Fachbereich Informatik und Medien

BACHELORARBEIT

Benutzung von gestenbasierter Interaktion und Multi-Touch zur Informationsdarstellung in einer prototypischen Applikation im öffentlichen Umfeld

Vorgelegt von: André Diemar

am: 13.04.2011

zum

Erlangen des akademischen Grades

BACHELOR OF SCIENCE

(B.Sc.)

Erstbetreuer: Prof. Stefan Kim

Zweitbetreuer: Prof. Dr. rer. nat. Kai von Luck

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit zum Thema

Benutzung von gestenbasierter Interaktion und Multi-Touch zur Informationsdarstellung in einer prototypischen Applikation im öffentlichen Umfeld

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Brandenburg/Havel, den 13.04.2011

Unterschrift

Danksagung

Ein großer Dank richtet sich an Herrn Prof. Dr. rer. nat. Kai von Luck und Herrn Prof. Stefan Kim für die geduldige Betreuung der Bachelorarbeit. Sie halfen mit einem offenen Ohr für Fragen und Probleme sowie bereichernden Anregungen während der Bearbeitungszeit.

Ein weiterer Dank richtet sich an den Fachbereich für Informatik und Medien der FH Brandenburg sowie an den Fachbereich für Technik und Informatik der HAW Hamburg für die Unterstützung und Förderung dieser Bachelorarbeit.

Des Weiteren bedanke ich mich bei meiner Familie für das Berichtigen sprachlicher und logischer Ungereimtheiten. Ohne ihre Unterstützung wäre die Vollendung dieser Bachelorarbeit nicht möglich gewesen.

Zusammenfassung

Neue Interaktionskonzepte werden benötigt, um Informationen durch die immer ausgefeilteren Technologien sinnvoll für den Benutzer zur Verfügung zu stellen. In dieser Bachelorarbeit wird ein Umriss über unterschiedliche Konzepte zur Nutzung von Gesten geschaffen sowie auf deren Umsetzbarkeit eingegangen.
Ein weiterer Abschnitt wird sich in einem eigenen Szenario mit einer möglichen Umsetzung beschäftigen. Dabei liegt der Schwerpunkt zur Informationsdarstellung im öffentlichen Bereich.

Abstract

Their is a need of new concepts for interaction to represent information to the user because of more sophisticated technologies. The Bachelor Thesis gives a sketch of different gestural concepts, their usings and implementations.

Another section consists of an own examplary scenario and it's possible implementation. The focus of the implementation is the representation of information in the public environment.

Inhaltsverzeichnis

1 Einleitung

" Remember that it took 30 years between when the mouse was invented by Engelbart and English in 1965 to when it became ubiquitous. "

Bill Buxton

In immer neuen Bereichen des Menschen tauchen Computerbasierte Systeme auf. Sie ermöglichen den sich beschleunigenden Zugang zu Informationen und deren Verarbeitung. Durch sie wird eine einfachere und schnellere Kommunikation erreicht. Intelligenter und selbstständiger verrichten sie ihre Arbeit, und sie sollen auch schneller und effektiver durch jedermann bedienbar sein. Somit müssen ständig angepasste Interaktionskonzepte erstellt werden.

Als gängige und sich etablierte Kommunikationsschnittstelle zwischen Mensch und Maschine wurde die Maus und die Tastatur geschaffen. Auch wurden andere Ideen, wie z.B. Sprachenerkennung, Augmented Reality [1], Gestenerkennung und Touchscreens entwickelt. In welchen Bereichen sich andere Konzepte nutzbringend etablieren werden, ist ein langwieriger Prozess. Doch in dieser Zeit können interessante Umsetzungen durch jedermann geschaffen werden.

Besonders im Bereich der Nutzung von Multi-Touch-basierten Systemen sind noch keine endgültigen Begrenzungen definiert, und es gibt keinerlei unumstößliche Regeln.

1.1 Motivation

Es wird immer wichtiger, die uns zur Verfügung stehenden Computerbasierten Systeme mit einfacher Bedienung bereitzustellen. Dabei sollte man keine Handbücher auswendig kennen, noch sollte z.B. der spielerische Faktor darunter leiden. Die Nutzung von Multi-Touch-basierten Systemen und geeigneter Gesten kann zu einer möglichen Verbesserung der üblich

[1] zu deutsch: Erweiterte Realität

genutzten Kommunikationshilfsmitteln führen. Neue Ideen zur möglichen Umsetzung werden schon seit Jahren getätigt und führen zu interessanten Denkansätzen. Mit der Zeit wird sich klären, welche in Zukunft möglicherweise weiter genutzt werden oder auch verbessert vorzufinden sind.

Gerade im öffentlichen Umfeld stellen sich andere Anforderungen an ein Computerbasiertes System, als beispielsweise im Arbeitsumfeld. So verschieben sich die Prioritäten der benötigten Anforderungen. Inwieweit sich die bestehenden Umsetzungen auch in Zukunft behaupten können, ob es interessantere Neuerungen geben wird oder eine Kombination mit anderen Kommunikationsformen möglich ist, wird sich in der nächsten Zeit zeigen.

Eine wichtige Inspiration dieser Bachelorarbeit ist es, eine Basis zukünftiger Herangehensweisen und deren sinnvolle Umsetzung als Interaktionskonzept zur Bedienung von Computerbasierten System im öffentlichen Umfeld zu schaffen. Ein Fokus auf einer möglichen Adaption von bestehenden Interaktionskonzepten sowie Szenarien ist ebenfalls inbegriffen.

1.2 Gliederung

Die theoretischen Grundlagen bezüglich Multi-Touch-basierter Eingabe beginnen mit dem 2. Kapitel. Eine Darstellung der Verknüpfungen zu anderen Fachbereichen in der Informatik wird aufgezeigt. Weiterführend wird sich mit einer Möglichkeit beschäftigt, eine Beschreibung von Interaktion mit Hilfe eines Vokabulars zu erstellen. Dazu dienen auch die anschließend vorgestellten Modelle. Besondere Aufmerksamkeit wird der Vermittlung von
”Freude und Spaß” eines Produktes gewidmet sowie den unterschiedlichen Techniken von Multi-Touch-basierten Gesten. Zum Schluss des Kapitels wird ein kurzer geschichtlicher Überblick und ein Überblick über interaktive Oberflächen gegeben.

Welche Software für den praktischen Anteil dieser Bachelorarbeit genutzt wird, wird in Kapitel 3 behandelt.

Kapitel 4 beinhaltet mehrere Punkte, die sich inhaltlich mit der Interaktion auf Basis von Multi-Touch-Gesten beschäftigen. Dazu werden die einzelnen Eigenschaften und deren Bedeutung erklärt. Welche Abhängigkeiten und zusammenfassbare Gestenformen möglich sind, bilden den Abschluss des Kapitels.

Nachdem die theoretischen Grundlagen zu Interaktion und Gesten getätigt wurden, wird mit dem 5. Kapitel eine Umsetzung geschaffen. Dazu wird das Konzept für die praktische Umsetzung anhand eines Szenarios erklärt. Die Gestendokumentation nimmt dabei einen wichtigen Anteil ein.

Im 6. Kapitel wird die praktische Umsetzung behandelt. Dazu wird ein Überblick über die Systemarchitektur geschaffen. Zur Datenhaltung wurden XML-Dokumente ausgewählt, weshalb sich ein Abschnitt mit dem Parsen von XML-Dokumenten beschäftigt. Am Ende wird die Nutzung der Entwicklungsumgebung und die Erklärung der einzelnen Elemente sowie deren Benutzung erörtert.

Im Anschluss wird ein Überblick über die Multi-Touch-Technik und deren Potential in der heutigen Zeit gegeben. Es geht auch um mögliche Symbiosen mit anderen Technologien und Interaktionskonzepten im Hinblick auf die zukünftige Nutzung.

1.3 Zielsetzung

Der Fokus dieser Arbeit unterteilt sich in zwei grobe Gebiete. Es wird darauf eingegangen, welche theoretischen Überlegungen getätigt werden sollten, um ein gestenbasiertes Interaktionskonzept realisieren zu können. Die Implementierung einer prototypischen Applikation versucht die vorher getätigten Überlegungen geeignet umzusetzen. Dies betrifft einerseits den Funktionsumfang, andererseits auch die visuelle Umsetzung der Applikation.

Der Aufbau der Applikation, welcher durch den Designprozess vorgegeben wird, soll dabei nicht unumstößlich sein. Vielmehr soll eine Erweiterbarkeit umsetzbar sein, besonders im Hinblick auf die Wiedergabe von Inhalten. Die Bedienung sollte auf einer einfachen Basis gehalten werden und dem Benutzer wenig Schwierigkeiten bereiten. Da sich das Einsatzgebiet im öffentlichen Umfeld befindet, muss die Einschränkung durch Hardware, Software und zukünftigen Anwendern so gering wie möglich gehalten werden.

2 Grundlagen

2.1 Interaktionsdesign

Bei der Fachrichtung Interaktionsdesign wird nicht das Erscheinungsbild, sondern das Verhalten des Produktes bei seiner Benutzung verstanden [Saf09]. Dabei ist die Interaktion für den Informationsaustausch zuständig, kann aber weiterführend auch in Form eines Services oder einer Ware auftreten. Sie findet in verschiedenen Varianten zwischen Menschen, Systemen und Computerbasierten Systemen statt. Diese aufgezeigten Lösungen sind weder als Musterlösung, noch für die Ewigkeit gedacht. Vielmehr ist es für die gegenwärtige Zeit und Problemstellung ein erstelltes Produkt, welches die verfügbaren Ausgangsmaterialien nutzt. Allgemein wird diese Fachrichtung als Kunst und weniger als Wissenschaft betrachtet. Dies liegt unter anderem daran, dass schwer allgemeingültige Regeln festgelegt werden können.

Es ist eine noch junge Fachrichtung, dessen Einordnung zu anderen bestehenden nicht gänzlich abgeschlossen ist. Durch die vielen Überlappungen der einzelnen Disziplinen untereinander, zeigt sich deutlich, dass keine nur für sich alleine steht (siehe Abbildung 2.1). Jedes gute Produkt heutiger Zeit ist durch ein Zusammenspiel mehrerer Disziplinen entstanden, welches mit dem Oberbegriff User Experience Design definiert wird. Auch zeigt sich deutlich, dass in einigen Disziplinen der Erfindungs- bzw. Herstellungsprozess inbegriffen ist, weshalb diese auch teilweise außerhalb des User Experience Design einzuordnen sind.

