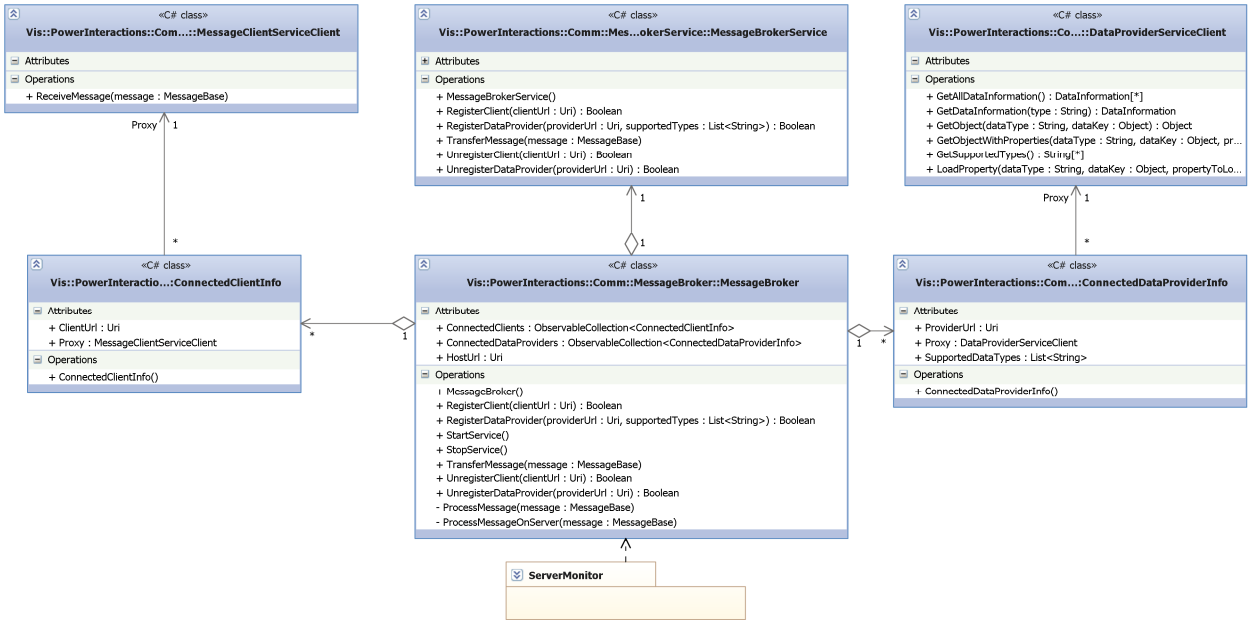


Abbildung 79: Klassenstruktur des Servers

Message Broker

Die zentrale Komponente des Servers ist der Message Broker. Dieser wird durch die `MessageBroker`-Klasse implementiert. Die Klasse enthält den Service des Servers, die Clients für die Rückkanäle zu den Visualisierungsclients und die Clients für den DataProvider-Service.

Die Funktionalität des `MessageBroker` wird durch folgende öffentliche Methoden bereitgestellt:

- `RegisterClient(clientUrl : Uri)`
  Diese Methode registriert einen Visualisierungsclient. Danach werden empfangene Broadcast-Nachrichten auch an den neu registrierten Client gesendet und es können Nachrichten direkt an den registrierten Client gesendet werden. Der Client ist über die `clientUrl` erreichbar. Diese muss auf einen gestarteten MessageClientService verweisen. Die Methode wird vom MessageBrokerService aufgerufen, wenn `RegisterClient(…)` von außerhalb ausgelöst wird.
- `RegisterDataProvider(…)`
  Es wird ein neuer DataProvider registriert. Dieser übergibt seine `providerUrl`, über die ein DataProviderService erreichbar sein sollte, und die Typen, die von dem DatenProvider abgerufen werden können. Nach der Registrierung stehen die Daten von dem DataProvider den Visualisierungsclients zur Verfügung.
  Ereignis des MessageBrokerService wird zu dieser Methode weitergeleitet.
- `StartService()`
  Startet die Services des Servers. Dieser muss gestartet sein, damit Clients registriert werden können.
- `StopService()`
  Stoppt die Services des Servers.
- `TransferMessage(message: MessageBase)`
  An diese Methode wird ein `TransferMessage(…)` –Ereignis des MessageBrokerService weitergeleitet. Um Asynchronität zu gewährleisten, wird die ProcessMessage-Methode in einem neuen Thread gestartet.
- `UnregisterClient(clientUrl: URI)`
  Entfernt die Registrierung eines Clients mit der `clientUrl`. Das Ereignis wird von MessageBrokerService an diese Methode weitergeleitet.
- `UnregisterDataProvider(providerUrl: URI)`
  Entfernt die Registrierung eines DataProviders. Das Ereignis wird vom MessageBrokerService an diese Methode weitergeleitet.
- `ProcessMessage(message: MessageBase)`
  Verarbeitet eine Nachricht (`message`) von einem Visualisierungsclient.
- `ProcessMessageOnServer(message: MessageBase)`
  Verarbeitet Nachrichten (`message`), die nicht an andere Visualisierungsclients gerichtet sind sondern an den Server selbst. Dazu gehören Nachrichten, die Daten von dem DataProvider abfragen.

MessageBrokerService
Der `MessageBrokerService` ist der Service, der die Funktionalität des Servers nach außen bereitstellt. Alle Aufrufe werden an den `MessageBroker` weitergeleitet. Die Methoden des `MessageBrokerService` finden sich daher auch im `MessageBroker`.

MessageClientServiceClient
Diese Clients bilden die Rückleitungen zu den Visualisierungsclients. Dafür wird auf den Visualierungsclients ein Webservice bereitgestellt, mit dem sich der Server verbindet sobald ein Client sich registriert. Die einzige Methode ist `ReceiveMessage(message: MessageBase)`. Die Methode sendet eine Nachricht an den Service.

DataProviderServiceClient
Diese Clients werden mit den DataProvidern verbunden. Wird in der DataProvider näher erklärt.

8.3.3 DataProvider

Die DataProvider implementieren die Funktionalität der Datenhaltung. Diese können von den Anwendern des Systems anhand ihrer Datenquellen implementiert werden. Jeder DataProvider kann eine Anzahl von Datentypen unterstützen. Die Datentypen können später von den Visualisierungsclients abgefragt werden. Die unterstützen Datentypen sollten mit Datentypen mit der gleichen Signatur bei den Visualisierungen kompatibel sein. Um Flexibilität bei der Implementierung zu ermöglichen, werden Datentypen als Strings definiert. Die DatenProvider sollten das Nachladen von Daten mit variabler Tiefe unterstützen. Dadurch lässt sich innerhalb der Clients, abhängig von der Datenmenge und Datenstruktur, ein ideales Gleichgewicht zwischen Anzahl der Nachladezyklen und der Datenmenge beim Nachladen festlegen. Einzelne Datenobjekte können hierbei, anhand ihres Datentyps und Indentifikationsschlüssels, eindeutig identifiziert werden. Der Identifikationsschlüssel kann von jeglichem Datentyp sein, der Aufrufer muss jedoch darauf achten, dass ein kompatibler Schlüssel übergeben wird.

Der DataProviderService stellt die Funktionalität des DataProvider nach außen zur Verfügung, wobei in der `IDataProviderService`- Schnittstelle die Methoden des DataProvider definiert sind.

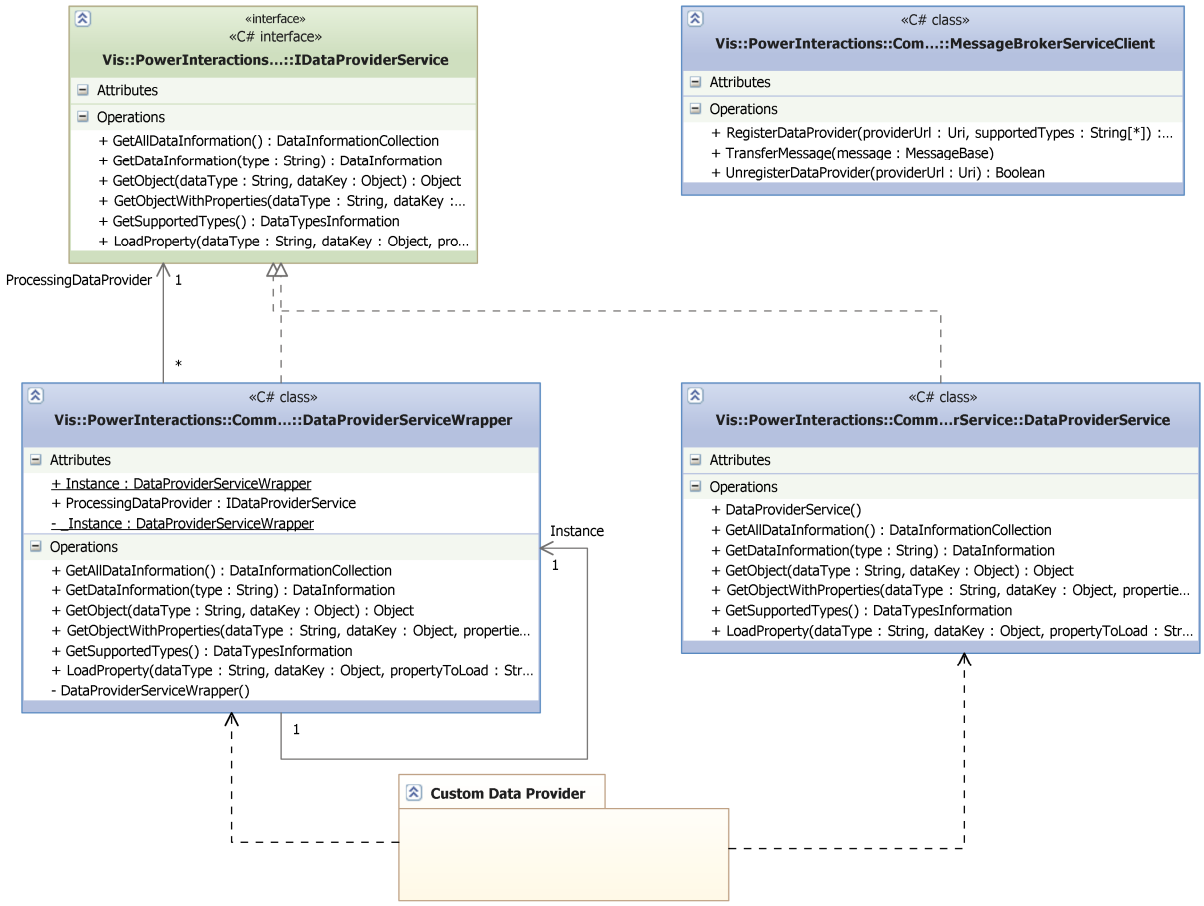


Abbildung 80: Klassenstruktur der DataProvider-Komponente

Die Methoden der Schnittstelle sind die folgenden:

- `GetAllInformation()`
  Gibt alle Datentypen mit den zugehörigen Datenidentifikationsschlüsseln zurück. Diese Methode kann aufgerufen werden, um einen Überblick über alle Datentypen des DataProvider zu bekommen.
- `GetDataInformation(type: String)`
  Gibt alle Datenindentifikationsschlüssel eines Datentyps zurück. Die Methode wird verwendet, um Zugriff auf alle Daten eines Datentyps zu erhalten.
- `GetObject(dataType: String, dataKey: Object)`
  Diese Methode gibt ein Datenobjekt anhand seines Typs(`dataType`) und Indentifikationsschlüssels (`dataKey`) zurück. Das zurückgegebene Objekt enthält keine weiteren Objekte. Alle Referenzen zu weiteren Objekten fehlen. Diese Objekte müssen identifiziert werden oder müssen über die `LoadProperty`-Methode nachgeladen werden.
- `GetObjectWithProperties(dataType: String, dataKey: Object, propertiesToInclude: String[])`
  Alternativ zu `GetObject(…)` können Datenobjekte auch mit referenzierten Objekten geladen werden. Die Strings in `propertiesToInclude` legen fest, welche referenzierten Objekte geladen werden. Dabei stellt jeder String ein Pfad im Datenmodell dar.
- `GetSupportedTypes()`
  Gibt die unterstützen Typen des DataProviders zurück.
- `LoadProperty(dataType: String, dataKey: Object, propertyToLoad: String)`
  Lädt eine Referenz, in C# eine Eigenschaft, eines Datenobjekts, welches wieder durch Typ und Schlüssel identifiziert wird. Der Name der Referenz wird durch `propertyToLoad` angegeben.

Wenn ein DataProvider für eine Datenquelle entwickelt werden soll, kann entweder ein Webservice angeboten werden, der die Schnittstelle implementiert oder eine Implementierung der Schnittstelle wird dem DataProviderServiceWrapper übergeben, der einen Webservice erstellt. Der DataProviderServiceWrapper ist eine Singleton-Klasse und implementiert den DataProvider-Service als Fassade. Die eigentliche Implementierung kann dem Wrapper über die Eigenschaft ProcessingDataProvider zugeordnet werden, welche wiederum die Schnittstelle implementieren muss.

8.3.4 VisualizationClient

Die Komponente VisualizationClient ist für die Visualisierungsclients, die den Benutzern die Benutzeroberfläche für die Analyseumgebung für bestimmte Geräte zur Verfügung stellen. Das Framework bietet ein grundsätzliches Client-Backend, welches die Kommunikation mit Ereignissen kapselt. Diese Ereignisse können von den Visualsierungsclients abonniert werden und sie können so den Empfang der Nachrichten direkt in ihre Anwendungslogik einbauen.

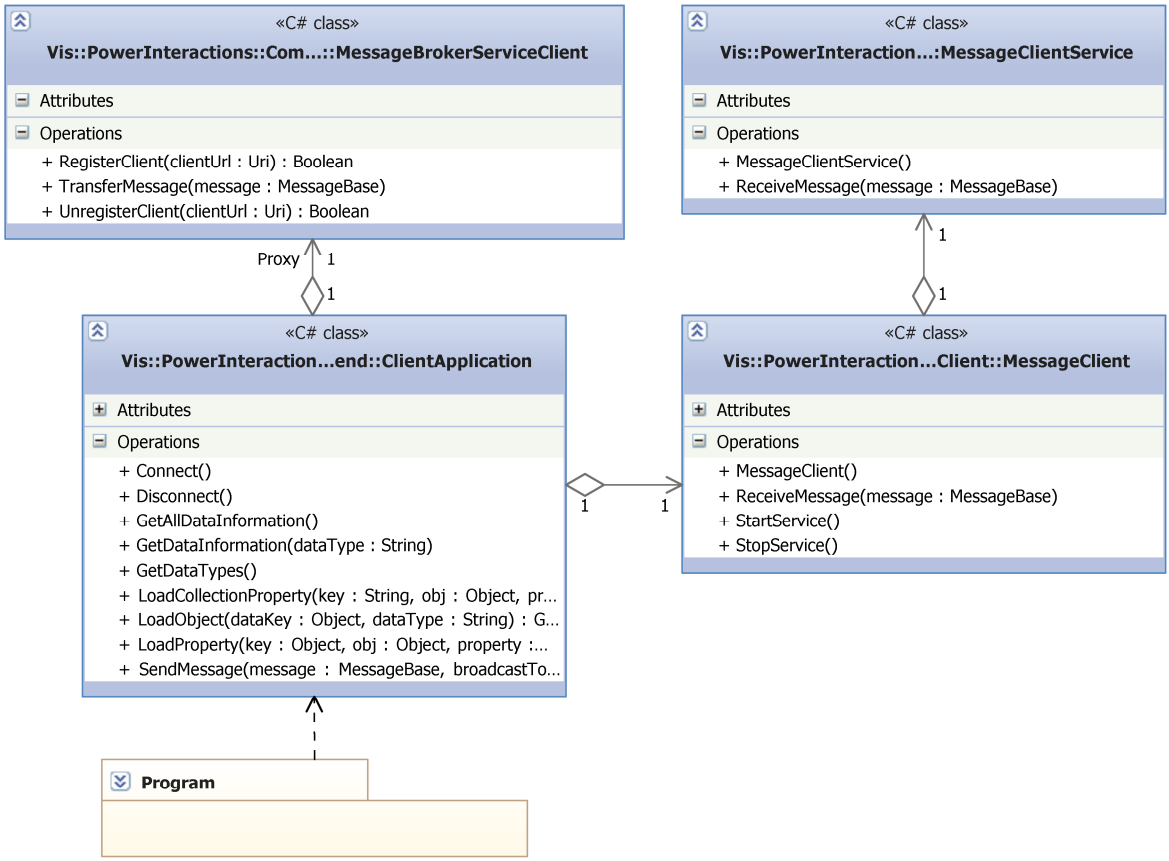


Abbildung 81: Klassenstruktur des VisualisierungsClient-Backends

Die Klasse `ClientApplication` dient als Grundlage für eine WPF-Anwendung. Sie kapselt die Webservice-Komponenten und bietet die oben genannten Zusatzfunktionen. Die Klasse `MessageClient` erzeugt einen Service, der vom Server aufgerufen wird und stellt damit die Rückleitung vom Server zur Verfügung. Der Service selbst wird durch die `MessageClientService`-Klasse dargestellt. Die einzige Methode des Webservice ist `ReceiveMessage(…)`. Diese empfängt Nachrichten vom Server und leitet sie an die `ClientApplication`-Klasse weiter. Die Klasse `MessageBrokerServiceClient` ist der Client des MessageBrokerService. Mit ihm können Nachrichten an den Server versendet werden. (siehe Abbildung 81)

Um einen Visualisierungsclient zu entwickeln, kann die ClientApplication-Klasse als Basisklasse verwendet werden. Ihre Funktionalität wird unter anderem mit folgenden Methoden bereitgestellt:

- `Connect()`
  Startet den MessageClient-Service damit der Server eine Verbindung zum Client aufbauen kann. Danach verbindet das Backend die Anwendung mit dem MessageBroker-Service und registriert den Client beim Server mit der RegisterClient(clientUrl) Methode. Die clientUrl ist die Adresse des MessageClient-Service um eine Verbindung zum Client zu ermöglichen.
- `Disconnect()`
  Entfernt die Registrierung des Clients beim MessageBroker-Service mit UnregisterClient(clientURL). Danach wird die Verbindung zum Server geschlossen und der MessageClient-Service beendet.