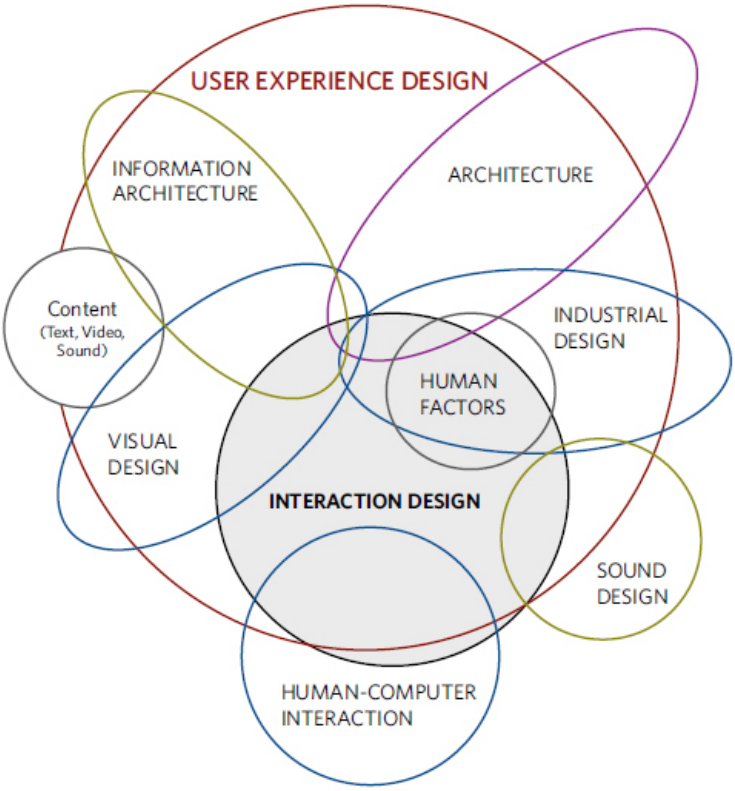

Abbildung 2.1: Einordnung des Interaktionsdesigns zu anderen Fachrichtungen

Somit ist ein Endprodukt das Zusammenspiel vieler Spezialisten aus unterschiedlichen Disziplinen und nicht nur das Werk einer Expertengruppe.

Hauptrichtungen des Interaktionsdesigns

Mit der Zeit entwickelten sich in den letzten Jahren unterschiedliche Richtungen des Interaktionsdesigns. Eine Vielzahl unterschiedlicher Schwerpunkte wird genutzt, worunter das Bekannte *user-centered desgin*[1] genannt werden kann.

Technology-Centered View

Die durch Ingenieure und Programmierer hergestellten Ausgangsprodukte[2] werden durch Interaktionsdesignern zu Endprodukten[3] komplettiert. Erst durch die Fertigstellung der Interaktionsdesigner kann das Produkt richtig benutzt werden.

[1] zu deutsch: Nutzungsorientiertes Design, auf Benutzer zugeschnitten
[2] auch digital: Software, Internet
[3] useful, usable, pleasureable

Behaviorist View

Durch Robert M. Reimann und Jodi Forlizzi [RMR01] eingeführt, beschäftigt sich diese Sichtweise mit der Funktionalität und Feedback eines Produktes je nach Benutzung durch Personen.

Social Interaction Design View

Der Schwerpunkt liegt bei der Kommunikation zwischen Mensch und Produkten. Bei den Kommunikationen können folgende Formen, je nach Teilnehmeranzahl, auftreten:

Tabelle 2.1: Kommunikationsformen Social Interaction Design View

Art	Beispiel
eins-zu-eins	Telefon
eins-zu-viele	Blog
viele-zu-viele	Aktienmarkt

Dabei ist die Art des Objektes, durch welches die Kommunikation bereitgestellt wird, völlig unwichtig. Somit spielt die Technologie höchstens eine untergeordnete Rolle.

2.2 Interaktionsvokabular

Die Interaktion wird im heutigen Gebrauch meist nur durch Technologie[4] oder durch Modalität[5] definiert. Indem ein Vokabular [SD10] bereitgestellt wird, kann eine gezieltere Beschreibung verschiedener Interaktionen vollzogen werden, besonders im Hinblick auf Analysen von Interaktionen während dessen Einsatzes. So kann z.B. auch eine Beurteilung möglich werden, welche Interaktionsformen zu bevorzugen sind.

Es ist bekannt, dass sich gewisse Bedienfunktionen gegenüber anderen etablieren konnten, es gibt aber wenig Aufschluss darüber, weshalb sie einen solchen Unterschied erzielten. Das durch Listen ”sliden”, anstatt sich durch sie zu ”klicken”, kann hier als Beispiel genannt werden. Indem allgemeine Interaktionseigenschaften definiert werden, welche nicht nur die bisherigen Kriterien üblicher Usability Forschung abdecken, können auch kleinere Interaktionsunterschiede beachtet werden. Das Zusammenspiel von Aktion und Reaktion von Interaktionen bekommt somit ebenso Beachtung.

[4] z.B. die Eingabe von Sprache
[5] z.B. die Sprache

Dem Interaktionsdesigner kann nicht nur ein weiteres Hilfsmittel bereitgestellt werden, es kann auch zu späteren Studien eingesetzt werden. Somit kann z.B. überprüft werden, ob sich die von einer Testperson wahrgenommenen Eindrücke mit zuvor definierten decken oder worin sie sich abweichend verhalten.

Ein üblicher Schritt zur Erstellung eines Vokabulars besteht darin, mit Hilfe eines Gegensatzpaares möglichst viele Kriterien zusammenzufassen.

Tabelle 2.2: Das Interaktionsvokabular, Usability Professionals 2010

langsam	schnell
abgestuft	stufenlos
ungefähr	präzise
behutsam	kraftvoll
sofort	verzögert
stabil	sich verändernd
vermittelt	direkt
räumliche Trennung	räumliche Nähe
beiläufig	gezielt
verdeckt	offensichtlich
anspruchslos	aufmerksamkeitsbedürftig

Eine weitere interessante Erkenntnis ist die Wahrnehmung der Interaktion als ein Prozess. Auch werden einige Interaktionen zu einem gewissen Grad ähnlich angesehen. Solch eine sich ergebende Clusterbildung kann in einer ”Karte” verdeutlicht werden.

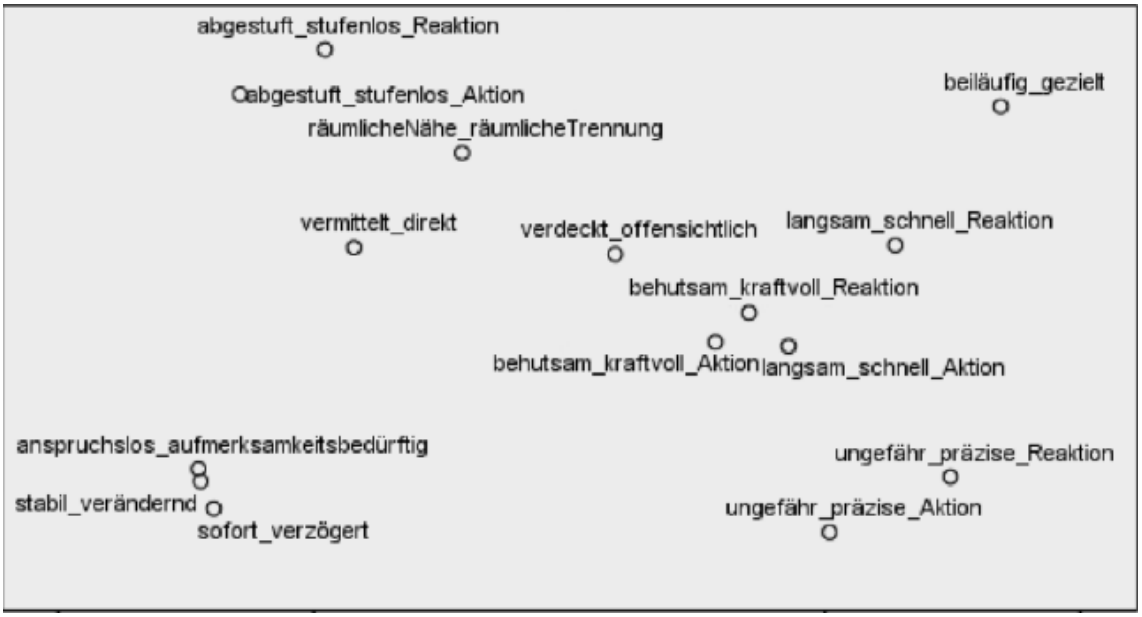

Abbildung 2.2: Ähnlichkeitsbeziehung in einer Karte

Eine weitere Spezifizierung kann mit Hilfe von u.a. pragmatischer und hedonischer Qualität durchgeführt werden. Der Fokus auf die hedonische Qualität steigert sich im größeren Ausmaß durch wichtige Erkenntnisse einiger Studien. Die hedonischen Qualitäten bekommen immer mehr Beachtung, besonders in der neuen Sichtweise *Joy of Use*. Somit erweitert sich die bisherige Palette von Nützlichkeit, Usability und einfacher Bedienung des Produktes. Es gilt immer weniger nur die Erfüllung des Faktors *easy to use*, sondern auch *joy to use*.

2.3 Joy of Use

> *" If you are still talking about ease to use then you are behind. It is all about the joy of use. Ease of use has become a given; it is assumed that your product will work. "*
>
> Bob Glass, 1997

In unserer heutigen Zeit der großen Auswahl Computerbasierter Systeme, kann sich joy of use [Hat01] als ein wichtiges Merkmal entwickeln, welches zu Erfolg oder Misserfolg führen kann. Es geht darum, nicht nur ein optisch gut designetes Produkt zu erschaffen, sondern viel mehr ein ansprechendes sowie funktionierendes. Nur Produkte, die auch Freude sowie Spaß an ihrer Benutzung vermitteln können, werden Erfolg haben. Freude gibt dabei die Tiefe bei einer Aktivität an und Spaß dient nur der reinen Ablenkung. Es gibt viele Ansichts- und Unterscheidungsweisen der genaueren Betrachtung von Freude. Insgesamt kann man es kurz als ein gern angestrebtes positives Gefühl ansehen. Mit Hilfe dieser Betrachtungsweise soll das Interesse und die Motivation des Nutzers geweckt werden. Vertrauen und Zufriedenheit werden ermöglicht, sowie kreativer Freiraum geschaffen.

Eine wohl bisher grundlegende Trennung zwischen Vergnügen und Arbeit sollte überdacht werden. Es kann nicht nur ein ernsthaftes oder rein zu Vergnügungszwecken erstelltes Produkt geben. Vielmehr gibt es einen großen Einsatzbereich von Computerbasierten Systemen, die im alltäglichen Lebensbereich dazwischen liegen. Inwieweit eine Umsetzung von joy of use erreicht werden kann, ist bisher umstritten. Bisher kann davon ausgegangen werden, dass in dem zur Zeit existierenden Usability-Konzept derartige Gesichtspunkte nicht vorhanden sind.

2.3.1 kurze Modellübersicht

Modell nach Brandtzæg[6] et al. [PBB04]:

Teilnahme und Kontrolle:
Das Gefühl der Kontrolle ist ein wichtiger Motivationsfaktor, um die Interaktion aufrecht zu erhalten. Ein Feedback sollte somit gestellt werden. Des Weiteren sollte das Computerbasierte System weniger unerklärliche Selbstentscheidungen[7] treffen. Es müssen aufgrund von sonst eintretender Langeweile Anforderungen gestellt werden.

Herausforderungen:
Langeweile oder Frustration, wenn keine Anpassung an die jeweiligen Benutzerfähigkeiten durchgeführt wird, kann entstehen.

Variation und Entscheidungsraum:
Der Mensch verlangt nach der Möglichkeit, Neues zu erleben und zu entdecken. Deshalb sollten gehäufte Wiederholungen weitestgehend vermieden werden.

Soziale Interaktion:
Seine Erfahrungen mit Anderen zusammen zu erleben oder auszutauschen wird ebenfalls als ein wichtiger Faktor angesehen.

Modell nach McCarthy[8]et al. [PW04]:

Anticipating:
Die Erlebnisbeeinflussung geschieht durch die Erwartungshaltung.

Connecting:
Neugier oder andere Gefühle führen möglicherweise dazu, sich auf ein Erlebnis einzulassen.

Interpreting:
Die Erwartungshaltung ändert sich mit den erzielten bzw. nicht erzielten Erwartungen des Benutzers.

Reflecting:
Die Erwartungen des Benutzers werden durch das Erlebnis erfüllt oder eingeschränkt. Die zeitliche Spanne umfasst dabei den Anfang bis hin zum Abschluss des Erlebnisses.

[6] abgeleitet von Demand-Control-Support-Model [RK90]
[7] zu intelligente, selbst agierende Systeme
[8] Gliederung des Erlebnisprozesses für gezieltere Benutzertests

Appropriating:

Das jetzige Erlebte wird mit dem Vorigen und Kommenden verglichen. Daraus resultierend entwickelt sich die Beurteilung, ob etwas positiv oder negativ ist.

Recounting:

Durch anderen Kontext oder Zeit verändert sich die Bewertung eines Erlebnisses. Weiterhin können persönliche Neigungen zu Wiederholungen von Erlebnissen führen.

Modell nach Hassenzahl [Has04]:

Dieses Modell beinhaltet den Beginn des Designprozesses bis hin zur Analyse der Produktcharaktere. Im Designprozess wird mit Hilfe von bestimmten Eigenschaften ein gewünschter Charakter erstellt. Anhand dieser Eigenschaften sowie der eigenen Produkterwartungen des Benutzers ergibt sich bei der Benutzung des Produktes ein subjektives Charakterbild. Damit bewertet der Benutzer die ihm empfundene Attraktivität des Produktes. Diese ist abhängig von der Situation, dem jeweiligen Kontext, zeitlich betrachtend und auch vergleichend mit anderen Produkten. Es wird immer Unterscheidungen zwischen der im Designprozess erstellten Produktcharaktere und der des Benutzers geben.

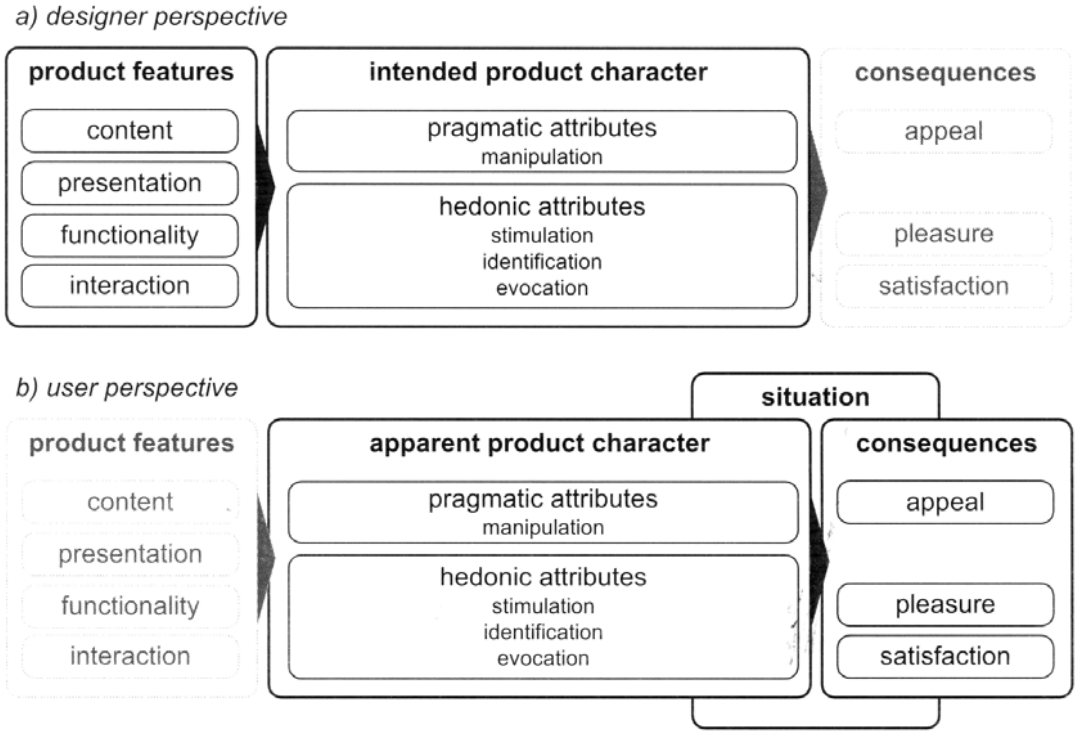

Abbildung 2.3: Nutzungserlebnisse und deren Schlüsselelemente (Hassenzahl Funology S. 32)

Anhand des von Hassenzahl aufgestellten Modells wird auf eine Unterscheidung zwischen pragmatischen und hedonischen Qualitäten eingegangen. Beide sind unabhängig voneinander und geben einen wichtigen Aufschluss über die dadurch resultierenden Produktcharaktere.

pragmatische und hedonische Qualität

Als pragmatische Eigenschaft wird die Manipulation des Umfeldes verstanden, welche aufgabenbezogen ist. Damit wird die Funktionalität des Produktes ausgedrückt sowie dessen Usability.Inbegriffen ist auch die als solche aufgefasste Wahrnehmung der Nutzer. Unter hedonischer Eigenschaft wird die Erweiterung möglicher Benutzerinteraktionen verstanden. Es werden Identifizierungen mit dem Produkt hergestellt, Herausforderungen gestellt und Stimulationen erzeugt. Man unterteilt die hedonischen Eigenschaften in unterschiedliche Gruppen:

Stimulation:
Aufmerksamkeit und Motivation können durch neu entdeckte Funktionen gefördert werden. Solch eine Stimulation kann zu neuen Lösungsansätzen führen, angespornt durch des Benutzers eigenem Wunsch nach Wachstum. Das Produkt sollte somit neue Eindrücke, Einblicke und Varianten ermöglichen.

Identification:
Das Produkt sollte eine Identität besitzen und diese dem Benutzer vermitteln.

Evocation:
Ein Produkt, das Erinnerungen bei dem Benutzer hervorruft, entwickelt meist einen starken persönlichen Wert.

Beide Eigenschaften, die pragmatischen und hedonischen, werden klar voneinander getrennt und als unabhängig voneinander angesehen. Zusammen bilden sie den Produktcharakter, der je nach ihrer Gewichtung ausfällt. Je nach Nutzungssituation und auch durch empfundene sowie erwartete Produktcharaktere können Abweichungen auftreten. Im Idealfall wird davon ausgegangen, dass sich beide Qualitäten gleich stark ausprägen. Da dieses nur selten in die Realität umsetzbar ist, enthält der Produktcharakter eine der beiden Qualitäten in ausgeprägterem Maße. So finden sich z.B. im Arbeitsumfeld hauptsächlich handlungs-orientierte[9] Produkte, wogegen besonders im Unterhaltungsbereich selbst-orientierende[10] Produkte vorzufinden sind.

[9] act-product: starke pragmatische Ausprägung
[10] self-product: starke hedonische Ausprägung

2.4 Erweiterung des Interaktionsvokabulars

Das bestehende Gegensatzpaar kann auf seine bisherigen Qualitäten erweitert werden. Jede der Eigenschaften ist meist einer der beiden Qualitäten zuzuordnen. Somit kann schon im Designprozess die gewünschten Intentionen des zu nutzenden Interaktionskonzeptes festgelegt werden.

Tabelle 2.3: Das Interaktionsvokabular mit *pragmatischen* und **hedonischen** Zusammenhängen, Usability Professionals 2010

langsam	*schnell*
abgestuft	stufenlos
ungefähr	*präzise*
behutsam	kraftvoll
sofort	verzögert
stabil	**sich verändernd**
vermittelt	*direkt*
räumliche Trennung	**räumliche Nähe**
beiläufig	**gezielt**
verdeckt	*offensichtlich*
anspruchslos	**aufmerksamkeitsbedürftig**

2.5 Multi-Touch-Technik

Bei der Nutzung von Multi-Touch zur Interaktion mit einem Computerbasierten System gibt es einige wichtige Unterscheidungen, die im Voraus bedacht werden sollten. Eine gute Gegenüberstellung wird von Bill Buxton [Bux07] geliefert.