- `GetAllDataInformation()`
  Fragt beim Server alle Dateninformation von allen DataProvidern ab. Dazu muss der Server bei allen DataProvidern die GetAllDataInformation()-Methoden aufrufen.
- `GetDataInformation(dataType: String)`
  Fragt die Dateninformation für einen Datentypen ab. Der Server fragt die Dateninformationen für den Datentyp bei den DataProvider mit `GetDataInfromation(…)` ab. Wenn mehrere DataProvider den gleichen Datentyp unterstützen müssen diese kompatibel sein.
- `GetDataTypes()`
  Fragt alle unterstützten Datentypen ab. Der Server ruft bei allen DataProvidern die Datentypen mit GetDataTypes() ab. Auch hier gilt, dass die gleichen Datentypen kompatibel sein müssen.
- `LoadObject(dataKey: Object, dataType: String)`
  Lädt ein Object mit den gegebenen Daten, siehe `LoadObject(…)` im DataProvider.
- `LoadProperty(key: Object, obj: Object, property: String)`
  Lädt eine Eigenschaft eines Objekts, siehe `LoadProperty(…)` im DataProvider.
- `SendMessage(message: MessageBase, broadcastToSelf: bool, target: Uri)`
  Sendet eine Nachricht über den Server an ein Ziel (`target`). Wenn das Ziel nicht angegeben ist, bekommen alle Visualisierungsclients diese Nachricht. Der Parameter `broadcastToSelf` gibt an, ob der Client selbst die Nachricht auch bekommt.

Für alle üblichen Nachrichten existieren zusätzlich noch eigene Methoden. Mit diesen kann eine Nachricht es bestimmten Typs versendet werden.

Neben der Funktionalität der `VisualizationClient`-Klasse können die Visualisierungsclients ein Pluginsystem für die Visualisierungen verwenden.

Visualisierungsplugins (Visualizations-Komponente)

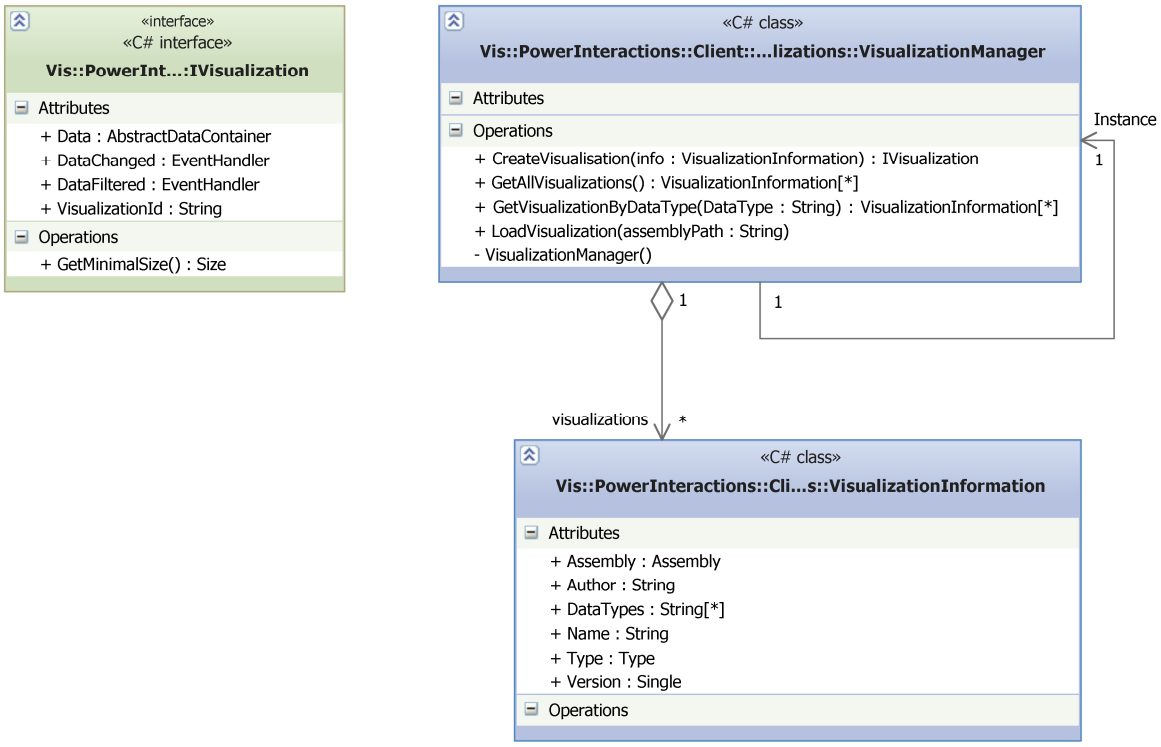


Abbildung 82: Klassenstruktur des Plugin-Systems für Visualisierungen

Zur Verwaltung und zum dynamischen Nachladen von Visualisierungen existiert ein Visualisierungsmanager. Dieser lädt, die in der Applikationskonfiguration angegebene, Assemblies für Visualisierungen. Danach werden die entsprechenden Visualisierungs-Controls ausgelesen und gespeichert. Diese müssen das `IVisualization`-Interface implementieren und vom Typ `UIControl` sein. Alle wichtigen Meta-Informationen des Controls können über die Klasse `VisualizationInformation` abgerufen werden.

Eine Visualization-Assembly wird über eine „visConfig.xml“ definiert und mit Meta-Informationen annotiert. Die XML ist folgendermaßen aufgebaut:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <Author>Nico</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 1: Zeigt den Aufbau einer Konfigurationsdatei für ein Visualisierungsplugin

Der Visualization -Assembly wird ein Autor mit dem entsprechenden Tag zugeordnet. Versionsinformationen können über die Version der Assembly festgelegt werden, um konform mit dem .NET-Framework zu bleiben. Die einzelnen Visualisierungen werden innerhalb des Visualizations-Tag definiert. Für jede Visualisierung muss ein Visualization-

Tag existieren. Der Tag erhält ein name-, version- und class- Attribut. Die drei Attribute bestimmen die Identität der Visualisierung. Das class-Attribut dient zusätzlich dazu, die Visualisierung später zu instanziieren. Innerhalb des Tags werden die Typen, die die Visualisierung verarbeiten kann, definiert. Jeder Typ wird durch einem Type-Tag beschrieben. (siehe Quellcode 1)

Die wichtigste Klasse, um die Visualisierungen nutzen zu können, ist die `VisualizationManager`-Klasse. Sie ist eine Singleton-Klasse und verwaltet die Visualisierungen. Die Klasse besitzt folgende Methoden:

- `CreateVisualization(info: VisualizationInfo)`
  Erstellt ein Visualisierungs-Control mit den gegebenen Meta-Informationen.
- `GetAllVisualizations()`
  Gibt die Meta-Informationen aller Visualisierungs-Controls zurück. Hiermit können alle geladenen Visualisierungen ausgelesen werden.
- `GetVisualizationsByDataType(dataType: string)`
  Gibt die Meta-Informationen aller Visualisierungs-Controls zurück, die den gegebenen Datentyp verarbeiten können. Die Methode ist nützlich, um alle Visualisierungen zu erhalten, die den aktuellen Datentyp unterstützen.
- `LoadVisualization(assemblyPath: string)`
  Lädt eine Assembly mit Visualisierungs-Controls manuell.

Jede der Visualisierungs-Controls muss das Interface `IVisualization` implementieren. Dadurch wird eine Grundfunktionalität und die Schnittstelle zum Visualisierungsclient gewährleistet. Das Interface hat folgende Elemente:

- `Data`
  Diese Eigenschaft enthält die aktuelle Datengrundlage der Visualisierung. Die Datengrundlage wird dabei von einem `AbstractDataContainer` repräsentiert. Die Implementierung muss dafür sorgen, dass eine Änderung der Grundlage eine Änderung der Sicht bewirkt.
- `DataChanged`
  `DataChanged` ist ein Ereignis, das von dem Visualisierungsclient abonniert werden kann. Das Ereignis wird ausgelöst wenn sich die Daten ändern. Das Ereignis ist ein „Stub“ für spätere Implementierungen, da Datenänderung noch nicht vorgesehen ist.
- `DataFiltered`
  DataFiltered ist ein Ereignis, das von dem Visualisierungsclient abonniert werden sollte. Es wird ausgelöst wenn die Daten von der Visualisierung gefiltert werden.
- `VisualizationId`
  Diese Id ist die eindeutige Identifikation der Visualisierungsinstanz. Sie sollte nur initial gesetzt werden.
- `GetMinimalSize()`
  Gibt die minimale Größe, die die Visualisierung sinnvoll anzeigen kann, an. Die Visualisierungsclients sollten die Visualisierungen nicht kleiner anzeigen oder eine alternative Ansicht zeigen.

8.4 Abläufe

Dieses Kapitel beschreibt einige wichtige Abläufe des Systems. Zunächst wird der Ablauf, mit der sich ein Visualisierungsclient beim Server registrieren kann, beschrieben. Dadurch wird der Client in die Analyseumgebung eingebunden. Danach wird gezeigt, wie der message broker Nachrichten im System verteilt. Zum Schluss wird erklärt wie die Visualisierungsclients Daten von den DataProvidern nachladen können.

8.4.1 Visualisierungsclient registrieren

Der Abschnitt zeigt den Ablauf, wenn ein Visualisierungsclient sich beim message broker registriert. Das Registrieren geschieht im Framework automatisch, wenn beim Visualisierungsclient `Connect(…)` aufgerufen wird.

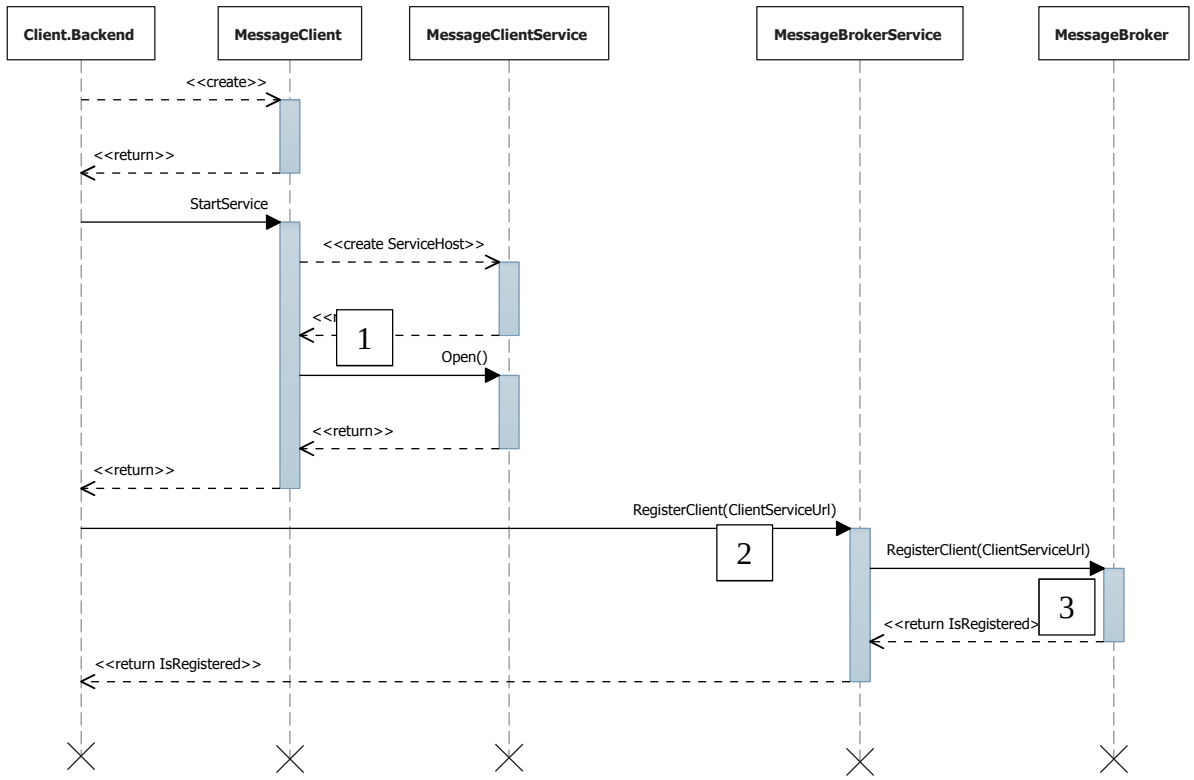


Abbildung 83: Ablauf beim Registrieren eines Visualisierungsclients

Der Visualisierungsclient erzeugt einen neuen `MessageClient`, der den Service bereitstellt, über den der Server Nachrichten zum Client versenden kann. Dieser Service wird gestartet, wodurch ein ServiceHost angelegt und der Service für Zugriffe geöffnet wird (1 in Abbildung 83). Danach wird der `MessageBrokerServiceClient` mit dem Server verbunden und `RegisterClient(…)` des Services aufgerufen (2 in Abbildung 83). Dieser leitet es zum `MessageBroker` weiter. Im `MessageBroker` wird drauf der Service des `MessageClients` registriert (3 in Abbildung 83). Die Aufrufe sind alle synchron.

8.4.2 Nachricht verteilen

Eine Übersicht über den Ablauf beim Versenden und Verteilen von Nachrichten.