Touch-tablets vs. Touch screen

Die Unterscheidung besteht hier im Allgemeinen zwischen einer berührungssensitiven Bildschirmoberfläche und einer örtlich getrennten Berührungsfläche. Erstere wird als Touchscreen bezeichnet und letztere als touch pad bzw. touch-tablet.

Discrete vs. Continuous

Eine weitere Unterscheidung wird zwischen einer einzelnen und abgeschlossenen Benutzereingabe sowie einer zeitlich andauernden getätigt. Im Falle von Multi-Touch könnte eine einzelne Aktion das Aufklappen eines Drop-Menüs mit einem Finger und die Auswahl eines

Objektes mit einem anderen Finger sein. Eine andauernde Benutzereingabe ist z.B. eine Rotation eines Objektes um die mit einem Finger angegebenen Achse.

Degrees of Freedom

Der Grad der Freiheiten wird als Erweiterung der Benutzereingaben angegeben. Damit soll ausgedrückt werden, inwieweit besonders andauernde Eingaben unterstützt werden können.

Size matters

Je nach Größe der interaktiven Oberfläche sind bestimmte Gesten am sinnvollsten. Ebenso wird Aufschluss über die mögliche Anzahl gleichzeitiger Finger/Hände geliefert.

Orientation Matters - Horizontal vs. Vertical

Die Probleme großer horizontal ausgerichteter Oberflächen bestehen durch ungewollte Berührungspunkte beim Handabstützen.

There is more to touch-sensing than contact and position

Im Normalfall gibt das berührungssensitive Gerät den Berührungspunkt an, doch existieren weitere Ideen:

1. **Degree of touch/pressure sensitivity:**
   Darunter versteht man die unabhängige und durchgehende Erkennung des Ausmaßes der Berührung mit der Oberfläche, wogegen der Druck[11] der Berührung den Berührungspunkt vergrößert.
2. **Angle approach:**
   Einige Systeme unterstützen die Erkennung des Winkels des berührenden Fingers zur Oberfläche[12]. Es kann der Finger als virtueller Joystick genutzt werden oder mit ihm in der virtuellen Welt ein Vektor definiert werden. Einsatzorte wären 3D Anwendungen.
3. **Force vectors:**
   Das Prinzip basiert auf den Reibungskräften zwischen dem Finger und der Oberfläche. Dabei wird mit dem Finger in die gewünschte Richtung parallel zur Oberfläche Druck ausgeübt. Durch die Reibungskräfte bewegt sich der Finger nicht, wird aber von der

[11] wird nicht selbst ermittelt, lediglich die daraus resultierende Ausweitung der Fingerspitze
[12] z.B. Sensor Frame von McAvinney

druck- und positionssensitiven Oberfläche erkannt und entsprechend umgesetzt. Eine Rotationserkennung ist auch möglich.

Size matters 2

Die Größenerkennung der Berührungsfläche ist eine gute Möglichkeit, um einen Finger von einem anderen Körperteil zu unterscheiden.

Single-finger vs. multi-finger

Der Großteil heutiger berührungssensitiver Oberflächen ist single touch, was die Möglichkeiten der Gesten stark einschränkt, vergleichbar mit der Nutzung von Maus und Joystick. Der Mensch hat mehrere Extremitäten und diese sollten auch ausgenutzt werden.

Multi-point vs. multi-touch

Der Fehler, der oft begangen wird, ist nur die Umsetzung von Gesten bzw. Interaktionskonzepten in Form von multi-point.

Multi-hand vs. multi-finger

Die Befehlseingaben können durch Nutzung von mehreren Fingern und Geräten entstehen, weiterführend auch durch Nutzung verschiedener Hände sowohl auf einem wie mehreren Geräten[13].

Multi-person vs. multi-touch

Das Computerbasierte System [14] muss folgende Unterscheidungen bei der Identifikation von Gesten leisten können, ob die erkannten Berührungspunkte zu einer oder zu mehreren Personen gehören.

Point vs. Gesture

Interessant ist nicht nur die Positionserkennung der Geste, sondern auch deren Ausführung.

[13] Abhängig von der Größe des Gerätes und des daraus resultierenden Lösungsansatzes
[14] z.B. DiamondSystem von MERL

Stylus and/or finger

Die Bedienung des Gerätes mit Hilfe eines Fingers oder einer Schreibspitze hat seine eigenen Vor- bzw. Nachteile und nicht Allen ist bewusst, dass Fingergesten nicht immer die optimalen Lösungen sind.

Hands and fingers vs. Objects

Es besteht die Möglichkeit, nicht nur Schreibspitzen, sondern auch andere reale 3D- Objekte als Bedienelement zu nutzen. Solche Oberflächen können bestimmte Objektformen registrieren sowie deren Ausrichtung und anderen Eigenschaften. Dies erweitert die denkbare Liste von Hilfsmitteln, um dem Gerät Eingaben zu übermitteln.

2.5.1 Geschichtlicher Überblick

> *" Multi-touch will have a lot next Kings, and with care, none of them will be Karl X of France. "*
>
> Multi-touch is Dead, Long live Multi-touch CHI 2009

Folgendes soll einen Überblick geben, welche Stadien die gestenbasierte Interaktion und Multi-Touch durchlief. Dabei muss bewusst betont werden, dass die jetzigen Systeme von der Arbeit vieler vergangener Jahre profitieren und für künftige Systeme den Weg bahnen werden.

Ein wichtiger Hintergrund, den man bei vielen angeblich neuen Technologien vergisst, ist ihr wirkliches Alter. So verhält es sich auch mit der in den letzten Jahren immer wieder gehypten Technologie bezüglich berührungs-basierten Eingaben. Seit Mitte 1960 arbeitete IBM an dieser Idee und in den Jahren um 1970 gab es die ersten Systeme. Ein wichtiges System dieser Zeit war das PLATO IV System. Weitere Entwicklungen wurden getätigt, doch das ganze Interesse verschob sich immer mehr zu Nischenprodukten. Bedingt war dies u.a. durch die eingeführte Cursorsteuerung der frühen Apollo Workstations, die später beispielsweise bei Laptops umgesetzt wurde. Es lassen sich die Nischenprodukte bis zum Jahr 2000 in 3 Gebiete einteilen:

1. Kioske, darunter auch ATM-Geräte
2. Point-of-sale-Geräte
3. Mobile Geräte, beginnend mit PDA und ab Mitte 1990 mit Mobiltelefonen

Im Allgemeinen ist bekannt, dass jede Technologie von ihrem Erfindungsprozess bis hin zur ihrer gehäuften Nutzung und somit ihres Höhepunktes einige Zeit benötigt. Nachdem über 20 Jahre an unterschiedlichen Umsetzungen von gestenbasierten Eingaben gearbeitet wurde, kam um das Jahr 2000 der Durchbruch. Interessanterweise sind die meist innovativ angepriesenen Umsetzungen nichts neu Erfundenes, sondern eine neue Umsetzung von Altem und Bekanntem.

Abbildung 2.4: Gartner's fünfphasiger Hype-Kreislauf: abgeändert für Multi-Touch nach einem Diagramm von Jeremy Kemp

In den letzten Jahren wird immer mehr Personen bewusst, dass sich mit Multi-Touch eine weitere gute Möglichkeit zur Interaktion mit Computerbasierten Systemen bietet. Es fehlen nur die Antworten, für welche Situationen und Umsetzungsformen sie dem Menschen hilfreiche Unterstützung bieten können. Dabei sollte der Schwerpunkt nicht unbedingt nur auf die Hardware gesetzt werden, sondern auch auf die möglichen Formen von Gesten sowie deren Umsetzung. Ein weiteres angestrebtes Gebiet ist die Kombination von Multi-Touch mit anderen Eingabevarianten, da die Stärken gestenbasierter Eingaben nur in bestimmten Situationen zu Tage kommen können.

2.5.2 Interaktive Oberflächen

Dieser Abschnitt dient als kurzer Überblick über die bisher gängigen Umsetzungen von Multi-Touch-Technologien der letzten Jahre, zusammengefasst aus [AT09]. Da sich die Bachelorarbeit weniger mit der technischen Umsetzung von Gestenerkennungen beschäftigt, wird nur die grobe Funktionsweise der unterschiedlichen Varianten aufgezeigt und zusätzlich ein Einblick zu Vor- und Nachteilen beschrieben.

Frustrated Total Internal Refelction (FTIR)

Das genutzte Prinzip dieser Variante ist die sogenannte Totalreflexion[15], welches mit Hilfe dem Snell'schen Gesetz berechnet wird. Fingerberührungen auf der mit Infrarotlicht ausgeleuchteten Plexiglasscheibe erzeugen diese Lichtbrechung, welche mit Infrarotkameras aufgezeichnet werden. Die Berechnung nach dem genannten Gesetzt liefert die Fingerposition.

Tabelle 2.4: Vor- und Nachteile von FTIR

Vorteile	**Nachteile**
starker Blob-Kontrast	Oberfläche muss nachgeben können
kleine Berührungspunkte erkennbar (Stifte)	keine Objekterkennung möglich
Berührungsstärke ermittelbar	
kein geschlossener Kasten nötig	

Diffused Illumination (DI)

Es gibt zwei Arten dieser Technik, wobei beide auf dem Kontrast zwischen Bild und Berührungspunkt beruhen. Bei der *Front Diffused Illumination* wird ein Schatten beim Berührungspunkt erzeugt, indem das Umgebungslicht von oben durch die Berührungsfläche scheint und ein *Diffuser* ober- oder unterhalb der Berührungsfläche angebracht ist. Dagegen wird bei der *Rear Diffused Illumination* das von unterhalb der Berührungsfläche kommende infrarote Licht an der Berührungsstelle gebrochen, mit einer Infrarotkamera aufgezeichnet und anhand dessen die Position berechnet.

Tabelle 2.5: Vor- und Nachteile von DI (Front)

Vorteile	**Nachteile**
jegliches durchscheinendes Oberflächenmaterial (fest wie nachgiebig)	schwer gleichbleibende Ausleuchtung beizubehalten
keine LED-Rahmen	häufige Fehlererkennung
Erkennung von Fingern, Überdeckung	keine Erkennung von Objekten, Fiducials
kein geschlossener Kasten nötig	unbekannte Einsatzbereitschaft, da von Umgebungslicht abhängig
benötigt nur Diffuser-/Projektionsoberfläche oben/unten	

[15] Lichtübergang von einem Material zu einem anderen erzeugt Lichtbrechungswinkel

Tabelle 2.6: Vor- und Nachteile von DI (Rear)

Vorteile	**Nachteile**
jegliches durchscheinendes Oberflächenmaterial (fest wie nachgiebig)	häufige Fehlererkennung
keine LED-Rahmen	geschlossener Kasten nötig
Erkennung von Objekten, Fingern, Fiducials, Überdeckung	schwer gleichbleibende Ausleuchtung beizubehalten
	geringer Kontrast von IR-Blobs

Laser Light Plane (LLP)

Der Einsatz von mehreren Lasern in den Ecken der Oberflächen helfen bei der Ermittlung einer Berührungspostion. An der jeweiligen Stelle, in der sich ein Strahl bricht, kann ein Berührungspunkt ausgemacht werden.

Tabelle 2.7: Vor- und Nachteile von LLP

Vorteile	**Nachteile**
jegliches durchscheinendes Oberflächenmaterial (fest wie nachgiebig)	keine Erkennung von Objekten, Fiducials
keine LED-Rahmen	keine Ermittlung von Berührungsstärke
kein geschlossener Kasten nötig	Berührungsverdeckung möglich (1-2 Laser)

Diffused Surface Illumination (DSI)

Prinzipiell ist DSI mit FTIR vergleichbar, es wird lediglich ein spezielles Acryl eingesetzt. Dieses beinhaltet besondere Partikel, welche das von den Ecken kommende IR-Licht zurücksenden und einen Effekt wie bei DI (mit gleichbleibender Ausleuchtung) erzeugen.

Tabelle 2.8: Vor- und Nachteile von DSI

Vorteile	**Nachteile**
kein nachgiebiges Oberflächenmaterial nötig	besonderes Acryl (teuer)
Erkennung von Objekten, Fiducials, Überdeckung	IR-Blobs haben geringen Kontrast
Erkennung von Berührungsstärke	Einschränkung der Oberflächengröße (aufgrund nachgiebigem Material)
gleichbleibende Ausleuchtung	

LED Light Plane (LED-LP)

Dieses kann mit LLP verglichen werden, nur werden um die Oberfläche herum (mindestens 2) Infrarot LED eingesetzt. Unterhalb der Oberfläche erkennt eine Kamera die Stellen, wo sich das Licht durch Oberflächenberührung bricht.

Tabelle 2.9: Vor- und Nachteile von LLP

Vorteile	**Nachteile**
jegliches durchscheinendes Oberflächenmaterial (fest wie nachgiebig)	keine Erkennung von Objekten, Fiducials
kein geschlossener Kasten nötig	Erkennung von Überdeckung nicht immer möglich
	Benötigt LED- Rahmen, LED mit geringerer Lichtintensität

3 Software

3.1 TUIO

Mit Hilfe des TUIO-Protokolls [MK05] wird eine Schnittstelle zur Datenübertragung bereitgestellt. Diese Daten enthalten Informationen über Interaktionen mit der Multi-Touch-fähigen Oberfläche. Dies können einerseits verschiedene Objekte oder andererseits Multi-Touch-Ereignisse sein.

Das gängige Funktionsprinzip dabei ist, dass eine Erkennungssoftware[1] als Server an eine Klientensoftware übermittelt. Der Server sendet die Daten mittels UDP an den Localhost, wofür im Normalfall der Port 3333 genutzt wird. Der Klient, welcher das TUIO-Protokoll encodiert, ist das verwendete Framework MT4j.

Für die Erstellung der prototypischen Applikation wird sich nicht weiter mit der Funktionsweise und Implementierung des TUIO-Protokolls und der Erkennungssoftware beschäftigt. Der Schwerpunkt liegt bei dem TUIO-Klienten, der durch das Framework MT4j umgesetzt wird.

[1] es gibt eine Vielzahl, darunter z.B. reacTIVision und CCV

3.2 MT4j

Die Funktionalität der MT4j Plattform begründet sich auf der Einteilung in verschiedene Schichten. Durch Ereignisse wird eine Kommunikation zur nächsten Schicht bereitgestellt. Nachfolgendes Bild gibt den Schichtenaufbau und dessen Funktion wieder [UL10].

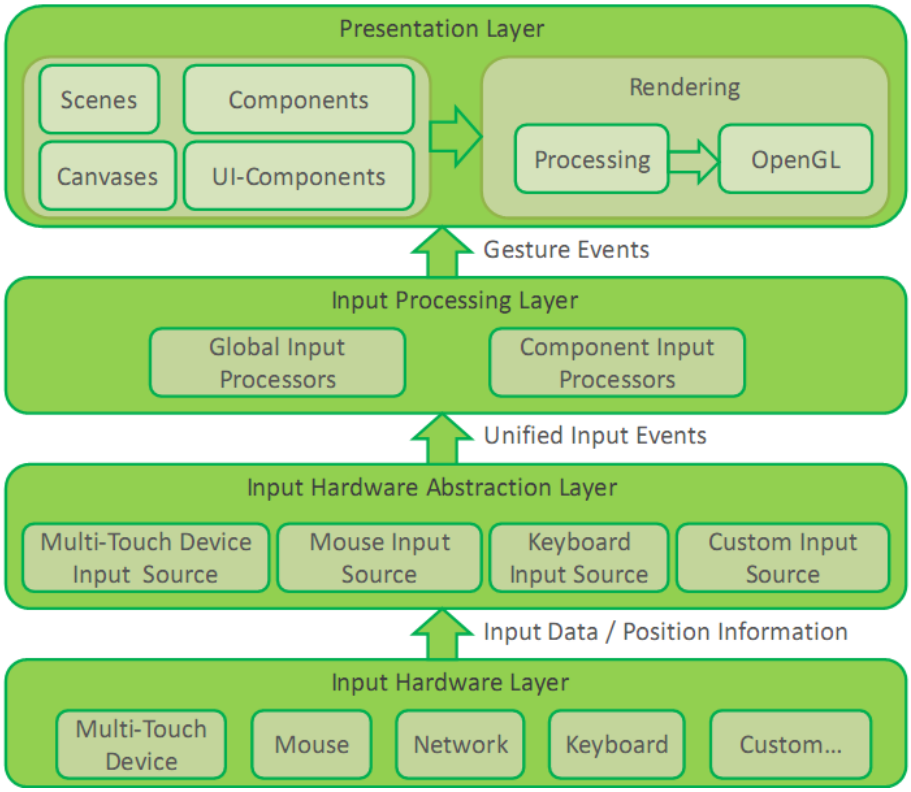

Abbildung 3.1: MT4j Plattformarchitektur Überblick

Input Hardware Abstraction Layer

Die durch Eingabehardware erzeugten Rohdaten werden in dieser Schicht zu einheitlichen Eingabeereignissen umgewandelt. Standardgemäß sind in dem MT4j die Eingabegeräte wie Maus, Tastatur und 2 Multi-Touch Eingabeprotokolle[2] enthalten und unterstützt. Alle diese Eingabequellen können synchron und in Kombination genutzt werden.

Des Weiteren kann jederzeit andere Hardware als Eingabequelle benutzt werden. Dazu muss die abstrakte Super-Klasse aller Eingabequellen erweitert werden sowie eine Funktionalität spezifisch des Typs der Eingabe erfolgen.

[2]TUIO-Protokoll und Windows 7 Touch API

Input Processing Layer

Der erste Zustand wird benutzt, wenn alle Eingaben bearbeitet werden müssen und nennt sich globale Eingabebearbeitung. Die registrierten Eingabebearbeiter reagieren nacheinander auf die jeweilige Eingabe durch die verschiedenen Eingabequellen. Es ist auch möglich, die Eingabe zu verändern, bevor sie zur nächsten Schicht weitergeleitet wird.

Der zweite Zustand befindet sich auf dem Komponentenlevel und erlaubt die Bearbeitung von Eingaben, die nur auf eine Komponente abzielen. Dazu gehören hauptsächlich Multi-Touch-Gesten wie Rotations- und Skalierungsgesten. Die Registrierung der Eingabebearbeiter von Komponenten kann modular für jede Komponente erfolgen. Nachdem die Kriterien für eine Multi-Touch-Geste getroffen wurden, gibt der Eingabebearbeiter ein Gestenereignis zu der jeweiligen Komponente weiter. Dieses beinhaltet die Informationen der erkannten Geste, welche durch die Komponente an den jeweiligen Gestenlistener gesendet wird. Je nach dem angesprochenen Gestenlistener wird die Komponente entsprechend modifiziert. Dies betrifft sowohl ihre Erscheinung als auch ihre Ausmaße.

Beide Zustände können abgeändert und erweitert werden.

Presentation Layer

Mit Hilfe von diversen grafischen Komponenten wird die Erstellung von 2D- und 3D-Applikationen ermöglicht:

Scenes

Szenen dienen dem Prinzip der Kapselung und ermöglichen die klare Trennung zwischen der Eingabebearbeitung und der Darstellung eines Teilaspektes der Applikation. Es ist dadurch möglich, mehrere Szenen zu benutzen sowie zwischen ihnen zu wechseln.

Components

Der strukturelle Aufbau des Szenen-Graphen besteht aus mehreren, hierarchisch untereinander angeordneten Komponenten. Die Basis stellt eine Grundkomponentenklasse dar. Unterklassen überschreiben bestimmte Methoden, um bestimmte Funktionen zu ermöglichen. So erreichen sie z.B. die Sichtbarkeit, indem die Zeichenmethode abgeändert wird. Primitive Grundformen[3] stehen vordefiniert zur Verfügung, ebenso auch darauf aufbauende komplexere Strukturen[4]. Eigene erstellte Komponenten können auf vorhandene Funktionalitäten[5] zurückgreifen.

Canvas

Die Ausgangskomponente jeder Szene bildet die Canvaskomponente und dient als Verbindung zwischen der Schicht für die globale Eingabebearbeitung und der Präsentationsschicht. Somit gehen alle Eingabeereignisse durch die Canvaskomponente, und sie kontrolliert die Position der einzelnen Komponenten. Die Canvaskomponente zeichnet sich und seine Kinderkomponenten auf rekursive Art.

Rendering

Für das Rendern werden die Processing Bibliotheken als Hilfe genutzt, ein Open Source Java Toolkit. Durch Nutzung einer Abstraktionsschicht für das Rendern kann eine Software- oder Hardwarebeschleunigung ermöglicht werden.