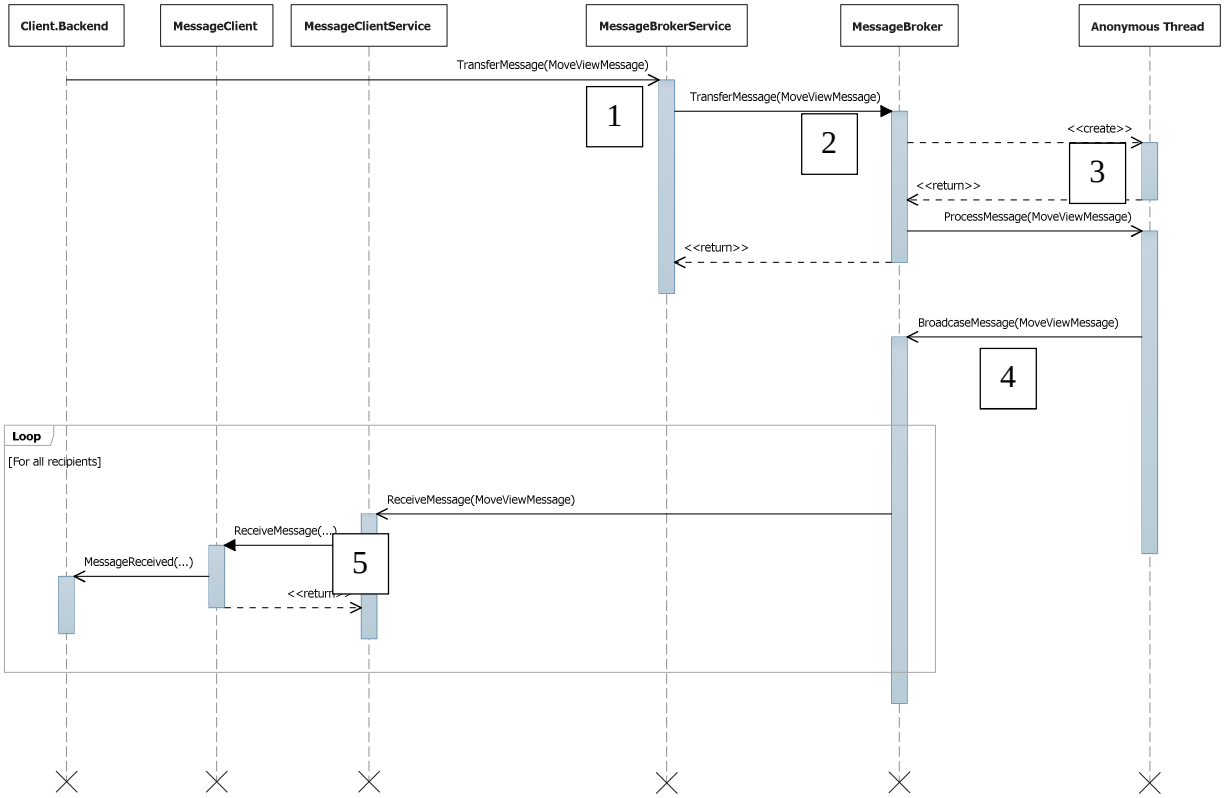


Abbildung 84: Ablauf beim Verteilen von Nachrichten im System

Wenn im Visualisierungsclient eine Nachricht versendet wird, löst das einen asynchronen Aufruf der `TransferMessage(…)`-Methode des `MessageBroker`-Service aus (1 in Abbildung 84). Dieser leitet den Aufruf an den message broker weiter, der wiederum die Zustellung der Nachricht übernimmt (2 in Abbildung 84). Im message broker werden die Sender und Empfänger Eigenschaften der Nachricht ausgelesen und die Nachricht entsprechend verarbeitet. Der message broker legt dazu einen neuen anonymen Thread an und verarbeitet die Nachricht mit diesem (3 in Abbildung 84). Der neue Thread soll die Erreichbarkeit des Servers gewährleisten, indem der Servicethread entlastet wird. Die Nachrichten die keinen Sender haben werden ignoriert, da es nicht möglich ist eine Antwort zu senden. Da das für einige Nachrichten aber nötig ist, genügt eine solche Nachricht nicht den Minimalanforderungen. Im weiteren Verlauf wird der Empfänger oder das Ziel der Nachricht ausgelesen. Dabei gibt es grundsätzlich zwei Möglichkeiten (siehe Abbildung 85). Es gibt keinen Empfänger, dann wird die Nachricht an alle Clients weitergeleitet oder es gibt ein einzelnes Ziel. Dabei muss zwischen dem Server als Empfänger und einem Client als Empfänger unterschieden werden. Wenn der Server der Empfänger ist, dann wird die `ProcessMessageOnServer(…)`-Methode aufgerufen. Der Server kann jedoch nur bestimmte Nachrichten verarbeiten, alle anderen werden ignoriert. Im Falle eines Clients als Empfänger wird gleich vorgegangen wie bei keinem Empfänger, mit dem Unterschied, dass nur der eine Client benachrichtigt wird. Dies geschieht, indem die `RecieveMessage(…)`-Methode des `MessageClient`-Service aufgerufen wird (5 in Abbildung 84). Die Antwort an die Clients findet ebenfalls asynchron statt. Der Client erhält zum Schluss die Nachricht vom `MessageClient`.

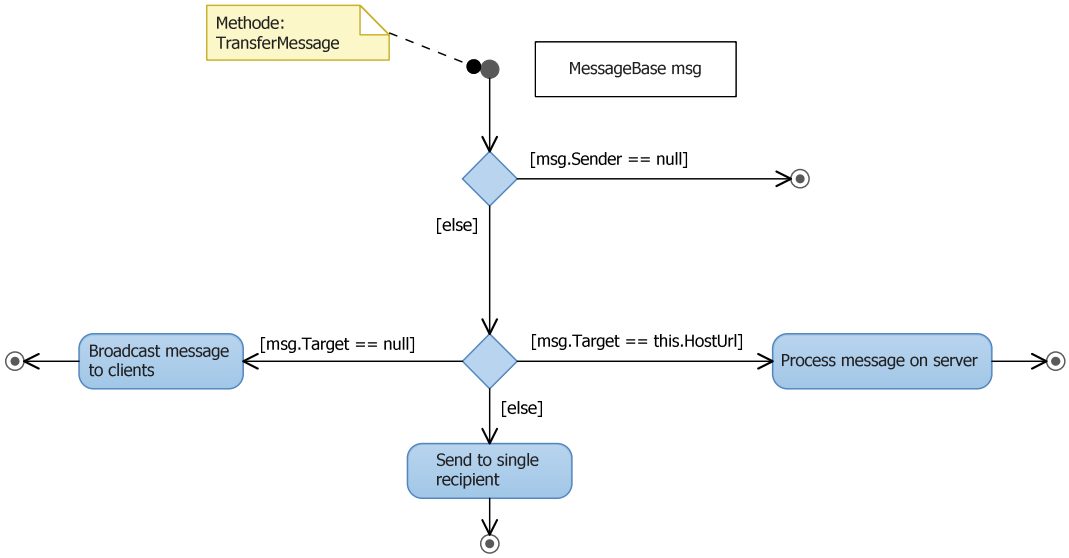


Abbildung 85: Zustände des message brokers beim Verteilen der Nachrichten

8.4.3 Daten nachladen

Angelehnt an das vorherige Kapitel wird in diesem der Ablauf beim Versenden einer Nachricht an den Server gezeigt. Als Beispiel dient das Nachladen eines Datenobjekts.

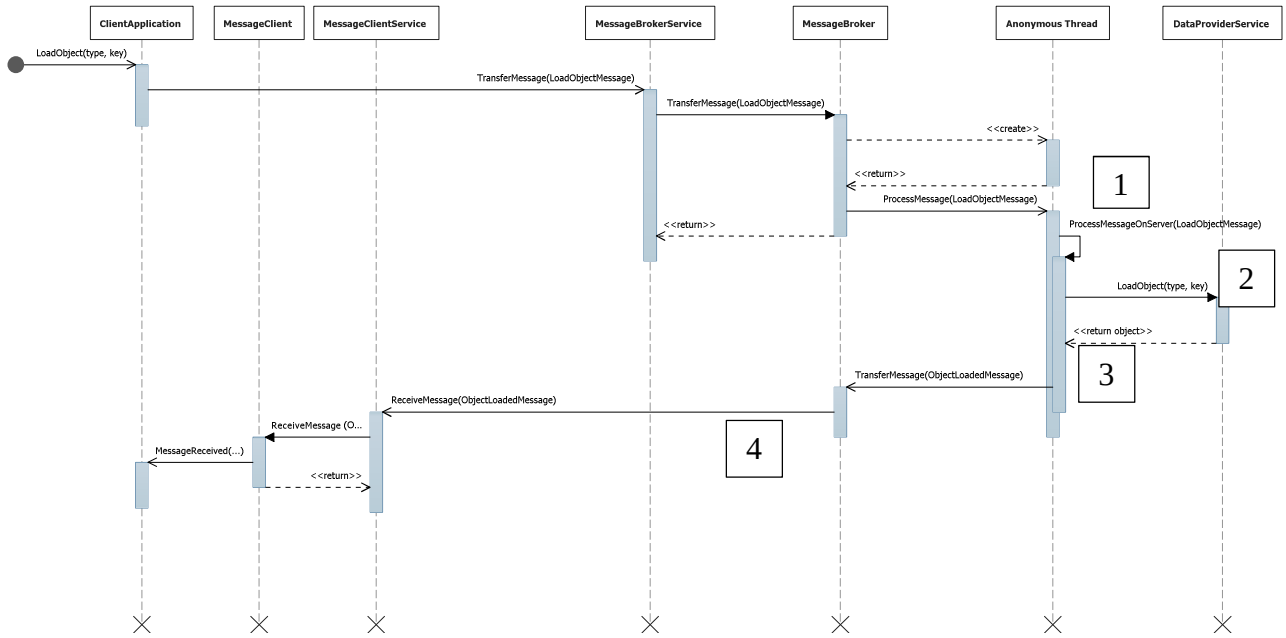


Abbildung 86: Ablauf beim Nachladen von Daten aus einem DataProvider

Wie beim Versenden anderer Nachrichten muss die Methode `TransferMessage(…)` aufgerufen werden. Da die Nachricht als Ziel den Server hat, wird innerhalb des anonymen Threads des message brokers die `ProcessMessageOnServer(…)` aufgerufen (1 in Abbildung 86). Die Methode erkennt die „LoadObjectMessage“-Nachricht und ruft auf dem DataProvider, über den `DataProviderService`, `LoadObject(…)` auf (2 in Abbildung 86). Der message broker muss vorher feststellen welchen DataProvider er nutzen muss, um ein Objekt dieses Typs laden zu können. Daraufhin gibt der DataProvider das Objekt synchron zurück (3 in Abbildung 86). Im message broker wird schlussendlich eine „ObjectLoadedMessage“ erstellt und asynchron an den Client zurückgesendet (4 in Abbildung 86).

9 Prototyp

Dieses Kapitel beschreibt die Implementierung des Prototyps der Tabletop-Steuerung innerhalb der multimodalen Analyseumgebung, sowie alle nötigen Komponenten dafür. Der Prototyp wird dabei als Client des Frameworks realisiert, um mittels eines Tabletops die Analyseumgebung für Eyetracking-Daten steuern zu können. Die Architektur wird von dem, in Kapitel 8, entworfenen Framework bereitgestellt. Die Tabletop-Steuerung wird als Client dieses Frameworks umgesetzt. Die Benutzeroberfläche und Interaktionen der Tabletop-Steuerung basieren auf dem Lösungsansatz aus Kapitel 7.

Im ersten Teil werden die Komponenten erläutert, die dem Framework für eine Analyseumgebung mit Eyetracking-Daten noch fehlen. Das ist zum einen die Datenkomponente, die die Eyetracking-Daten innerhalb der Analyseumgebung bereitstellt. Zum anderen fehlen noch die Visualisierungen, die zur Anzeige der Eyetracking-Daten benötigt werden. Diese geräteunabhängigen Komponenten wurden, wie das Framework, in Zusammenarbeit mit der Diplomarbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ implementiert.

Im Anschluss wird die prototypische Implementierung des Tabletop-Clients beschrieben, der über eine Benutzeroberfläche und geeignete Interaktionen verfügt, so dass eine Analyse der Daten möglich wird. Das beinhaltet, die Beschreibung der Implementierung der Benutzeroberfläche und der Interaktionen mit dem Surface SDK.

Im Anschluss beschreibt der letzte Teil noch den Powerwall-Client, der notwendig ist, um Visualisierungen auf der Powerwall anzuzeigen.

9.1 Datenkomponente der Analyseumgebung

Um Visualisierungen für Eyetracking-Analysen anzuzeigen, ist eine Datenkomponente notwendig, die die Datengrundlage zur Verfügung stellt. Die Datenkomponente besteht aus zwei Teilen. Das Datenmodell für Eyetracking-Daten stellt das Grundgerüst für die Daten dar. Außerdem wird ein Dataprovider benötigt wie er in Kapitel 8.3.3 beschrieben wird, damit die Daten des Datenmodells in die Analyseumgebung integriert werden können.

9.1.1 Eyetracking-Datenmodell

Das Eyetracking-Datenmodell umfasst die Daten, die während einer Eye-Tracking-Studie aufgezeichnet werden, um eine spätere Analyse vornehmen zu können. Im vorliegenden Fall basiert das konkrete Modell auf den Daten, die ein Tobii Eyetracker zur Verfügung stellt. Für die Datensätze dieses Eyetrackers wurde eine Importfunktion implementiert, mit der die Aufzeichnungen in das Eyetracking-Modell importiert werden.

Das Eyetracking-Datenmodell ermöglicht es, sowohl eine Klassenstruktur in C#, als auch ein Datenbankschema für Microsoft SQL anzulegen. Daher können die Daten in einer Datenbank gespeichert werden und gleichzeitig als Klassen in einem Programm genutzt werden.

Das Datenmodell besteht aus Entitäten, die über Relationen miteinander verbunden sind. Für die Eyetracking-Daten existieren dabei geeignete Entitäten (siehe Abbildung 87). Die Eyetracking-Daten werden in zwei Stufen in Projekten organisiert. Die erste Stufe stellt die Entität `SuperProject` dar. Ein solches kann mehrere `Project` -Entitäten enthalten. Diese Projekte bestehen aus Szenarios, welche durch die Entität `Scenario` verkörpert werden. Szenarios sind aus Stimuli aufgebaut, das sind die Bilder in einer Eyetracking-Studie. Sie

werden durch die Entität `Stimulus` repräsentiert und enthalten Metadaten, wie Größe und Name, und die Grafik als Binärdaten. Jedes Projekt des Datenmodells hat außerdem eine Anzahl von Probanden. Diese werden in der Entität `Participant` gespeichert. Eine einzelne Eyetracking-Analyse mit einem Proband und einem Stimulus wird in der Entität `Recording` gespeichert. Damit Probanden während einer Studie ein Szenario mehrmals durchlaufen können, wird zwischen Stimulus und Recording eine N:M-Beziehung erstellt. Diese kommt in der Entität `StimulusInRecording` zum Ausdruck. Die Fixationen sind die Grundlage der Visualisierungen der Eyetracking-Daten. Sie werden der Entität `Fixation` zugeordnet. Da die Entität `Recording` eine Analyse enthält, besitzt sie eine Menge an Fixationen. Außerdem wird bei den Fixationen der zugehörige Stimulus gesetzt, auf dem Fixation aufgenommen wurde.