[3] Linie, Rechteck, Polygon, Kreis, Ellipse
[4] Rendern von Bildern, skalierbaren Vektorgrafiken und 3D Modellen
[5] Positionierung, Berührungserkennung, Gestenbearbeitung

4 Interaktion mit Hilfe von Multi-Touch

4.1 Gestenbasierte Interfaces

Wenn es darum geht, ein gutes gestenbasiertes Interface zu erstellen, sollte folgenden Punkten jeweils Beachtung geschenkt werden [Saf08]:

Discoverable

Dies ist eines der wichtigsten Probleme, welches gelöst werden muss. Es geht darum, jedem Nutzer ohne Umwege schnell erkenntlich zu machen, wie man das Computerbasierte System ansprechen kann. Allgemein muss erkannt werden, dass es sich zunächst um eine interaktive Umgebung oder im Falle von Multi-Touch um eine berührungssensitive Oberfläche handelt. Des Weiteren muss auch das Wissen vorhanden sein, wie eine Interaktion möglich ist.

Bei den Objekteigenschaften, die über dessen Bedienung Auskunft geben, handelt es sich im Allgemeinen um *affordances*[1]. So ist eine affordance bei einer Taste das Drücken und gibt somit Aufschluss über dessen Benutzung.

Trustworthy

Ein Computerbasiertes System, dass bei einem Benutzer kein vertrauenswürdiges Gefühl weckt, wird schlicht und einfach gemieden. Diese Entscheidung geschieht innerhalb einer kurzen Zeitspanne. Es muss ein Mittel geschaffen werden, den Argwohn des Benutzers zu überbrücken. Auch müssen Dinge wie Kompetenz, Sicherheit sowie Privatsphäre eingehalten werden.

[1] mindestens 1 benötigt, n viele möglich

Responsive

Der Benutzer muss das Gefühl haben, verstanden und gestützt zu werden. Durch den physischen Kontakt zwischen Mensch und Gerät, der bei einem Multi-Touch-basierenden System hergestellt wird, findet im menschlichen Gehirn eine Erwartungshaltung einer sofortigen Reaktion statt. Es ist deshalb vonnöten, ein Feedback[2] zu geben. Dies gilt für jede noch so kleine Interaktion, denn ohne eine Übermittlung einer Antwort kann der Benutzer seine Eingabe als nicht erkannt ansehen. Bei einer längeren Bearbeitungszeit einer Eingabe durch das System muss dessen Bearbeitung signalisiert werden. Im Normalfall geschieht dies z.B. durch einen Fortschrittsbalken.

Sie wird abhängig von der jeweiligen Reaktion in vier weitere Fälle unterschieden:

1. **Immediate:**
   Das Produkt bzw. der Service gibt eine Antwort innerhalb von 10 bis 100 Millisekunden. Eine solche Zeitspanne wird im Allgemeinen als eine sofortige Antwort wahrgenommen und sollte daher immer angestrebt werden. Wenn eine Buchstabentaste gedrückt wird, erscheint dessen Buchstabe sofort auf dem Bildschirm.

2. **Stammer:**
   Benötigt die Antwort des Systems länger und dauert bis zu einer Sekunde, ist dies eine empfundene Verzögerung. Einzelfälle werden schnell ignoriert, bei gehäuftem Auftreten wird es aber als schwammig empfunden.

3. **Interuption:**
   Eine noch längere Antwortzeit als 1 Sekunde führt dazu, dass der Benutzer seine Aufmerksamkeit von der getätigten Aufgabe hin zum Produkt bzw. Service selbst richtet.

4. **Disruption:**
   Handelt es sich um 10 oder mehr Sekunden, geht der Benutzer von einer nicht bearbeiteten oder korrekten Eingabe aus. Mit Hilfe eines Fortschrittsbalkens oder anderen Hilfsmitteln wird dem Benutzer nicht nur dessen Bearbeitung angegeben, sondern auch die Möglichkeit der Bearbeitungsabbrechung bereitgestellt.

[2] System hat Kommunikation mitbekommen und verstanden

Appropriate

Es sollte immer beachtet werden, was angemessen ist und was nicht möglich ist. Es gibt viele Kulturen, Religionen und je nach Einsatzort auch verschiedene Situationen sowie Kontextabhängigkeiten. Besonders bei einem Interaktionskonzept, dessen Grundlage Gesten sind, werden diese nicht auf der ganzen Welt ähnlich verstanden. Gesten haben nie dieselbe Bedeutung und sind somit nie ähnlich einsetzbar.

Als Überblick dienend, muss folgenden Punkten Beachtung geschenkt werden:

- **Power distance:**
  Sie gibt den Wert der Akzeptanz je nach kulturübergreifendem gebotenen Inhalt an.
- **Individualism vs. collectivism:**
  Hierbei wird in der jeweiligen Kultur berücksichtigt, ob starke Familienbindungen vorhanden sind oder nicht.
- **Masculinity vs. feminity:**
  Die Beachtung von starken bzw. dominierenden Geschlechtern sowie deren jeweiliger Rolle ist ausschlaggebend.
- **Uncertainty avoidance:**
  Darunter wird das Vorhandensein einer Toleranz zu Unbekanntem sowie dessen Ausprägungsgrad verstanden.
- **Long-term vs. Short-term orientation:**
  Es wird hiermit die Ausprägung von Vergangenheit, Gegenwart und Zukunft zueinander erläutert.

Meaningful

Ein System, welches keinen Benutzungsgrund übermitteln kann, ist kein gutes System.

Smart

Das Computerbasierte System hat die Aufgabe all das, was der Mensch nicht oder weniger gut ist in der Lage zu bewerkstelligen, selbst zu erledigen. Darunter wird die Fähigkeit der Unvergesslichkeit, schnellen Berechnung oder der Erinnerung verstanden.

Clever

Eine gute Fehlertoleranz des Systems bei getätigten Eingaben wird unter diesem Punkt verstanden. Im Falle von gestenbasierter Interaktion muss geklärt werden, welche Abweichung von den prototypischen Gesten erkannt wird.

Playful

Durch spielerisch aufgebaute Benutzungsweise kann eine Interaktion mit dem Computerbasierten System gefördert werden. Auch wird dadurch die eigentliche Benutzung initialisiert. Das Ausprobieren, wie etwas funktioniert, neue Funktionen sowie neue Eingabevarianten zu ergründen, bei Multi-Touch dementsprechend Gesten, wird dadurch ermöglicht.

Es sollte nur schwer sein, Fehler zu produzieren, und sie müssen jederzeit rückgängig gemacht werden können. Spiel und Spaß bei der Nutzung des Systems werden beendet, wenn der Mensch sich machtlos bzw. gefangen fühlt.

4.2 Gesten

Um einen guten Umriss der möglichen Umsetzungen von Gesten in den unterschiedlichen Systemen der letzten Jahre schaffen zu können, muss eine erste Unterscheidung getätigt werden. Der Hauptpunkt liegt in deren Häufigkeit der Nutzung. Gesten, welche schon in vielen Systemen davor umgesetzt wurden, sind bei dem späteren Endbenutzer mit einem Wiedererkennungswert behaftet. Bei neuartigen Gesten muss zunächst deren Ausübung und Funktion bekannt sein. Da eine Einschätzung möglicher oder zur Zeit in der Testphase befindlicher Gesten weniger sinnvoll ist, soll sich dieser Abschnitt mehr mit bereits etablierten und bekannten Gestenformen beschäftigen, die beispielhaft im Anhang zu finden sind.

Eine weitere Eingrenzung von Gesten wird vorgenommen, da nur die Gesten für Touchscreen bzw. interaktive Oberflächen von Interesse sind. Viele andere Gestenformen, welche im Raum[3] geschehen, werden nicht weiter betrachtet. Anzumerken ist, dass eine Kombination einiger Gesten im Raum mit interaktiven Oberflächen umsetzbar sind, nur müssen dafür zusätzliche Sensoren existieren.

Die folgenden Gesten besitzen alle ein typisches Einsatzgebiet, doch lassen sie sich auch für andere Funktionen einsetzen. Die Umsetzungsform und das Einsatzgebiet der einzelnen Gesten haben die gleiche Gewichtung wie das Zusammenspiel der Gesten untereinander.

[3] Gesten für interaktive Oberflächen geschehen auf der x- und y-Achse, da die Berührungsstärke keine richtige Tiefenerkennung ist.

4.2.1 Gesten auf Basis von Tap

Solche Gesten sind in der Ausführung die einfachsten und besitzen einen hohen Wiedererkennungswert. Im Grunde ist ihre Ausführung ähnlich zu denen, welche mit anderen Eingabegeräten getätigt werden. Sie können dazu genutzt werden, um folgende Funktionen bereitzustellen:

Tap to Open/Activate

Eine Berührung auf eines bestimmten Objektes oder in einem bestimmten Bereich führt zu einem Auslöser. Diese Gestenform findet ihre Anwendung in fast jedem Gerät mit gestenbasierter Berührungseingabe und dient allgemein als Pendant zum Mausklick.

Tap to Select

Diese Geste dient dazu, ein Objekt auf dem Bildschirm auswählen zu können. Des Weiteren wird sie meist als Alternative zur Geste Tap to Open/Activate genommen. Sind beide Gesten vorhanden, wird meist mit einer einzelnen Berührung das Objekt ausgewählt und mit einer doppelten Berührung das Objekt aktiviert.

Wichtig bei beiden Gesten ist die Art, wann die Befehlsausführung tatsächlich stattfindet. Man spricht auch von *activate-on-press* sowie *activate-on-release*, und es kann eine entscheidende Auswirkung mit sich führen. Standardgemäß wird bei mechanisch basierten Eingabegeräten[4] die Befehlsausführung nach Erkennen der Berührung durchgeführt, während bei Touchscreens die Ausführung erst nach Beendigung der Berührung geschehen sollte.

Tap to Stop

Solange eine Aktion vom System automatisch über längere Zeit weiter durchgeführt wird, sollte dem Benutzer die Möglichkeit gegeben werden, diese zu pausieren und stoppen zu können. Die einfachste und intuitivste Form zur Unterbrechung ist eine Berührung des Benutzers in dem Bereich, wo die Aktion durchgeführt wird. Es kann auch dazu genutzt werden, eine vorher unterbrochene Aktion weiter durchlaufen zu lassen.