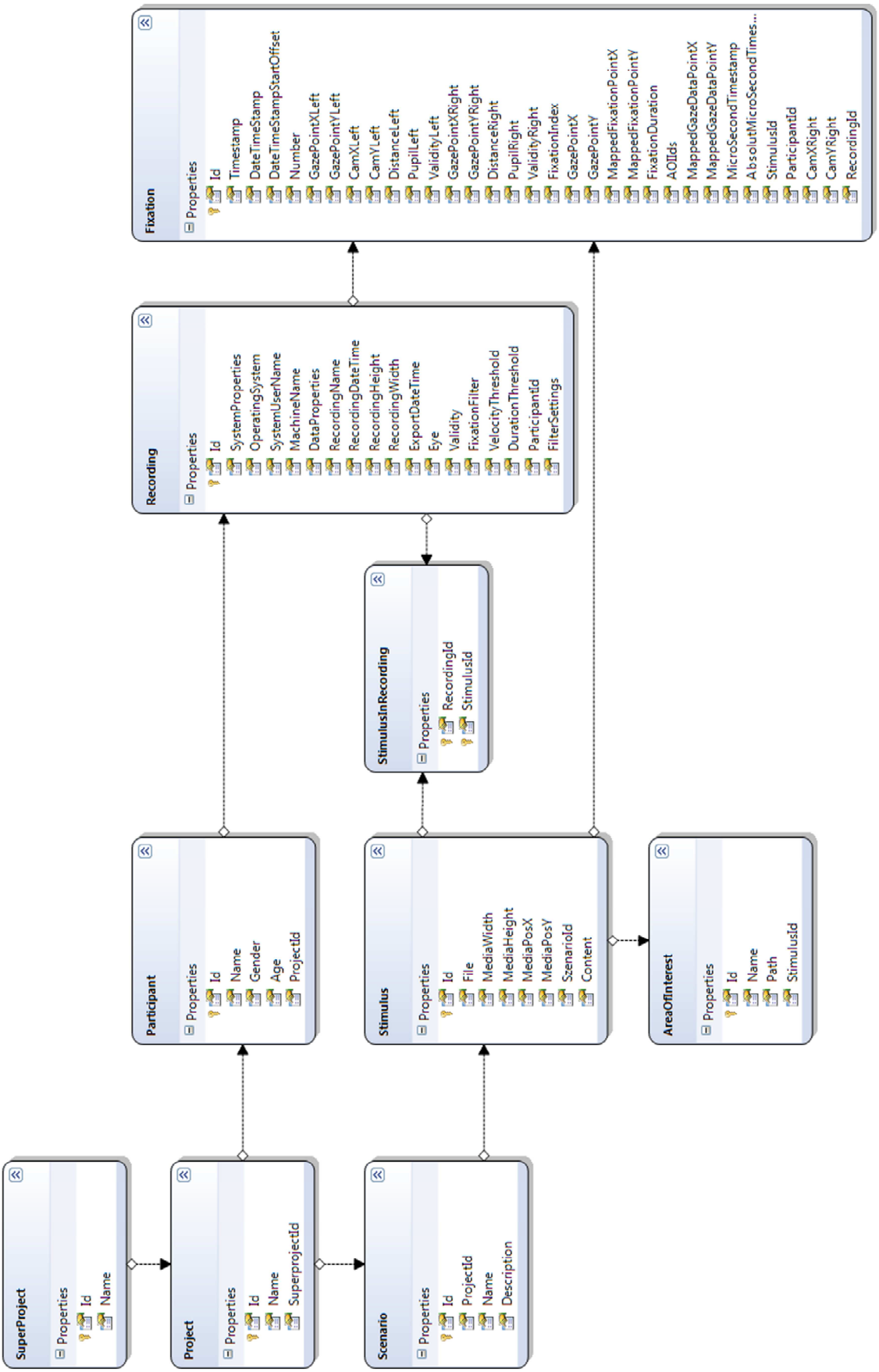


Abbildung 87: Eyetracking Datenmodell für den Daten-Provider des Prototyps
Die Pfeile zwischen den Klassen stellen 1:N-Beziehungen dar.

9.1.2 Dataprovider

Der DataProvider für Eyetracking-Daten stellt diese in der Analyseumgebung bereit, so dass sie mit Visualisierungen angezeigt werden können.

Dazu muss eine DataProvider-Komponente implementiert werden, wie es in Kapitel 8.3.3 angesprochen wurde. Diese DataProvider-Komponente stellt über das Framework Daten aus dem Eyetracking-Datenmodell für die Clients der Analyseumgebung zur Verfügung. Der Eyetracking-DataProvider besteht aus zwei Klassen: der Klasse `EyeTrackingDataProvider` und der Klasse `Program`, welche ausgeführt werden kann und so den DataProvider startet.

Der Eyetracking-DataProvider greift auf eine SQLServer-Datenbank zu, die das Eyetracking-Datenmodell (siehe Kapitel 9.1.1) als Schema hat. Die Verbindung zur Datenbank erfolgt über das Microsoft ADO.NET EntityFramework [53] und kann in der Anwendungs-konfigurationsdatei eingestellt werden.

9.2 Visualisierungen

Wie in Kapitel 8.3.4 beschrieben, können für die Clients Visualisierungen als Plugins implementiert werden. Im Rahmen des Prototyps für Eyetracking-Daten wurden eine Auswahl an Visualisierungen implementiert. Diese Visualisierungen sind die Stimulus-Visualisierung, die Heatmap-Visualisierung und die „Gaze-Duration-Sequence“-Visualisierung.

Stimulus-Visualisierung

Diese Visualisierung zeigt ausschließlich den Stimulus als Bild an. Mit dieser Visualisierung kann nicht interagiert werden.

Heatmap-Visualisierung

Die Heatmap-Visualisierung zeigt im Hintergrund ebenfalls den Stimulus als Bild an. Im Vordergrund wird eine halbtransparente Heatmap auf Basis einer ausgewählten Menge an Fixationen angezeigt. Daher werden Gegenden mit vielen Fixationen in Rot dargestellt und Gegenden mit wenigen in grün. Die Heatmap wird erzeugt, indem für jede Fixation ein Graustufen-Pinselpunkt in eine Intensitätskarte gezeichnet wird. Diese Intensitätskarte wird über einen Pixelshader und einer vorgegebenen Textur zur Heatmap transformiert. Die Textur enthält für jeden Intensitätswert eine Farbe. Auch die Heatmap-Visualisierung beherscht keine weiteren Interaktionen.

Gaze-Duration-Sequenz-Visualisierung

Diese Visualisierung zeigt ein „Gaze Duration Sequence“-Diagramm an. Auf der rechten Seite ist eine Legende, die die Probanden und deren Farben anzeigt. Auf diesem Diagramm können die vertikalen Achsen für die AOIs umsortiert werden.

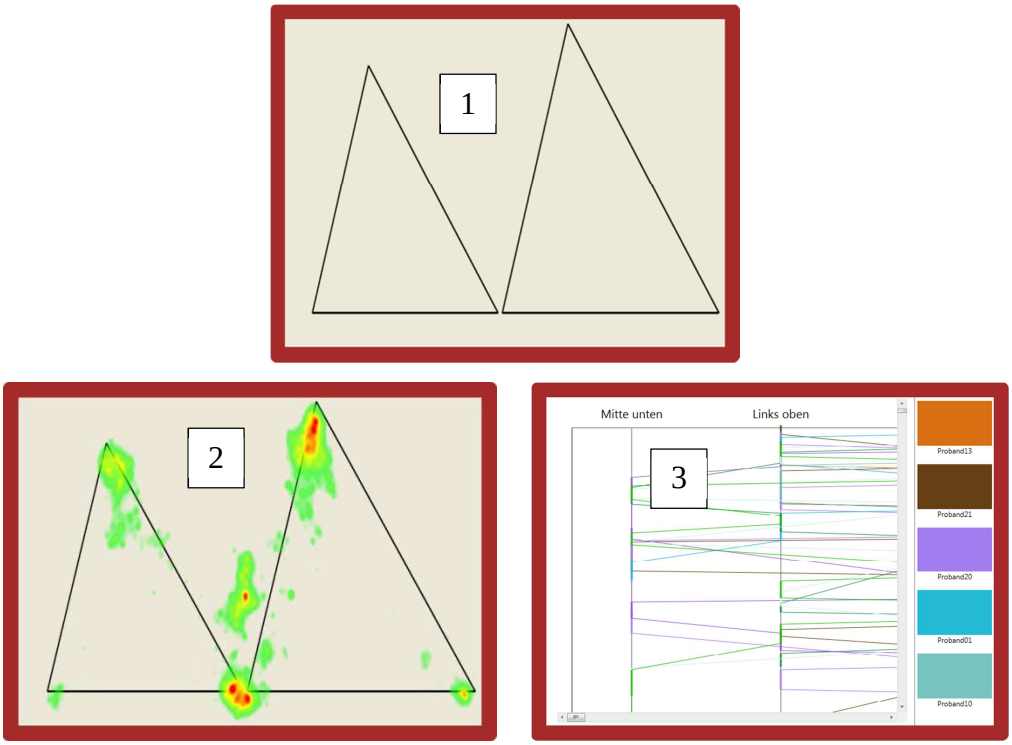


Abbildung 88: Visualisierungen für Eyetracking-Daten
1) Stimulus-Visualisierung, 2) Heatmap-Visualisierung, 3) Gaze-Duration-Sequenz-Visualisierung

9.3 Tabletop-Client

Dieser Teil beschreibt den Prototypen des Tabletop-Client. Er ist in C# auf Basis des .NET 4 Frameworks geschrieben. Seine Benutzeroberfläche baut auf dem Windows Presentation Framework(WPF) auf und nutzt gleichzeitig das Surface SDK 2. Die eingesetzte Hardware ist der Microsoft Pixelsense-Tabletop, dieser wird von Samsung als SUR40 hergestellt (siehe Abbildung 89).

Abbildung 89: Der Microsoft Pixelsense des VIS

Der Client-Prototyp setzt ausgewählte Konzepte aus dem Lösungsansatz (Kapitel 7) um. Zu Anfang wird die Nutzung des Frameworks kurz wiederholt. Danach wird die Benutzeroberfläche des Tabletop-Clients beschrieben. Auf die Benutzeroberfläche folgt ein Abschnitt über die Implementierung von Gesten für den Client. Darauf wird beschrieben, wie die Interaktionen mit Tangibles umgesetzt wurden.

9.3.1 Nutzung des Frameworks

Jeder Client der Analyseumgebung muss in das Framework (siehe Kapitel 8) eingebunden werden. Da der Tabletop-Client in C# geschrieben ist, kann er die vorgefertigten Klassen des Frameworks für Clients verwenden. Die Klasse `ClientApplication` des Frameworks bietet dem Client die Funktionalität das Framework zu nutzen. Sie ermöglicht dem Client den Webservice einfacher zu nutzen.

Um Nachrichten mit der Analyseumgebung auszutauschen, bietet die Klasse `ClientApplication` Methoden zum Senden der Nachrichten und Ereignisse zum Empfangen der Nachrichten.

Für die Datenabstraktion werden die Datencontainer eingesetzt. Diese werden ebenfalls vom Framework zur Verfügung gestellt und durch die Klasse `AbstractDataContainer` repräsentiert. Jeder Datenaustausch zwischen den verschiedenen Clients geschieht über diese Datencontainer.

9.3.2 Benutzeroberfläche

Die Benutzeroberfläche besteht aus vorrangig aus einem `ScatterView` des Surface SDK. Dieses Steuerelement ermöglicht es, eine Oberfläche mit frei verschiebbaren Elementen zu erstellen, wie es im Lösungsansatz beschrieben wird. Unter dem `ScatterView` liegt ein `TagVisualizer`, ebenfalls aus dem Surface SDK. Der `TagVisualizer` ermöglicht die Interaktion mit Tangibles, die mit visuellen Markern, den Byte-Tags von Microsoft, ausgestattet wurden.

Auf dem `ScatterView` können beliebig viele `ScatterViewItems` liegen. Alle Elemente, abgesehen von den Anzeigeelementen für die Tangibles, sind `ScatterViewItems` oder von diesen abgeleitet.

Die festen Elemente der Benutzeroberfläche, wie sie im Lösungsansatz beschrieben werden, sind statische `ScatterViewItems`. Diese werden statisch über die XAML-Datei zum `ScatterView` hinzugefügt. Diese Elemente umfassen die Powerwallanzeige, den Datenexplorer und den Administrationsdialog. Der Admin-Dialog kommt nicht im Lösungsansatz vor, da er keine Bedeutung für eigentliche Funktion des Programmes hat.

Ferner können dynamische Elemente auf dem `ScatterView` liegen. Bei diesen handelt es sich um die Visualisierungscontainer des privaten Raumes. Für die Visualisierungscontainer selbst gibt es zwei Arten. Erstens die Visualisierungscontainer des privaten Raums, sie werden durch die Klasse `SurfaceVisWindow` dargestellt. Zweitens die Visualisierungscontainer auf der Powerwallansicht, repräsentiert durch die Klasse `PowerwallVisWindow`. Abstrahiert werden die Visualisierungscontainer durch die Oberklasse `AbstractVisWindow`.

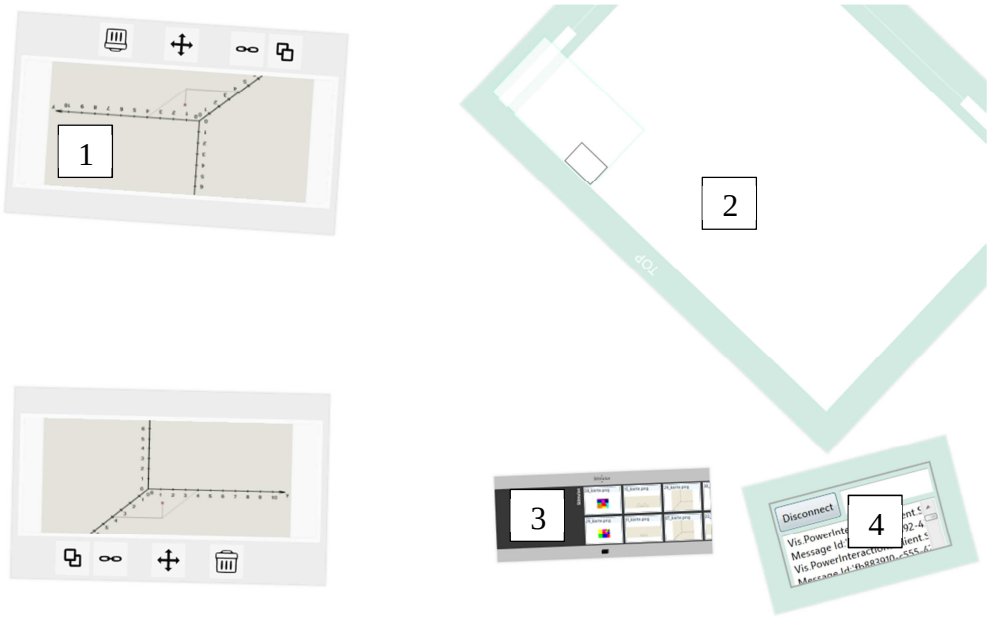


Abbildung 90: Benutzeroberfläche des Clients
1) privater Visualisierungscontainer, 2) Powerwallansicht
3) Datenexplorer, 4) Administrationsdialog

Powerwallansicht
Die Powerwallansicht (siehe Abbildung 91) besteht aus einem Canvas innerhalb eines `SurfaceScrollViewers`, einem Steuerelement aus dem Surface SDK. Das Canvas hat die Auflösung der Powerwall. Der ScrollViewer beschränkt das Canvas auf die Größe des umschließenden `ScatterViewItems`, in dem die Powerwallansicht dargestellt wird. Dadurch wird nur ein Bereich der Powerwall, sein Viewport, angezeigt. Um den Bereich zu verschieben, unterstützt der `SurfaceScrollViewer` eine Pan-Geste. Außerdem kann der Bereich auch gezoomt werden. Dafür musste eine Pinch-Geste nachimplementiert werden. Die allgemeine Vorgehensweise für die Implementierung von Gesten kann in Gesteninteraktion(9.3.3) nachgelesen werden.

Die Minimap (siehe Abbildung 91) wird ebenfalls durch ein Canvas gelöst, das ein gerendertes Abbild des großen Canvas, der Powerwall-Darstellung, verkleinert darstellt. Außerdem wird mit einem Rechtecks-Element der Viewport angezeigt. Die Aktualisierung der Minimap funktioniert über Ereignisse des ScrollViewers, die ausgelöst werden, wenn dieser durch den Benutzer verändert wird.

Innerhalb des Canvas werden die Visualisierungscontainer der Powerwallansicht angezeigt. Diese werden durch die `PowerwallVisWindow` Klasse abgebildet. Die Visualisierungscontainer auf der Powerwallansicht sind eine Darstellung des öffentlichen Raums. Daher müssen sie in die Analyseumgebung und somit auch auf die Powerwall synchronisiert werden. Die Synchronisierung geschieht über die Nachrichten des Frameworks der Analyseumgebung (siehe 8.3.1).

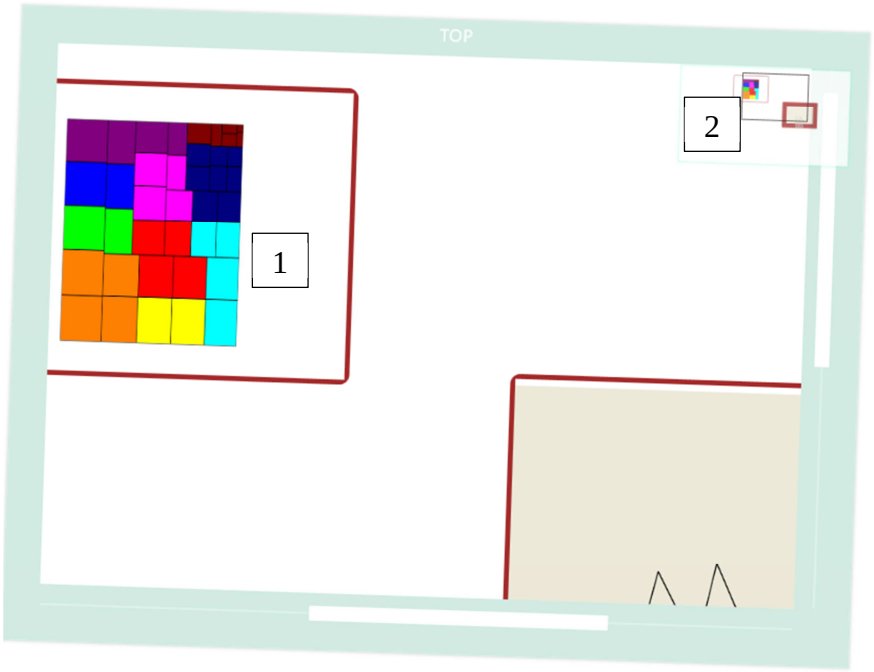


Abbildung 91: Powerwallansicht
1) öffentlicher Visualisierungscontainer, 2) Minimap

Neue Visualisierungscontainer werden durch eine `NewViewMessage` bekanntgegeben. Neue Container haben meist auch neue Daten, diese werden durch die `NewDataMessage` an die Analyseumgebung weitergegeben. Neuen Daten folgen meist `ChangeDataMessages`, mit denen die Daten eines Visualisierungscontainers aktualisiert werden.

Beim Verschieben eines Visualisierungscontainer wird die `MoveViewMessage` mit den Koordinaten versendet. Die Skalierung eines Containers wird über die `ResizeViewMessage`, die die neue Größe enthält, weitergegeben.

Wenn zwei Visualisierungscontainer gekoppelt werden, so erfolgt dies über eine `CoupleViewMessage` an die Analyseumgebung. Beim Wechsel der Visualisierung eines Containers wird eine `ChangeVisualizationMessage` gesendet. Wurden die Daten einer Visualisierung gefiltert, dann wird der Analyseumgebung mit der `FilterDataMessage` über die Filtereigenschaften informiert.

Das Löschen eines Visualisierungscontainers wird durch die Nachricht `RemoveViewMessage` weitergeleitet.

Datenexplorer
Der Datenexplorer wird über ein `LibraryContainer` des Surface SDK gelöst. Der `LibraryContainer` bietet von sich aus Drag&Drop-Interaktionen an. Der Datenexplorer zeigt Daten aus der Analyseumgebung an, die als Visualisierungen dargestellt werden können. Jedes Datenelement hat eine Vorschau.

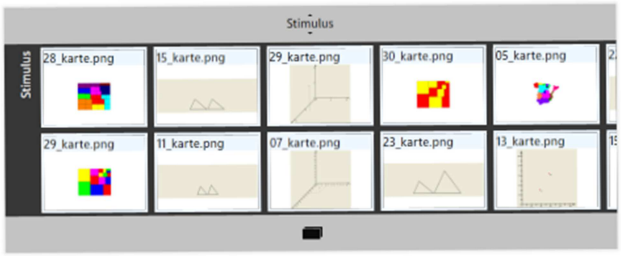


Abbildung 92: Datenexplorer mit Stimuli

Administrationsdialog
Dieser Dialog dient dazu, den Client mit dem Server zu verbinden. Es kann zu einem beliebigen Server oder dem Server, der in der Konfigurationsdatei angeben ist, verbunden werden. Außerdem zeigt der Dialog Debug-Informationen für die Analyseumgebung an (siehe Abbildung 92).

Visualisierungscontainer
Die grundlegende Funktion der Visualisierungscontainer wird durch die Klasse `AbstractVisWindow` bereitgestellt (Abbildung 93). Die Klasse ist jedoch nicht abstrakt implementiert, da abstrakte Klassen sich nicht mit dem Designer des Visual Studios vertragen. Die Klasse hat einen Platzhalter für das Steuerelement, das die Visualisierung enthält. Die Klasse stellt ebenfalls einen Container für die Daten bereit, die von der Visualisierung angezeigt werden. Außerdem lädt es mit dem Plugin-System für Visualisierungen aus dem Framework die Steuerelemente der Visualisierungen. Mit der Methode `GetVisualizationsByDataType(…)` holt sich die Klasse das Steuerelement für eine Visualisierung, die mit den zugewiesenen Daten kompatibel ist. Dieses wird instanziiert und auf dem Platzhalter der `AbstractVisWindow` Klasse gesetzt. Die Unterklassen dieser Klasse sind dann dafür zuständig, dass das Steuerelement angezeigt wird.

AbstractVisWindow
#_Visualizations : List<Vis.PowerInteractions.Client.Visualizations.IVisualization>
+VisualisationContentControl() : ContentControl +DisplayedDataType() : string +DisplayedVisualization() : IVisualization +WindowGlow() +ChangeVisualization(eing. visualizationNameToChangeTo : string) +ChangeVisualization(eing. plusOrMinusOne : int) +Refresh() +VisualizationChanged() : EventHandler<System.EventArgs> +Data() : AbstractDataContainer

Abbildung 93: `AbstractVisWindow`
Stellt die grundlegenden Funktionen eines Visualisierungscontainer bereit.

Die `SurfaceVisWindow` Klasse ist eine Unterklasse von `AbstractVisWindow` und dient als Visualisierungscontainer im privaten Raum (siehe Abbildung 94). Sie wird als `ScatterViewItem` auf dem `ScatterView` angezeigt. Da das `ScatterViewItem` durch die so genannten „Manipulations“ schon durch das Surface SDK verschoben, rotiert und skaliert werden kann, müssen diese Funktionen nicht implementiert werden. Die Funktionsknöpfe des Visualisierungscontainers am unteren Rand werden über `SurfaceButtons` des Surface SDKs gelöst. Die Buttons für die Duplizierung und Kopplung starten eine Drag&Drop-Operation. Das Surface SDK bietet im Vergleich zu WPF

eine erweiterte Drag&Drop-Unterstützung, diese wird durch die Klasse `SurfaceDragDrop` zur Verfügung gestellt.

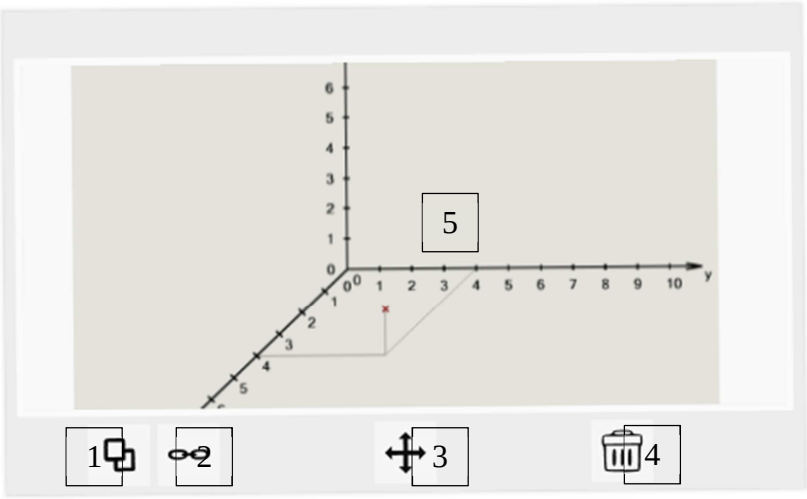


**Abbildung 94: Privater Visualisierungscontainer (`SurfaceVisWindow`)
1: Duplizierenknopf, 2: Koppelnknopf, 3: Interaktionsmodus, 4: Löschenknopf,
5: Visualisierung**

Die `PowerwallVisWindow` Klasse ist die zweite Unterklasse von `AbstractVisWindow`. Sie ist für die Anzeige von Visualisierungscontainern auf der Powerwallansicht zuständig (siehe Abbildung 95). Die Steuerelemente, die für die Interaktion zuständig sind, sind die gleichen wie bei `SurfaceVisWindow`. Das Aussehen wurde aber an die des Powerwallclient, der in Kapitel 9.4 erwähnt wird, angepasst. Die Interaktions-Steuerelemente werden nach einer Zeit ausgeblendet, so dass die Visualisierungscontainer auf der Powerwallanzeige wie die auf der Powerwall aussehen. Die Gesten für Verschieben und Skalieren mussten mit dem Touchframework aus WPF neu implementiert werden, da die Manipulations innerhalb des `SkatterViewItems` nicht funktionsfähig waren.

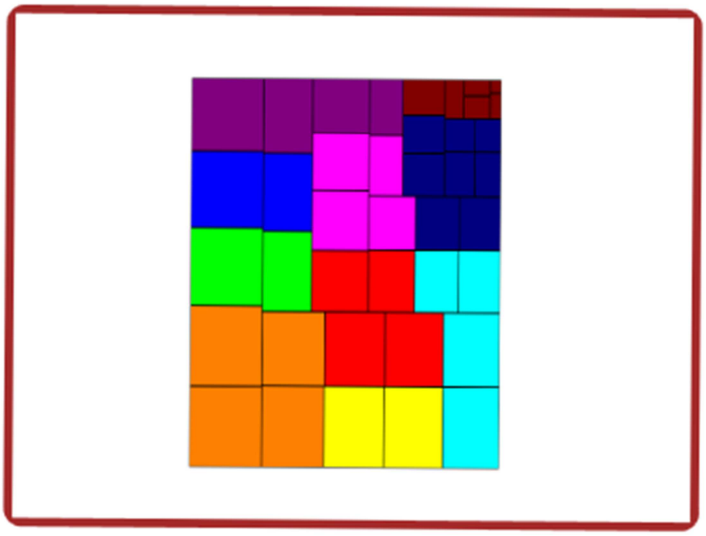


**Abbildung 95: Öffentlicher Visualisierungscontainer(`PowerwallVisWindow`)
mit ausgeblendeten Steuerelementen**

Das Verschieben eines SurfaceVisWindows auf die Powerwallansicht soll entsprechend Kapitel 7.3 möglichst nahtlos funktionieren. Dazu führt das SurfaceVisWindow an dem Punkt, an dem es verschoben wird, ein Hittesting aus, um herauszufinden ob die Powerwallansicht sich unter dem Punkt befindet. Wenn dies zutrifft, wird das SurfaceVisWindow entfernt und ein PowerwallVisWindow angelegt. Die Verschiebeinteraktion wird dann an das PowerwallVisWindow abgegeben, indem dessen Berührpunkt weitergegeben wird. Der Berührpunkt wird dabei durch ein TouchDevice des WPF repräsentiert. Umgekehrt funktioniert die Interaktion analog.

9.3.3 Gesteninteraktion

Die Gesten des Clients werden über das Touchframework von WPF implementiert. Vorgefertigte Gesten werden durch die Manipulations aus WPF bereitgestellt. Diese ermöglichen es, hauptsächlich geometrische Transformationen durch Gesten durchzuführen. Weiterhin bietet das Surface SDK Methoden an, um Wisch- und Tippgesten umzusetzen. Gesten für die es keine Unterstützung gibt, müssen über das grundlegende Touchframework umgesetzt werden. Dieses Framework funktioniert über Ereignisse, die Berührungen und das Verschieben von Fingern melden. Dabei bekommt jeder Berührungspunkt ein TouchDevice. Auf diese Art und Weise können die verschiedenen Finger auf dem Tabletop auseinandergehalten werden.

Die Vorgehensweise bei der Erkennung neuer Gesten ist die Folgende: Zuerst werden Bereiche festgelegt, die nur von einem Benutzer gleichzeitig verwendet werden. So können sich die Benutzer nicht gegenseitig mit ihren Gesten stören. Diese Bereiche werden sinnvoll gewählt, z.B. einzelne Visualisierungscontainer oder die Fläche der Powerwallansicht für das Paning. Der nächste Schritt ist ein einfacher Zustandsautomat. Da die entworfenen Gesten durch die Anzahl der eingesetzten Finger auseinander gehalten werden können, sind die ersten Zustände durch die Anzahl der Finger auf den genannten Bereichen definiert. Umgesetzt wird dies durch das Zählen der TouchDevices mit dem TouchDown Ereignis für die jeweiligen Bereiche. Danach wird mit dem TouchMove Ereignis die eigentliche Durchführung der Geste erkannt. Da die Geste schon im Schritt davor festgelegt ist, muss dabei nur noch darauf geachtet werden, dass die Geste wie erwartet abläuft. Falls das nicht der Fall ist, wird die Geste abgebrochen.

9.3.4 Tangibleinteraktion

Für den Microsoft Pixelsense gibt es nur eine Möglichkeit Tangibles zu nutzen. Dafür werden die Byte-Tags von Microsoft an den entsprechenden Objekten befestigt. Als physikalische Interaktionskonzepte wurden Plexiglas Zylinder hergestellt. Auf eine der Grundflächen der Zylinder ist ein Byte-Tag aufgeklebt (siehe Abbildung 96). Die Transparenz des Plexiglases ermöglicht es das Display besser zu sehen als bei opaken Materialien. Der Zylinder wurde gewählt, da er auf den Tisch gestellt werden kann und zum Anfassen eine angenehme Form hat. Der Tabletop erkennt das Tangible über den Byte-Tag, wenn es auf den Tisch platziert wird. Damit der Benutzer die verschiedenen Tangibles unterscheiden kann könnten Abbildungen auf die Oberseite geklebt werden. Für den Prototyp werden die Tangibles mit Markern beschrieben.

Abbildung 96: Tangible mit Byte-Tag

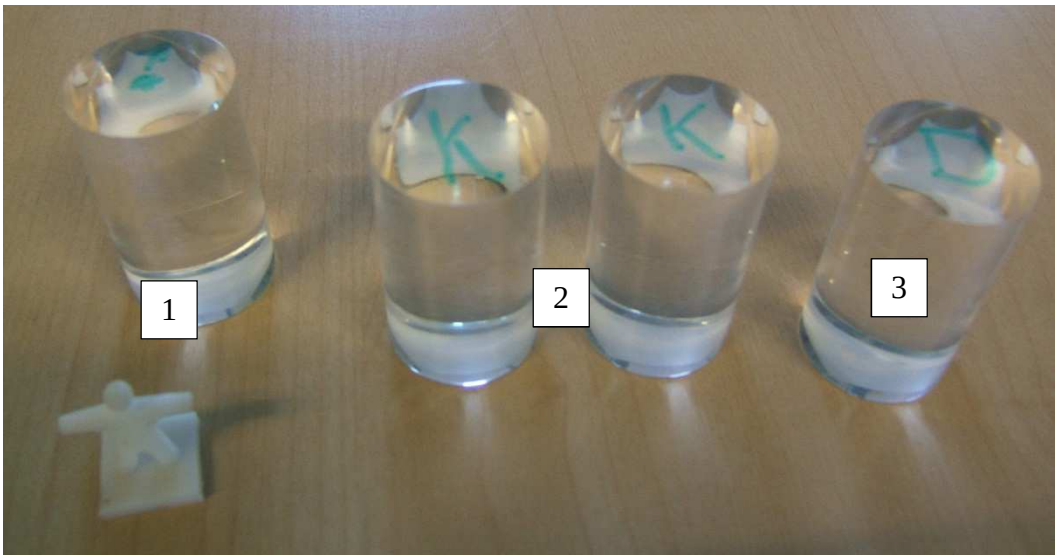


Abbildung 97: Verschiedene Tangibles des Client
1) Datenrepräsentationstangibles,
2) Kopplungstangiblepaar, 3) Duplizierungstangible

Die verschieden Tangible-Typen werden durch unterschiedliche Bereiche der Byte-Tags festgelegt. Für die Datenrepräsentationstangibles werden die ersten 50 Tags reserviert. Für die Filter- und Duplizierungstangibles werden jeweils immer 10 Tags reserviert. Die Kopplungs-Tangibles erhalten 20 Tags. Da die Kopplungstangibles immer paarweise eingesetzt werden, sind die Tags mit gleichem Modulo 10 Wert gepaart. Den Tags der Id-Karte von den privaten Ablagen werden ebenfalls 10 Tags zugeordnet.