[4] z.B. Button einer Tastatur

4.2.2 Drag to Move Object

Während die Berührung über einem Objekt aufrecht erhalten wird, kann es simultan mit der Fingerbewegung an eine andere Position bewegt werden. Der Vorteil dieser Geste ist, dass das natürliche Empfinden, etwas greifen zu können, ermöglicht wird. Somit lassen sich auch bestimmte Kontrollelemente[5] in das Interface einbinden.

Damit die Position eines bestimmten Objektes veränderbar ist, muss entweder eine vorherige Selektion erfolgen oder man verbindet die Selektion und Bewegung in einer Geste. Ein zusätzlicher Beachtungspunkt ist der Bereich, in dem die Neupositionierung eines Objektes geschehen kann. In einigen Fällen wird meist die Bewegung in waagerechter oder senkrechter Ausrichtung erlaubt, besonders bei bestimmten Kontrollelementen oder aber auch um Benutzerfehleingaben entgegen wirken zu können.

4.2.3 Scrollen mittels Gesten

Es gibt eine Vielzahl verschiedener Gesten, die das Scrollen in einer Liste oder Ansammlung von Objekten ermöglicht. Einige von ihnen können sich gegenseitig ersetzen oder aufeinander aufbauen.

Slide to Scroll

Eine Berührungsbewegung in eine Richtung bewirkt das Scrollen des Bildschirminhaltes oder das Durchblättern der jeweiligen Objektliste. Das Scrollen wird solange durchgeführt, wie die Bewegung des Benutzers anhält. Oft wird eine Parallele zur Geste *Drag to Move* gezogen, da die Durchführung der Geste gleich ist. Es sollte aber bedacht werden, dass für das Scrollen keine vorherige Selektion des Objektes stattfindet. Es wird vielmehr der Bereich festgelegt, in dem die Berührungsbewegung ein Scrollen ermöglichen kann.

Slide and Hold for Continuous Scroll

Die Besonderheit ist, dass mit Hilfe der Berührungsbewegung die Scrollrichtung festgelegt und vom System solange ausgeführt wird, wie die Berührung durch den Benutzer aufrecht gehalten wird. Somit handelt es sich prinzipiell um die gleiche Funktion wie bei *Slide to Scroll*, das Scrollen ist nur eine kontinuierliche Aktion.

[5] z.B. Sliders (Schieberegler)

Two Finger to Scroll

Mit Hilfe von zwei Fingern wird eine Bewegungsgeste getätigt, um innerhalb des gewünschten Bereiches zu scrollen. Diese Geste wird vornehmlich dazu genutzt, um einerseits zu scrollen und die Möglichkeit zu besitzen, zeitgleich den Cursor für etwas anderes bedienen zu können. Somit wird diese Geste eher bei Trackpads von Laptops genutzt und ist weniger sinnvoll für Touchscreens.

Spin to Scroll

Indem eine kreisförmige Bewegung ausgeführt wird, kann mit dieser Gestenform ein Scrollen ermöglicht werden. Dabei erzeugt eine im Uhrzeigersinn ausgeführte Kreisbewegung ein Vorwärtsblättern und gegen den Uhrzeigersinn ein Zurückblättern. Oft wird diese Variante dazu eingesetzt, um ein schnelles Scrollen zu ermöglichen, besonders im Hinblick auf mobile Geräte und deren kleinen Bildschirmen zur Informationsdarstellung.

Inzwischen gibt es einige Abwandlungen, sodass mit einer kurz ausgeführten Kreisbewegung einfaches Scrollen erzeugt wird, während eine komplette Kreisbewegung die nächsten Elemente anzeigt. Es gibt auch Umsetzungen, die ein kontinuierliches, zeitliches oder schnelles Scrollen erzeugen. Demnach muss dem Benutzer einerseits die Funktion der Geste, wie auch der strukturelle Aufbau[6] der Liste, durch die geblättert wird, bewusst sein.

Fling to Scroll

Wird eine schnelle Bewegung eines Fingers über den zu scrollenden Bereich ausgeführt, erreicht man ein schnelles Durchblättern durch die Liste oder den Bildschirmbereich. Dabei gibt es Varianten, die ein langsames Durchblättern mit Hilfe der Fingerspitze erzeugen. Energischere Bewegungen des Fingers erzeugen dagegen ein schnelles Scrollen. Auch kann eine solche Geste zum Stoßen und Ziehen von Objekten genutzt werden.

Durch Nutzung unterschiedlicher Geschwindigkeiten der Ausführung, kann ein verlangsamendes Scrollen oder Bewegen eines Objektes ermöglicht werden. Erweiterbar ist auch die Benutzung eines zweiten Fingers zur Erzeugung einer schnellen, seitenüberspringenden Eingabe.

[6] z.B. Alphabetische Sortierung

4.2.4 Flick to Nudge

Diese Geste besitzt starke Parallelen zu *Fling to Scroll*. Die Unterscheidung liegt darin, dass eine in die Richtung ausgeführte Geste eine Bildschirm- oder Objektbewegung gleicher Richtung auslöst. Eingesetzt wird sie meist für einfache *drag and drop*-Aktionen, Gleitbewegungen oder um Objekte wegzuschieben[7].

Weitere Umsetzungen geben den Objekten ein künstliches physikalisches Verhalten. Ein angestoßenes Objekt verlangsamt sich von alleine während seines Zielweges. Des Weiteren um die Ausführungsstärke mit entsprechender Bewegungsstärke unterschieden.

4.2.5 Pinch to Shrink and Spread to Enlarge

Die Ausführung dieser Geste geschieht mit Hilfe von zwei Fingern. Dabei kann es sich um die Finger einer Hand oder um je einen Finger beider Hände handeln. Indem beide Finger näher zusammengeführt werden, wird der *Pinch to Shrink* Part aktiviert, während bei dem Auseinanderführen der Finger der Part *Spread to Enlarge* aktiviert wird. Im Allgemeinen wird die Geste über einem Objekt ausgeführt und bewirkt eine Objektverkleinerung bzw. eine Objektvergrößerung. Die Skalierung geschieht in den meisten Fällen proportional und in einem fließenden Übergang ohne explizite Skalierungsstufen. Beachtet werden sollte, dass das Ausgangsobjekt oft als Ausgangsgröße dient und von diesen 100% die Vergrößerung oder Verkleinerung stattfindet. Es sollte vom System aber auch eine Grenze gesetzt werden, inwieweit es möglich ist, eine neue Objektgröße festzulegen.

Mit dieser Geste kann nicht nur die Manipulation der Objektgröße erreicht werden, sie lässt sich auch für andere Funktionen einsetzen. So kann mit ihr an den Anfang und Ende eines Dokumentes oder Videos gelangt werden. Eine andere Einsatzform stellt die Zoom-Funktion ohne Veränderung der Objektgröße dar.

[7] auch aus dem sichtbaren Bildschirmbereich hinaus

4.2.6 Ghost Finger

Eine schon länger bestehende Variante, doch weniger bekannt, ist die Nutzung der Visualisierung[8] von Fingern. Dieses Prinzip wird angewendet, wenn sich des Benutzers Hände nicht im direkten Sichtbereich befinden. Darunter versteht man Situationen, in denen sich die Finger auf der Rückseite des Gerätes, innerhalb des Objektes oder einer anderen Position als der Visualisierung befinden. Somit sieht der Benutzer die Position seiner Finger auch auf kleineren Visualisierungsflächen und die eigenen Finger können die Bildschirmelemente oder Objekte nicht verdecken.

Solche Umsetzungen müssen dem Benutzer von der Funktion und Nutzung erst bewusst gemacht werden. Damit wird die Grenze zu den bekannten und genutzten Gestenformen, die in der vergangenen Zeit erfolgreich umgesetzt wurden, erreicht.

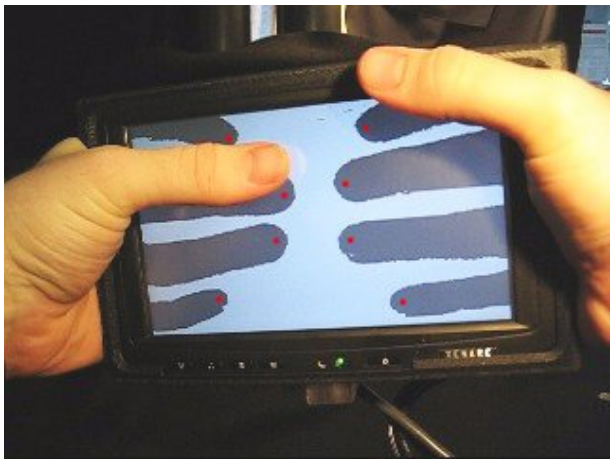

Abbildung 4.1: LucidTouch und Einsatz von Ghost Finger, LucidTouch: A See-Through Mobile Device

[8] transparente Formnachbildung der Fingerposition

5 Konzept

Mit dem folgenden Szenario wird eine mögliche Umsetzung der Nutzung von Multi-Touch betrachtet. Dabei soll das erdachte Szenario den Funktionsumfang des finalen Produktes grob umschreiben.

5.1 Szenario

Die prototypische Applikation soll jedem öffentlich die Möglichkeit bieten, einen Überblick über die eingetragenen Informationen zu erhalten. Bei den Informationen wird es sich hierbei um durchgeführte oder laufende Projekte von Studenten handeln. Ziel ist es, dass der jeweilige Nutzer sich über Projektparallelen und Vertiefungen erkundigen kann.

Das Multi-Touch-fähige Gerät befindet sich in einem frei zugänglichen Bereich. Im Idealfall wird es sich um einen Ort handeln, der von Jedem täglich betreten werden kann. So bieten sich der Pausenbereich oder Rezeptions- bzw. Eingangsbereich an, andere öffentliche Bereiche sind jedoch ebenso möglich. Es ist davon auszugehen, dass weder das System noch deren Funktion sowie Bedienung bekannt sein wird. Dadurch muss das System fähig sein, zukünftigen Benutzern eine Bedienung auch ohne Vorkenntnisse von gestenbasierten Eingaben zu ermöglichen.