Die Tangibles werden in die Software über sogenannte `TagVisualizations` des Surface SDKs eingebunden. Die `TagVisualizations` werden auf dem `TagVisualizer` der Benutzeroberfläche angezeigt. Die `TagVisualizations` sind beliebige Steuerelemente auf der Benutzeroberfläche und werden normalerweise anhand des Tangible ausgerichtet und positioniert. Sie werden angezeigt sobald das Tangible auf dem Tabletop steht und beim Wegnehmen wieder entfernt. Jeder der Tangible-Typen bekommt über die Tags eine Klasse zugeordnet, die von der `TagVisualization`-Klasse abgeleitet ist.

- Duplizierungstangible: `DuplicateTagVisualization`-Klasse
- Kopplungstangible: `CoupleTagVisualization`-Klasse
- Filtertangible: `FilterTagVisualization`-Klasse
- Datenrepräsentationstangible: `GenericTagVisualization`-Klasse
- Id-Karte der privaten Ablage: `DepotTagVisualization`-Klasse

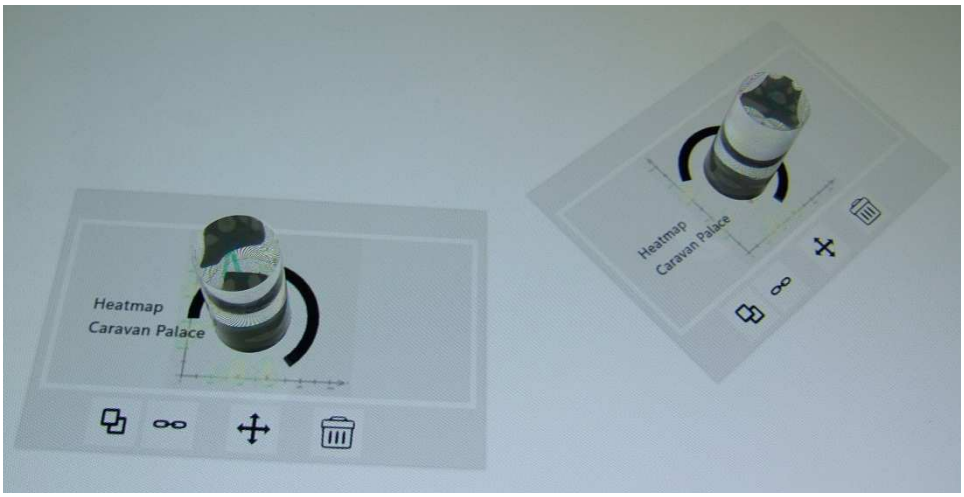


Abbildung 98: Kopplung von zwei Visualisierungscontainer mit Tangibles Die Kreisanimation zeigt den Fortschritt an.

Sichtbare TagVisualizations haben nur die Datenrepräsentationstangibles, die Filtertangibles und die private Ablage.

- **Datenrepräsentationstangible**
  Die `TagVisualization` dieses tangible wird angezeigt, wenn es auf freier Fläche des privaten Raums abgelegt wird. Innerhalb der `TagVisualization` befindet sich ein Dialog, der es ermöglicht dem tangible Daten zuzuordnen. Über den Dialog können dem tangible entweder Daten für das Filterkonzept oder Visualisierungscontainer zugeordnet werden. Wurde dem tangible ein Visualisierungscontainer zugeordnet, dann wird die `TagVisualization` des tangible unsichtbar und der Visualisierungscontainer mit der `TagVisualization` verschoben und rotiert. Das bedeutet wiederum, dass der Visualisierungscontainer direkt vom tangible gesteuert werden kann. Verliert die `TagVisualization` ihr tangible wird sie entfernt, vorher kann aber über das Ereignis `LostTag` der Visualisierungscontainer ausgeblendet werden. Wird das Tangible wieder hingestellt, dann blendet die neue `TagVisualization` über das Ereignis `GotTag` den Visualisierungscontainer wieder ein. Eine andere Möglichkeit dem Tangible einen Visualisierungscontainer zuzuordnen, ist durch längeres Abstellen des Tangible auf den Container.

- **Filtertangible**
  Die `TagVisualization` des Tangibles wird angezeigt, wenn das Tangible auf eine freie Fläche des privaten Raums abgelegt wird und ihm ein Visualisierungscontainer zugeordnet wurde. Dem Tangible kann ein Visualisierungscontainer zugeordnet werden, indem das Tangible kurzzeitig auf den Container abgestellt wird. Die Anzeige der `TagVisualization` stellt entsprechend des Lösungsansatzes einen Kreis dar. Die Datenrepräsentationstangibles suchen über Hittesting eine `TagVisualization` des Filtertangible. Finden sie eine wird der `TagVisualization` mitgeteilt wo sich das Datenrepräsentationstangibles befindet und sie kann entsprechend des Filterkonzepts einen zugehörigen Filter anzeigen.

- **Private Ablage**
  Die `TagVisualization` der privaten Ablage enthält ein `LibraryContainer` des SurfaceSDK. Visualisierungscontainer können wie bei der Powerwallansicht in

den Container verschoben werden. Nur wird hier ein Drag&Drop-Element erzeugt, sobald sich der Visualisierungscontainer im `LibraryContainer` befindet. Herausgenommen können Visualisierungscontainer wieder, indem sie aus dem `LibraryContainer` per Drag&Drop herausgezogen werden. Dies funktioniert wie bei dem Datenexplorer.

Die `TagVisualizations` aller Tangible-Typen müssen bei ihrer Interaktion Objekte auf dem Display erkennen können. Dies wird über Hittesting umgesetzt. In WPF kann durch Hittesting ein Steuerelement unterhalb eines Punktes gefunden werden. Um bestimmte Steuerelemente, z.B. Visualisierungscontainer, zu finden führen die `TagVisualizations` der Tangibles Hittesting durch sobald sie erzeugt oder verschoben werden. Dies wird durch die Klasse `VisWindowHitTester` ermöglicht.

Die registrierten Daten der Tangibles, z.B. welche Visualisierungscontainer zugeordnet wurden, werden in der lokalen Datenhaltung gespeichert.

9.3.5 Lokale Datenhaltung

Die lokale Datenhaltung verwaltet die Daten des Client. Die Datenhaltung sorgt dafür, dass die Daten mit der Analyseumgebung synchronisiert werden und unterscheidet dabei, welche Daten im öffentlichen Raum benötigt werden und welche nur lokal im privaten Raum vorhanden sind. Außerdem verwaltet die Datenhaltung die Kopplung von Visualisierungscontainern und die Filterung der Daten einer Visualisierung. Die Hauptklasse der lokalen Datenhaltung ist der `LocalDataManager`.

LocalDataManager
+PublicDataContainers : ExtendedObservableCollection<Vis.PowerInteractions.Data.AbstractDataContainer> +PrivateDataContainers : ExtendedObservableCollection<Vis.PowerInteractions.Data.AbstractDataContainer> +PowerwallVisualizationContainers : ExtendedObservableCollection<Vis.PowerInteractions.Client.Surface.PowerwallVisWindow> +SurfaceVisualizationContainers : ExtendedObservableCollection<Vis.PowerInteractions.Client.Surface.SurfaceVisWindow>
+TryDeleteDataContainer(eing. dataContainer : AbstractDataContainer) +RegisterPrivatelyAssignedDataContainer(eing. dataContainer : AbstractDataContainer, eing. visContainer : AbstractVisWindow) +UnregisterPrivatelyAssignedDataContainer(eing. dataContainer : AbstractDataContainer, eing. visContainer : AbstractVisWindow) +RegisterPublicilyAssignedDataContainer(eing. dataContainer : AbstractDataContainer, eing. visContainer : AbstractVisWindow) +UnregisterPublicilyAssignedDataContainer(eing. dataContainer : AbstractDataContainer, eing. visContainer : AbstractVisWindow) +SetVisualizationCoupled(eing. source : AbstractVisWindow, eing. target : AbstractVisWindow, eing. type : CouplingType) : bool +RemoveCoupling(eing. visualization : AbstractVisWindow) +IsCoupled(eing. visContainer1 : AbstractVisWindow, eing. visContainer2 : AbstractVisWindow) : bool

Abbildung 99: `LocalDataManager`-Klasse

Die Eigenschaft `PublicDataContainers` des `LocalDataManager` enthält alle Datencontainer der Visualisierungen des öffentlichen Raums. Werden Daten zu dieser Eigenschaft hinzugefügt, dann werden die neuen Datencontainer automatisch mit der `NewDataMessage` an die Analyseumgebung weitergegeben. Analog dazu speichert die `PrivateDataContainers`-Eigenschaft die Datencontainer des privaten Raums.

Die Eigenschaft `PowerwallVisualizationContainers` enthält die Visualisierungscontainer auf der Powerwallansicht. Ein Hinzufügen oder Löschen eines Visualisierungscontainer aus der Liste bewirkt, dass er auf dem Canvas der Powerwallansicht dargestellt oder entfernt wird. Die Eigenschaft `SurfaceVisualizationContainers` speichert die Visualisierungscontainer des privaten Raums. Hier werden beim Ändern der Liste die Visualisierungscontainer von dem `ScatterView` hinzugefügt oder von diesem entfernt.

Mit den „registries“ für zugewiesene Datencontainer, die mit `Register*AssignedDataContainer` und `Unregister*AssignedDataContainer` bedient werden, werden die zu Visualisierungscontainern zugewiesenen Datencontainer registriert. Das ermöglicht es, die Datencontainer nur zu löschen, wenn sie nicht mehr einem Visualisierungscontainer zugeordnet sind. Dabei wird, wie bei den anderen Eigenschaften, zwischen privaten und öffentlichen Räumen unterschieden. Gelöscht werden die `DatenContainer` mit `TryDeleteDataContainer(…)`.

Die Methode `SetVisualizationCoupled(…)` koppelt zwei Visualisierungscontainer und die Methode `RemoveCoupling(…)` entfernt diese wieder.

9.4 Kinect-Powerwall-Client

Um tatsächlich Visualisierungen auf der Powerwall anzuzeigen, wird neben dem Tabletop-Client auch ein Powerwall-Client benötigt.

Dieser Powerwall-Client hat zunächst nur die Funktion, die Visualisierungen die zur Analyseumgebung hinzugefügt werden auf der Powerwall anzuzeigen. Das kann z.B. durch Hinzufügen einer Visualisierung zur Powerwallanzeige innerhalb des Tabletop-Clients geschehen. Weiterhin wartet der Client auf Nachrichten des Frameworks und passt seine Darstellung entsprechend an.

Die Erweiterung des Powerwall-Clients ist der Prototyp der Diplomarbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ [46]. Hier gibt es zusätzlich die Möglichkeit die Powerwall mit Freihandgesten über eine Microsoft Kinect zu steuern.

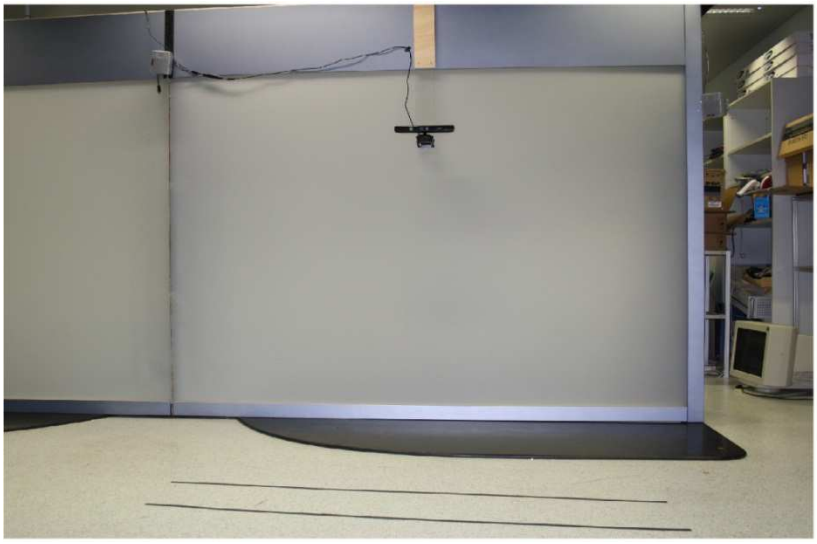


Abbildung 100: Aufbau für den Powerwall-Client

10 Zusammenfassung und Ausblick

Das letzte Kapitel fasst die Ergebnisse der Diplomarbeit zusammen und wichtige inhaltliche Punkte werden wiederholt. Es werden einige Aspekte noch einmal bewertet und Stellen angesprochen, die positiv oder negativ aufgefallen sind. Zum Schluss wird ein Ausblick in Bereiche aus dieser Arbeit gegeben, an denen weitergearbeitet werden kann.

10.1 Zusammenfassung und Bewertung

Dieses Kapitel fasst zuerst die Arbeit noch einmal zusammen. Zu Beginn wurde eine Anforderungsanalyse durchgeführt, bei der unter anderem zwei Szenarien für eine Analyseumgebung entstanden sind. Daraufhin wurde ein Konzept für eine Analyseumgebung für Powerwalls und ein Interaktionskonzept für eine Tabletop-Steuerung innerhalb der Analyseumgebung entworfen. Dabei wurde das Interaktionskonzept in einer Vorstudie evaluiert und verbessert. Danach wurde das Konzept für die Analyseumgebung in einem Framework umgesetzt und das Interaktionskonzept in einem Prototyp implementiert. Der Prototyp nutzt dabei das Framework für die Analyseumgebung.

10.1.1 Zusammenfassung

Das Ziel dieser Arbeit war es ein Interaktionskonzept für die Steuerung einer Analyseumgebung für die Powerwall mit einem Tabletop zu entwickeln. Die Analyseumgebung soll dabei Visualisierungen zur Analyse auf der Powerwall anzeigen. Um dies zu ermöglichen wurde auch ein Konzept für die Analyseumgebung entwickelt.

Um einen Überblick über mögliche Funktionen einer solchen Analyseumgebung und deren Einsatz zu bekommen, wurde eine Anforderungsanalyse durchgeführt. Diese Anforderungsanalyse wurde unter anderem durch Beobachtung der Analyse einer Eyetracking-Studie durchgeführt. Die Anforderungsanalyse führte zu zwei Szenarien für die Anwendung der Analyseumgebung:

- **Szenario für die Auswertung von Eyetracking-Studien**
  Das erste Szenario ist aus dem Bereich der Informationsvisualisierung. In dem Szenario wird die Analyseumgebung bei der Auswertung einer Eyetracking-Studie eingesetzt. Bei einer solchen Auswertung werden unter Umständen viele Visualisierungen geöffnet. Diese müssen verglichen werden, um Muster zu finden. Darum wird die Auswertung an einer Powerwall durchgeführt.

- **Szenario für MegaMol**
  Das zweite Szenario ist aus dem Bereich der wissenschaftlichen Visualisierung. Bei diesem Szenario wird versucht die Analyseumgebung zu nutzen, um große Moleküle anzuzeigen.

Anhand der Szenarien wurden Anforderungen an die Analyseumgebung und die Tabletop-Steuerung aufgestellt.

Im Anschluss wurde anhand der Anforderungen ein Konzept für die Analyseumgebung mit Powerwall entwickelt. Dabei wurde analysiert welche Geräte eingesetzt werden. Es wurde erkannt, dass sowohl Geräte zur direkten Steuerung der Powerwall als auch Geräte, die die Powerwall fernsteuern, eingesetzt werden können. Im Weiteren wurde der grundsätzliche Aufbau des Frameworks für die Analyseumgebung mit Kommunikation und Datenabstraktion festgelegt. Das Framework sollte außerdem in besonderem Maße erweiterbar sein. Zum

Schluss wurde festgelegt wie ein Tabletop in die Analyseumgebung eingebaut werden kann. Der Tabletop kann sowohl einen öffentlichen Bereich für die Steuerung der Powerwall als auch einen privaten Bereich für sich selbst besitzen. Innerhalb des privaten Bereichs ermöglicht der Tabletop durch seine Technik einen Mehrbenutzerbetrieb. Es können daher Gruppen von Benutzern an einem Tabletop arbeiten. Für den Mehrbenutzerbetrieb sollten Funktionen wie Koppeln oder Ablagen unterstützt werden. Außerdem wurde identifiziert welche Aufgaben die Interaktionsmodalitäten des Tabletops, wie Gesten oder Tangibles, übernehmen können.

Auf Basis des Konzepts wurde ein Framework für die Analyseumgebung entworfen und implementiert. Das Framework wurde in .NET 4 entwickelt und nutzt für eine mögliche Plattformunabhängigkeit Webservices.

Auf Basis des abstrakten Konzepts wurden ein Interaktionskonzept und die Bedienungsoberfläche für die Tabletop-Steuerung entworfen.

Dabei wurde in Vorüberlegungen ein Lösungsansatz für das Interaktionskonzept entworfen. Der Lösungsansatz enthält sowohl Ansätze für die Benutzeroberfläche, als auch für die Interaktionen mit den verschiedenen Interaktionsmodalitäten des Tabletops. Der Ansatz wird durch existierende Arbeiten und Toolkits für Tabletops inspiriert. Da das Interaktionskonzept möglichst nah an einem Natural User Interface sein soll, wurde daraufhin eine Vorstudie durchgeführt.

Da das Interaktionskonzept nicht in einer normalen Benutzerstudie getestet werden kann wurde Paperprototyping eingesetzt. Bei der Vorstudie konnten Teile der Benutzeroberfläche und die Interaktionen untersucht werden und außerdem neue Ideen gefunden werden. Die Studie ergab, dass es keine einheitlichen Vorlieben für ein Interaktionskonzept gibt. Die Probanden der Studie waren sich bei einigen Gesten und der Anwendung von Tangible-interaktionen einig und sie bestätigten viele Teile der Benutzeroberfläche, wie z.B. die Powerwallansicht und das Filterkonzept. Ein gemeinsames Muster der Vorlieben konnte aber nicht gefunden werden. Manche Probanden schlugen vor verschiedene Interaktionsmodalitäten zu nutzen, so dass jeder Benutzer nach eigenem Belieben arbeiten kann. Viele der Probanden wollten einen starken Aufforderungscharakter für die Interaktionen und Mehrdeutigkeiten bei diesen wurden meistens abgelehnt. Die Ergebnisse der Vorstudie flossen daraufhin in eine Verbesserung des Lösungskonzeptes ein. Bei der Verbesserung wurden einige Ideen, wie das Drag&Drop-Interaktionselement, verworfen und andere verbessert.

Im Anschluss an den Lösungsansatz des Interaktionskonzepts wurde der Prototyp für die Tabletop-Steuerung im Rahmen der Analyseumgebung entwickelt. In diesem Sinne nutzt der Prototyp das Framework für die Analyseumgebung. Der Prototyp selbst wird für den Microsoft Pixelsense entwickelt. Der Pixelsense nutzt ein eigenes Toolkit, genannt Surface SDK 2.0, welches wiederum auf dem Windows Presentation Framework von Microsoft aufbaut. In dem Prototyp werden einige Teile des Lösungsansatzes umgesetzt. Abstriche wurden dabei bei komplexeren Gesten und aufwendigem Feedback gemacht. Die Visualisierungen und die Datengrundlage beruhen auf dem ersten Szenario, das für Eyetracking-Studien entwickelt wurde. Für den Teil der Analyseumgebung, der die Anzeige auf der Powerwall übernimmt, wurde der Prototyp der Diplomarbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ [46] verwendet.

10.1.2 Bewertung

In diesem Kapitel werden einige Aspekte der Arbeit beleuchtet und auf Probleme oder vorteilhafte Erkenntnisse untersucht.

Anforderungsanalyse

Mit der Anforderungsanalyse konnten die Ziele und die grundsätzlichen Vorgehensweise bei einer solchen Auswertung verstanden werden. Die Schwierigkeit, ein Szenario für die Analyseumgebung zu entwickeln, war das Fehlen von guten Werkzeugen und einheitlicher Vorgehensweise für die Auswertung.

Konzept und Framework für die Analyseumgebung

Das Konzept und das Framework für die Analyseumgebung wurden in Zusammenarbeit mit der Diplomarbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ [46] ausgearbeitet. Die Zusammenarbeit beim Konzept lieferte schnell Ergebnisse, die von beiden Seiten als gut empfunden wurden. Kernpunkte waren dabei die Abstraktion und die Erweiterbarkeit des Frameworks. Daher wurde, unter gemeinsamem Einverständnis, der Einsatz von Webservices beschlossen und eine Plugin-Architektur für die Visualisierungen entworfen. So können einfach neue Geräte in das System eingebunden werden und neue Visualisierungen entwickelt werden.

Durch Verteilung der Aufgaben und Einsatz von „Extreme Programming“ bei schwierigeren Teilen konnte das Framework schnell implementiert werden.

Lösungsansatz des Interaktionskonzepts

Die Entwicklung des Lösungsansatzes des Interaktionskonzeptes stellte eine Herausforderung dar, da es wenig Erfahrung auf dem Gebiet der Tabletop-Anwendungen und Analyseumgebungen mit Powerwalls gibt. Auch wenn Arbeiten in diese Richtung existieren, beschränken sich diese meist auf Erforschung der Grundlagen und Verbesserung der Technologien. Auch Anwendungen, die auf dem Surface SDK beruhen, sind meist auf speziellere Anwendungsgebiete beschränkt oder kommen aus dem Unterhaltungsbereich. Daher wurde nach einigen Entwürfen entschieden die Vorstudie durchzuführen.

Die Studie lieferte kein vollkommen einheitliches und rundes Konzept. Lediglich das Filterkonzept war allgemein sehr beliebt

Vorstudie

Die Vorstudie gab einen guten Einblick in die Wünsche und Erwartungen von Benutzern. Es konnten einige Stellen im Lösungskonzept erkannt werden, die dem Benutzer nicht klar waren oder unintuitiv sind. Außerdem konnten in der Absprache mit Probanden neue Ideen entwickelt werden. Das Paperprototyping ermöglichte schnelles Testen von Konzepten bei Interaktionen und Benutzeroberflächen. Ebenso können relativ schnell Anpassungen vorgenommen werden und diese wieder geprüft werden. Die Kombination mit „User Defined Design“ [48] macht die Studie noch flexibler und bringt schnell Inspiration für den Entwurf der Benutzeroberfläche und deren Interaktionen.

Der Einsatz von Probanden aus dem Bereich der Informatik hat zwar viele Ideen von Experten geliefert. Allerdings konnten so keine Meinungen von unerfahrenen Benutzern eingeholt werden, was den Einsatz für Laien erschweren könnte. Die Studie liefert, so wie sie in dieser Arbeit durchgeführt wurde, nur qualitative Ergebnisse. Um quantitative Ergebnisse zu bekommen müsste ein Fragebogen entworfen werden, mit dem jeder Proband eine Bewertung abgeben kann.

Microsoft Pixelsense
Der Microsoft Pixelsense ist ein sehr kompakter Tabletop und kann daher für viele Szenarien eingesetzt werden. Allerdings ist die Pixelsense-Technik noch nicht ausgereift genug. Infrarote Umgebungsstrahlung stört die Erkennung von Berührungen und Tangibles stark und kann unter Umständen die Eingabe komplett unmöglich machen. Daher muss genau auf die Umgebungsbeleuchtung geachtet werden.

Das Surface SDK 2.0 erlaubt es schnell einfache Anwendungen zu erstellen. Es bietet grundlegende Unterstützung für Gesten und Tangibleinteraktion. Wenn allerdings komplexere Aufgaben zu lösen sind, stößt das SDK gelegentlich an Grenzen.

10.2 Ausblick

Dieses Kapitel gibt einen Ausblick über Ideen die in dieser Arbeit nicht, oder nur geringfügig, bearbeitet werden konnten. Außerdem werden Vorschläge genannt, wie technische Lösungen die Arbeit verbessern könnten.

10.2.1 Benutzerstudie

Um die tatsächliche Nutzbarkeit des Prototyps zu testen, sollte eine Benutzerstudie für diesen durchgeführt werden. Um in dieser ersten Studie den Prototypen des Tabletops zu evaluieren, kann für die Powerwall ein Client eingesetzt werden, der keine Interaktion zulässt. Der Versuchsaufbau besteht aus der Powerwall mit einem Steuerrechner, der diesen Client ausführt und einem Microsoft Pixelsense, der den Prototypen ausführt. Die Aufgaben können aus den implementierten Teilen des Lösungsansatzes entworfen werden. Dabei werden die einzelnen Interaktionsmodalitäten (Tangibles, Gesten) getrennt, damit ein Vergleich dieser für die einzelnen Aufgaben durchgeführt werden kann. Die Durchführung der Studie beginnt mit einer Einführung in das System und eine grobe Vorstellung der Aufgaben. Bei der Ausführung der Aufgaben wird die Aufgabe zuerst erklärt. Danach wird sie mit den einzelnen Interaktionsmodalitäten vorgeführt. Darauf wird der Proband gebeten die Aufgabe auszuführen. Dabei soll er zwar mit der Interaktion anfangen, die ihm am besten gefällt, aber alle einmal ausprobieren. Für Aufgaben, die für Mehrbenutzerbetrieb relevant sind, können ein oder mehrere Statisten den Probanden unterstützen. Wenn der Proband die Interaktionen ausprobiert hat, soll er einen Fragebogen zum Vergleich der Interaktionsmodalitäten für die spezielle Aufgabe ausfüllen. Am Ende der Studie soll der Proband einen Fragebogen ausfüllen wie er die Aufgabe und ihre Interaktionen vom Schwierigkeitsgrad bewertet. Aus den Ergebnissen kann der Prototyp des Tabletop-Clients und dessen Interaktionen weiter verbessert werden.

Da die Arbeit aber auch die multimodale Analyseumgebung betrifft, sollte auch eine weitere Studie durchgeführt werden. In dieser Studie sollte ein Szenario evaluiert werden, bei dem verschiedene Geräte eingesetzt werden. Der erste Schritt dafür wäre der gleichzeitige Einsatz des Tabletop-Clients und des Powerwall-Clients mit Freihandgestensteuerung aus der Arbeit „Gestensteuerung für Powerwall-basierte Visualisierungen“ [46]. Dies ermöglicht die Einsatzfähigkeit der Analyseumgebung mit mehreren Geräten zu evaluieren. Weiterhin gibt es aber auch Aufschluss darauf, wie gut die Interaktionskonzepte zusammen funktionieren und an welchen Stellen sie sich ergänzen. Die Ergebnisse der Studie können schlussendlich zur Verbesserung der Interaktionskonzepte in Bezug auf Kompatibilität verwendet werden. Außerdem ermöglichen die Ergebnisse das Framework der Analyseumgebung zu erweitern.

10.2.2 Technische Lösungen

Dieser Abschnitt gibt einen Ausblick welche technischen Möglichkeiten die Bedienung des Tabletop-Client verbessern könnten.

Tabletop
Der Microsoft Pixelsense hat Probleme mit externem Infrarotlicht, was wahrscheinlich auf die Empfindlichkeit des Pixelsense-Verfahrens rückzuführen ist. Die Probleme können durch Wahl der richtigen Beleuchtung verringert werden. Allerdings ist das nicht immer möglich, wenn die Räumlichkeiten Fenster haben, da Sonnenlicht ebenfalls einen starken Infrarotteil enthält. Eine Verbesserung der Technologie durch Microsoft wäre eine Lösung dieses Umstands. Eine andere Lösung wäre der Einsatz eines Tabletops, der auf DI (siehe Kapitel 2.3) basiert. Diese Technik ist nicht ganz so anfällig für externes Infrarotlicht. Die Tabletops sind leider schwer und sperrig, allerdings könnte man solch einen Tabletop fest vor einer Powerwall einbauen, da diese ebenfalls nicht mobil ist.

Tangibles
Im Gegensatz zu den passiven Tangibles, die keine eigenes Feedback, Anzeige oder Interaktionen erlauben, könnten ebenso elektronisch augmentierte Tangibles eingesetzt werden. Diese Tangibles könnten über drahtlose Kommunikationsmittel mit ihrem Host verbunden sein. Dies ermöglicht es, über Tangibles selbst Informationen auszugeben oder die Interaktionsmöglichkeiten zu erweitern. In diesem Zusammenhang wären folgende Erweiterungen denkbar.

- Display
  Wenn das Tangible mit einem Display ausgestattet ist, kann es den aktuellen Status anzeigen. Bei Datenrepräsentationstangibles können außerdem Informationen über die repräsentierten Daten angezeigt werden, ohne dass das Tangible auf dem Tabletop steht. Das wiederum erhöht den Nutzen des Tangibles als Ablage erheblich, da der Benutzer sich nicht merken muss welche Daten im Tangible gespeichert sind.

- Interaktionselemente
  Die Interaktionsmöglichkeiten des Tangibles können durch echte Steuerelemente auf dem Tangible, wie elektronischen Knöpfen oder Reglern, erweitert werden. Das erweitert die Tangibles um neue Interaktionen, die auch haptisches Feedback mitbringen. Des Weiteren können Interaktionen ausgeführt werden, ohne dass das Tangible sich auf dem Tabletop befindet. So können z.B. Daten entfernt werden, die einem Tangible zugeordnet sind oder der Status des Tangibles zurückgesetzt werden. In diesem Fall sollte die Erweiterung aber mit dem Display kombiniert werden sonst erhält der Benutzer zu wenig Feedback.

- Annäherungserkennung
  Eine Annäherungserkennung des Tangibles an den Tabletop würde es erlauben Feedback zu der Funktionalität des Tangibles zu geben, bevor irgendwelche Aktionen ausgeführt werden. Dies kann analog zu der Hover-Eigenschaft der Maussteuerung genutzt werden. Auf diese Weise kann eine Vorschau von Aktionen angezeigt werden.

Eine anderer Art von Tangible, die auf Tabletops eingesetzt werden könnten, sind die „Silicone iLluminated Active Peripherals(SLAP) Widgets“ [54]. Diese Art von Tangibles sind Widgets, die als reale Objekte auf dem Tabletop positioniert werden können. Die Tangibles werden aus Silikon hergestellt und sind halbtransparent, können also vom Display beleuchtet werden. Einfache Beispiele sind Knöpfe, Drehschalter oder Schieberegler. Diese Widgets liefern im Gegensatz zu ihren digitalen Gegenstücken ein haptisches Feedback. Außerdem zeigen ihre physikalischen Objekte ihren Status an, ohne dass das Display des Tabletops zum Einsatz kommen muss.

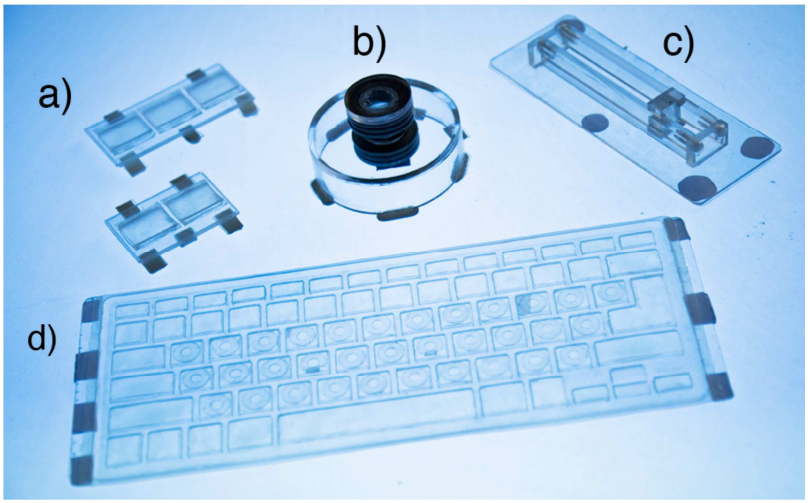


Abbildung 101: Eine Auswahl an SLAP Widgets
a) Radiobuttons, b) Drehknopf, c) Regler, d) Tastatur

10.2.3 Implementierung

Erweiterung für die Implementierung des Frameworks und des Tabletops sind möglich und werden im Weiteren beschrieben.

Framework der Analyseumgebung

Das Framework bietet momentan nur die Möglichkeit, Daten zu laden. Sollen Veränderungen oder Notizen gespeichert werden, muss das Framework angepasst werden. Das ist allerdings nicht ganz so einfach. Das Framework ist momentan so ausgelegt, dass Clients Daten auf Bedarf nachladen. Die Clients synchronisieren sich, indem sie kurz gesagt weitergeben welche Daten sie verwenden. Um aber Daten auch verändern zu können, muss auch aktiv synchronisiert werden, denn sonst wissen die Clients nicht, dass Daten verändert wurden. Die einfachste Anpassung des Frameworks, um dies zu ermöglichen, ist eine Erweiterung der passiven Synchronisierung. Bei dieser Erweiterung werden die Daten in der Datenhaltung der Analyseumgebung gespeichert. Danach wird eine Nachricht an die anderen Clients gesendet, dass sich gewisse Daten geändert haben. Daraufhin können sich die Clients die geänderten Daten nachladen.

Das Framework erlaubt momentan nur einen öffentlichen Bereich. Ein Austausch zwischen privaten Bereichen verschiedener Clients ist nicht möglich. Das Framework unterstützt den Austausch von Daten zwischen Clients. Daher wäre ein Austausch zwischen privaten Bereichen durch eine Einführung von neuen Nachrichten möglich. Der Einsatz eines Austauschs zwischen zwei Clients ist vielfältig. Es könnten Tangibles ähnlich der MediaBlocks [25] zum Austausch von Daten zwischen Clients verwendet werden. So können sich Tabletops Tangibles und ihre Daten teilen. Eine andere Anwendung wäre eine Verschmelzung der privaten Bereiche zweier oder mehrerer Clients. Dies ermöglicht eine kombinierte Nutzung mehrerer Tabletops.

Clients für persönliche Geräte

Es können Clients der Analyseumgebung für persönliche Geräte entwickelt werden. Die persönlichen Geräte sollten kompakt und einfach zu bedienen sein. Wenn für die persönlichen Geräte Tablets und Smartphones verwendet werden, können die Konzepte des Tabletops verwendet werden und für den Einzelnutzerbetrieb rückoptimiert werden. Das Tablet braucht keine Oberfläche mit frei rotierbaren Fenstern. In Smartphones kann auf Grund ihrer Größe nur eine Visualisierung angezeigt werden. Die Powerwallanzeige ist in ihrem abstrakten Konzept genauso auf Tablets oder Smartphones einsetzbar. Auf Grund der geringen Größe

der Geräte müssen aber sinnvolle Vorkehrungen getroffen werden, dass die Powerwallanzeige sinnvoll bedient werden kann. Ebenso können die entworfenen Gesten auch auf solchen Geräten verwendet werden.

Tabletop-Client
Da zur Zeit der Implementierung keine Clients für persönliche Geräte existiert haben, konnten auch keine Verbindungsmöglichkeit des Tabletop-Clients mit diesen Geräten umgesetzt werden. Im Lösungsansatz wird vorgeschlagen eine Erweiterung der privaten Ablagen vorzunehmen. Die Erweiterung sollte den privaten Raum des Geräts anzeigen. Doch sehen die privaten Räume der verschiedenen Geräte unterschiedlich aus und werden unterschiedlich bedient. Daher wäre es sinnvoll für jedes Gerät eine eigene Darstellung zu entwickeln. Da das Framework jedoch offen für neue Geräte ist, sollte eine einheitliche Schnittstelle verwendet werden. Diese sollte es erlauben die Darstellung für eine Verbindung zu einem neuen Gerät zum Client hinzuzufügen, ohne dass dessen Quellcode angepasst werden muss. Dies kann wiederum über ein Plugin-System gelöst werden, wie es auch bei den Visualisierungen der Fall ist. Für den Fall, dass es kein Plugin für ein Gerät gibt sollte es eine Standarddarstellung geben.

Die Erkennung der Tangibles ist beim Ziehen oder bei schlechten Lichtbedingungen etwas instabil, daher sollten die Interaktionen so angepasst werden, dass ein kurzes Verlieren des Tangibles keine Auswirkung auf die Funktion hat. Zusätzlich könnte man Tags mit höherer Qualität nehmen, die die Erkennung erleichtern.

Ebenfalls könnten noch weiter Funktionen für die Hilfestellung eingeführt werden. Für die Visualisierungscontainer könnte eine Kurzhilfe entwickelt werden, die jedes Steuerelement des Containers mit einem Popup kurz beschreibt. Dadurch können völlige Neueinsteiger einen kurzen Überblick über die Funktionen erhalten

Visualisierungen
Im Prototyp werden einige Visualisierungen für Eyetracking-Auswertung entwickelt. Neben der im Prototyp entwickelten Visualisierung existieren noch weitere Parallel-Scan-Path – Visualisierungen, wie in Kapitel 5.1.2 vorgestellt. Diese können ebenfalls als Plugins für die Analyseumgebung entwickelt werden.

Literaturverzeichnis

[1] EMC, „EMC Digital Universe,“ EMC, [Online]. Available: http://germany.emc.com/leadership/programs/digital-universe.htm. [Zugriff am 19. August 2012].

[2] D. Keim, „Information visualization and visual data mining,“ *IEEE Transactions on Visualization and Computer Graphics*, Bd. 8, Nr. 1, pp. 1-8, 2002.

[3] S. Grottel, „MegaMol Projektseite,“ Universität Stuttgart VISUS, [Online]. Available: https://svn.vis.uni-stuttgart.de/trac/megamol/. [Zugriff am 19. August 2012].

[4] R. Fraedrich, J. Schneider und R. Westermann, „Exploring the millennium run--scalable rendering of large-scale cosmological datasets.,“ *IEEE transactions on visualization and computer graphics*, pp. 1251-1258, Novermber 2009.

[5] G. Mark, A. Kobsa und V. Gonzalez, „Do four eyes see better than two? Collaborative versus individual discovery in data visualization systems,“ *Proceedings Sixth International Conference on Information Visualisation*, pp. 249-255, 2002.

[6] G. Schmidt, O. Staadt, M. Livingston, R. Ball und R. May, „A Survey of Large High-Resolution Display Technologies, Techniques, and Applications,“ in *IEEE Virtual Reality Conference (VR 2006)*, 2006. pp. 223-236., DOI 10.1109/VR.2006.20

[7] P. Dietz und D. Leigh, „DiamondTouch: a multi-user touch technology,“ in *Proceedings of the 14th annual ACM symposium on User interface software and technology - UIST '01*, New York, 2001. pp. 219-226., DOI 10.1145/502348.502389

[8] Microsoft, „Microsoft Pixelsense,“ Microsoft, [Online]. Available: www.pixelsense.com. [Zugriff am 19. August 2012].

[9] Microsoft, „Harrah’s Entertainment Launches Microsoft Surface at Rio iBar, Providing Guests With Innovative and Immersive New Entertainment Experiences,“ Microsoft, [Online]. Available: http://www.microsoft.com/en-us/news/press/2008/jun08/06-11hetsurfacepr.aspx. [Zugriff am 19. August 2012].

[10] Microsoft, „AT&T First to Introduce Microsoft Surface in Retail Stores to Enhance Mobile Shopping Experience,“ Microsoft, [Online]. Available: http://www.microsoft.com/en-us/news/press/2008/apr08/04-01surfaceretailpr.aspx. [Zugriff am 19. August 2012].

[11] M. Weiser, „The computer for the 21 st century,“ *ACM SIGMOBILE Mobile Computing and Communications Review*, pp. 3-11, 1999.

[12] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas und H. Ziegler, „Visual Analytics: Scope and Challenges,“ University of Konstanz.

[13] J. J. Thomas und K. A. Cook, Illuminating the Path, IEEE.

[14] D. Keim, J. Kohlhammer, G. Ellis und F. Mansmann, Mastering the Information Age Solving Problems with Visual Analytics, Eurographics Association, 2010. , ISBN 978-3-905673-77-7

[15] C. R. Johnson und C. D. Hansen, The Visualization Handbook, Elsevier, 2005. , ISBN 978-0123875822

[16] H. Schumann und W. Müller, Visualisierung, Grundlagen und allgemeine Methoden, Berlin: Springer-Verlag, 2000. , ISBN 978-3-540-64944-1

[17] H. Bosch, D. Thom, M. Worner, S. Koch, E. Puttmann, D. Jackle und T. Ertl, „ScatterBlogs: Geo-spatial document analysis," in *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2011. pp. 309-310., DOI 10.1109/VAST.2011.6102488

[18] B. Shneiderman, „The Eyes have it: A task by data type taxonomy for information visualization.," *IEEE Symposium on Visual Languages*, p. 336–343, 1996.

[19] H. D. Hellige, Mensch-Computer-Interface: Zur Geschichte und Zukunft der Computerbedienung, transcript Verlag, 2008.

[20] V. Bush, „As we may think," *ACM SIGPC Notes*, Bd. 1, Nr. 4, pp. 36-44, 1979.

[21] I. E. Sutherland, „Sketchpad: a man-machine graphical communication system," in *Proceedings of the May 21-23, 1963, spring joint computer conference on - AFIPS '63 (Spring)*, 1963. pp. 329-346., DOI 10.1145/1461551.1461591

[22] R. J. Jacob, A. Girouard, L. M. Hirshfield, M. S. Horn, O. Shaer, E. T. Solovey und J. Zigelbaum, „Reality-based interaction: a framework for post-WIMP interfaces," in *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, 2008. pp. 201-210., DOI 10.1145/1357054.1357089

[23] D. Wigdor und D. Wixon, Brave Nui World, Burlington, MA: Elsevier Inc, 2010. , ISBN 978-0-12-382231-4

[24] H. Ishii und B. Ullmer, „Tangible bits: towards seamless interfaces between people, bits and atoms," in *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, 1997. , DOI 10.1145/258549.258715

[25] B. Ullmer, H. Ishii und D. Glas, „mediaBlocks: Physical Containers, Transports, and Controls for Online Media," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, 1998. pp. 379-386., DOI 10.1145/280814.280940

[26] S. Jordà, G. Geiger, M. Alonso und M. Kaltenbrunner, „The reacTable : Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces," in *Proceedings of the 1st international conference on Tangible and embedded interaction -*

TEI '07, 2007. pp. 139 - 146., DOI 10.1145/1226969.1226998

[27] R. Chang, F. Wang und P. You, „A Survey on the Development of Multi-touch Technology," in *2010 Asia-Pacific Conference on Wearable Computing Systems*, 2010. pp. 363-366., DOI 10.1109/APWCS.2010.99

[28] P. Brandl, F. Daiber und F. Echtler, „Multi-touch surfaces: A technical guide," *Interactive Surfaces*, 2008.

[29] J. Y. Han, „Low-cost multi-touch sensing through frustrated total internal reflection," in *UIST*, Seattle, WA, 2005.

[30] J. Y. Han, „Multi-touch interaction wall," in *SIGGRAPH*, Boston, Massachusetts, 2006.

[31] Microsoft, „MSDN - Surface SDK SP1," Microsoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/ee804767(v=surface.10).aspx. [Zugriff am 19. August 2012].

[32] Microsoft, „MSDN - Surface SDK 2," Microsoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/ff727815. [Zugriff am 19. August 2012].

[33] S. Jordà, „The Reactable: Tangible and Tabletop Music Performance," in *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA '10*, 2010. pp. 2989-2994., DOI 10.1145/1753846.1753903

[34] SAGE, „SAGE :: GALLERY," SAGE, [Online]. Available: http://www.evl.uic.edu/cavern/sage/gallery.php. [Zugriff am 19. August 2012].

[35] VISUS, „Technischer Aufbau - VISUS," VISUS, Uni Stuttgart, [Online]. Available: http://www.visus.uni-stuttgart.de/institut/visualisierungslabor/technischer-aufbau.html. [Zugriff am 19. August 2012].

[36] D. J. Sandin, T. Margolis, J. Ge, J. Girado, T. Peterka und T. A. DeFanti, „The Varrier autostereoscopic virtual reality display," in *ACM SIGGRAPH 2005 Papers on - SIGGRAPH '05*, 2005. pp. 894 - 903., DOI 10.1145/1186822.1073279

[37] W. A. König, R. Rädle und H. Reiterer, „Squidy: a zoomable design environment for natural user interfaces," in *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems - CHI EA '09*, 2009. pp. 4561-4566., DOI 10.1145/1520340.1520700

[38] T. Gjerlufsen, C. Klokmose, J. Eagan, C. Pillias und M. Beaudouin-Lafon, „Shared substance: developing flexible multi-surface applications," in *Proceedings of the 2011 annual conference on Human factors in computing systems*, 2011. p. 3383–3392., DOI 10.1145/1978942.1979446

[39] H.-c. Jetter, W. A. König, J. Gerken und H. Reiterer, „ZOIL – A Cross-Platform User Interface Paradigm for Personal Information Management," in *Personal Information Management: PIM 2008, CHI 2008 Workshop*, 2008. pp. 1-9.

[40] A. Bragdon, R. DeLine, K. Hinckley und M. R. Morris, „Code Space: Touch + Air Gesture Hybrid Interactions for Supporting Developer Meetings," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces - ITS '11*, 2011. pp. 212-221., DOI 10.1145/2076354.2076393

[41] F. IOSB, „SmartControlRoom," Fraunhofer IOSB, [Online]. Available: http://www.iosb.fraunhofer.de/servlet/is/6620/. [Zugriff am 19. August 2012].

[42] N. Streitz, T. Prante, C. Müller-Tomfelde, P. Tandler und C. Magerkurth, „Roomware: the second generation," in *CHI '02 extended abstracts on Human factors in computing systems - CHI '02*, 2002. pp. 506-507., DOI 10.1145/506443.506452

[43] O. Shaer, M. Strait, C. Valdes, T. Feng, M. Lintz und H. Wang, „Enhancing genomic learning through tabletop interaction," in *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, 2011. pp. 2817-2826., DOI 10.1145/1978942.1979361

[44] B. Schneider, M. Strait, L. Muller, S. Elfenbein, O. Shaer und C. Shen, „Phylo-Genie: engaging students in collaborative 'tree-thinking' through tabletop techniques," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, 2012. pp. 3071-3080., DOI 10.1145/2207676.2208720

[45] X. Chen, „Visuelle Analyse von Eye-Tracking-Daten," 2011.

[46] N. Ploner, „Gestensteuerung für Powerwall-basierte Visualisierungen," 2012.

[47] C. Snyder, Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces, San Francisco: Elsevier, 2003. , ISBN 978-1-55860-870-2

[48] J. O. Wobbrock, M. R. Morris und A. D. Wilson, „User-defined gestures for surface computing," in *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09*, 2009. p. 1083ff., DOI 10.1145/1518701.1518866

[49] P. Dragicevic und Y. Shi, „Visualizing and manipulating automatic document orientation methods using vector fields," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces - ITS '09*, 2009. pp. 65-68., DOI 10.1145/1731903.1731918

[50] S. Scott, M. Sheelagh, T. Carpendale und K. Inkpen, „Territoriality in collaborative tabletop workspaces," in *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 2004. p. 294–303., DOI 10.1145/1031607.1031655

[51] D. A. Norman, The Design of Everyday Things, Perseus Books, 2002. , ISBN 978-0465067107

[52] D. Saffer, Designing Gestural Interfaces, Sebastopol: O'Reilly, 2008. , ISBN 978-0-596-51839-4

[53] Microsoft, „MSDN - ADO.NET Entity Framework," [Online]. Available:

http://msdn.microsoft.com/de-de/library/bb399572. [Zugriff am 19. August 2012].

[54] M. Weiss, J. Wagner, Y. Jansen, R. Jennings, R. Khoshabeh, J. D. Hollan und J. Borchers, „SLAP Widgets: Bridging the Gap Between Virtual and Physical Controls on Tabletops,“ in *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09*, 2009. pp. 481-490., DOI 10.1145/1518701.1518779

Anhang A: 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 einen 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?		

Haben sie schon einen Tabletop-PC (Microsoft Surface, …) benutzt?	Ja	Nein
Haben sie in diesem Zusammenhang Interaktionen mit Realweltobjekten durchgeführt?	Ja	Nein

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben. Wörtliche und sinngemäße Übernahmen aus anderen Quellen habe ich nach bestem Wissen und Gewissen als solche kenntlich gemacht.

Stuttgart, den 21. August 2012 _____