Durch eine bestimmte Geste kann das System aus dem Ruhemodus bzw. der Vorschaufunktion in seinen Arbeitsmodus gebracht werden. Für den Benutzer sollte dann ein leerer Hauptbildschirmbereich erscheinen. Im unteren Teil befindet sich die Projektliste aller eingetragenen Projekte. Durch diese Projektliste kann mit unterschiedlichen Eingaben navigiert werden, bei Interesse wird von dem jeweiligen ausgewählten Projekt eine Vorschau von wichtigen Metadaten angezeigt. Wird das Projekt geöffnet, so erscheinen dessen zugehörige Daten im Hauptbildschirmbereich. Die Projektliste kann jederzeit für einen größeren Hauptbildschirmbereich und zur Übersicht ein- und ausgeblendet werden. Es kann im Hauptbildschirm durch die Projektinhalte einfach navigiert werden, Medien können bei Interesse aus ihrer Vorschaufunktion vergrößert oder in eine detailliertere Sicht gebracht werden.

Ist die Benutzung des Systems durch die jeweilige Person abgeschlossen, kann dies einerseits durch die Person oder nach bestimmten Bedingungen durch das System erfolgen. Beendet die Person den Arbeitsmodus, geht das System in den Ruhemodus über. Ein selbstständiger Übergang in den Ruhemodus kann durch eine festgelegte Zeit von keinen Eingaben erfolgen oder durch Nutzung von externen Sensoren. Ein vom System erfolgter Übergang sollte vorher immer vom System angezeigt werden, um einen noch möglichen, anwesenden Benutzer vorwarnen zu können. Somit kann jederzeit eine unerwünschte Aktivierung des Ruhemodus durch den Benutzer verhindert werden.

5.2 Dokumentation der Gesten

Abhängig davon, was für ein zu entwerfendes System angestrebt wird, werden entsprechende Vorüberlegungen, meist in Papierform getätigt. So wird ein Überblick über die im gesamten System auftretenden Gesten geschaffen. Deren Funktionsweise und Regeln sind ebenfalls von Interesse, um ein Gestenvokabular für das System aufzustellen. Einige wichtige Varianten, die eine Anwendung finden, werden noch folgend aufgeführt. Dabei handelt es sich um die gängigsten Varianten, welche oft Anwendung finden.

5.2.1 Gestural Modules

Für einfache Systeme wird inzwischen des Öfteren auf einfache Übersichten zurückgegriffen. Das Beispiel am Szenario umgesetzt, sieht wie folgt aus:

Tabelle 5.1: Gestural Module Document: genutzter Gestenpool und deren Funktion

Geste	**Aktion**
Tap	Item oder Kontrollfunktion auswählen/öffnen
Drag to Move	Bildschirmschoner deaktivieren Projektliste einblenden/ausblenden
Slide to Scroll	einfaches Scrollen: es wird 1 neues Element angezeigt
Spin to Scroll	schnelles Scrollen: es werden 5 neue Elemente angezeigt
2 Finger Pinch and Spread	Medien vergrößern/verkleinern

Zum Großteil wird hauptsächlich auf textbasierte Formen zurückgegriffen. In einigen Fällen, besonders um spezielle Situationen oder komplexe Gesten zu erklären, werden Skizzen verwendet. Da sich einige Gesten als gebräuchlich etabliert haben, reicht diese Dokumentationsform aus.

5.2.2 Use Cases

Bei den Use Cases handelt es sich um eine zeitlich aufwändigere Variante. Der Vorteil liegt darin, dass man gesondert auf einzelne Aufgabenbewältigungen eingehen kann. Von besonderer Relevanz sind die Felder Ausgangs- und Endbedingung, wie an folgendem Beispiel deutlich wird:

Titel

Projektliste einblenden/ausblenden

Beteiligte

1 Finger des Benutzers

Zweck

Der Benutzer möchte die Projektliste ausblenden/anzeigen.

Ausgangsbedingung

Die Projektliste wird vom System angezeigt/zur Zeit nicht sichtbar. Es gibt keine besondere Ausgangsbedingung seitens des Benutzers.

Endbedingung

Die Projektliste wurde vom System ausgeblendet/eingeblendet.

Schritte

Ausblenden: Der Reiter der Projektliste (linke obere Ecke) wird vom Benutzer berührt und festgehalten. Danach vollführt der Benutzer eine Abwärtsbewegung, bis die Projektliste, außer der Reiter selbst, nicht mehr sichtbar ist.
Einblenden: Der Reiter der Projektliste (linke untere Bildschirmecke) wird vom Benutzer berührt und festgehalten. Danach vollführt der Benutzer eine Aufwärtsbewegung, bis die Projektliste vollständig aufgeklappt und sichtbar ist.

Alternativen

Es sind keine Alternativen vorhanden.

Ähnliche oder alternative Use Cases

Engagement (Bildschirmschoner bzw. Demo des Startbildschirms beenden und Hauptbildschirm anzeigen lassen)

5.2.3 Task Analysis

Eine weitere Möglichkeit zur Dokumentation besteht darin, die möglichen Prozesse einzelner Aktivitäten in ihrem Ablauf zu formulieren. Typischerweise erstellt man Flow-Diagramme[1] zur Verbildlichung der einzelnen Entscheidungsmöglichkeiten. Im Hinblick auf die Benutzung von gestenbasierten Systemen sind die

- möglichen Entscheidungspunkte
- und deren resultierende Aktionen für den Benutzer und das System

von Wichtigkeit und sollten entsprechend behandelt werden.

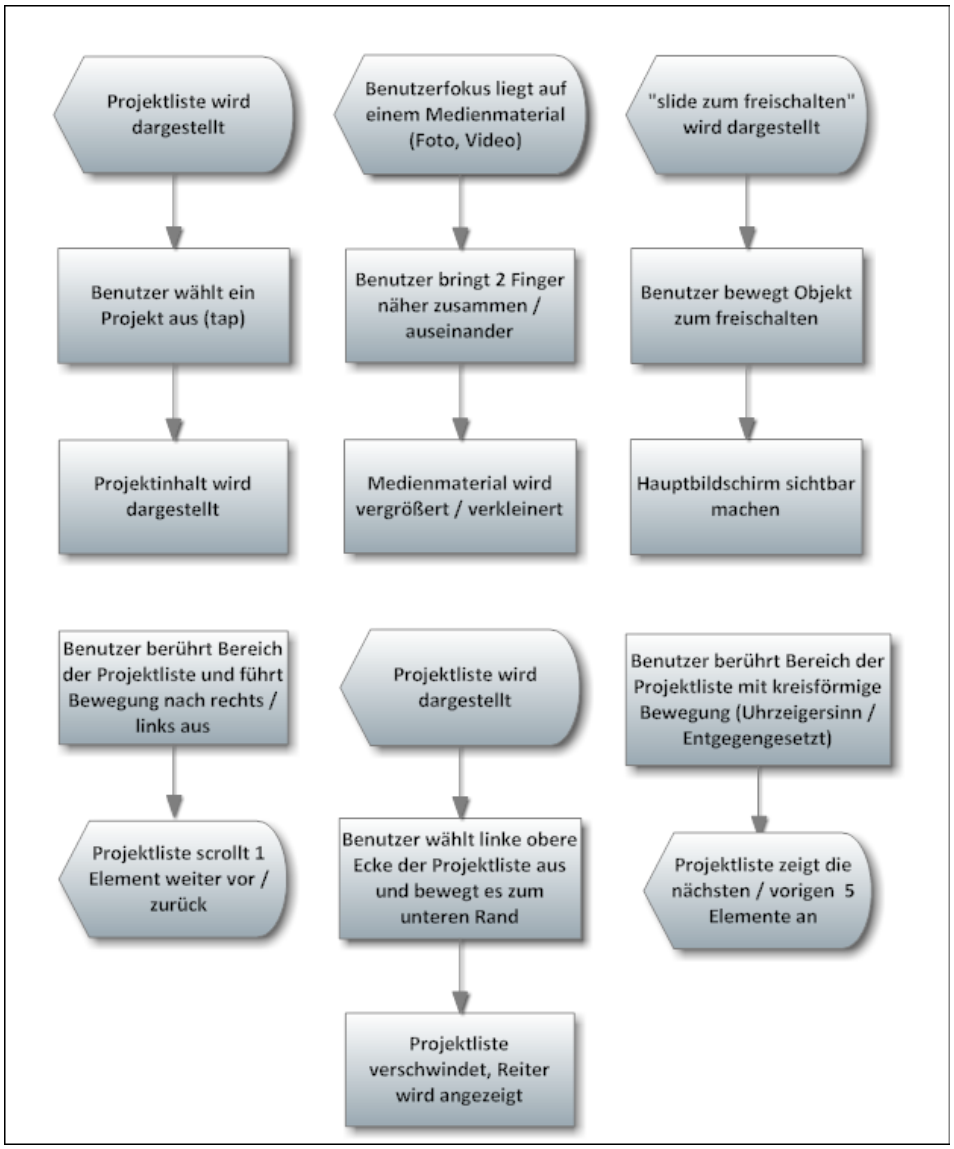

Abbildung 5.1: Ablaufdiagramme am Szenario

[1] Ablaufdiagramm

Daraus resultierend lässt sich auch die Art und Menge benötigter Gesten ableiten, die zum Funktionieren des Systems vonnöten sind. Es lassen sich auch mögliche Sensoren und andere Hardware erörtern. Solange die Gesten keine genauere Beschreibung benötigen oder Situationen mit Worten schwer ausdrückbar sind, ist die Nutzung von Flow-Diagrammen empfehlenswert.

5.2.4 Wireframes

Eine der geläufigen Papier-Prototypen, die bei Touchscreens von Interesse sind, kennt man auch unter dem Namen *pixel-perfect*. Wie schon der Name ausdrückt, soll bei dem Prototypen das gleiche Größenverhältnis genutzt werden, wie es später bei dem Endprodukt sein wird. Dadurch lassen sich u.a. zu kleine interaktive Schaltflächen oder andere Designfehler erkennen.

Controls

Der wohl wichtigste Punkt beinhaltet die Informationen der Position und Funktion der Bedienelemente. Was ist durch den Benutzer wie manipulierbar und wie reagiert das System auf bestimmte Aktionen durch den Benutzer? Es ist zu beachten, dass die Position, Größe und Bildschirmausnutzung nicht nach gleichen Maßstäben wie bei Web-Design oder Software-Design umsetzbar ist.

Conditional objects and states

Wenn ein Bedienelement erst nach einer bestimmten Bedingung benutzbar ist, muss dies kenntlich gemacht werden und eine Erklärung beinhalten. Im Idealfall wird davon ausgegangen, dass alle möglichen Zustände für das jeweilige Objekt aufgeführt werden. Zusätzlich ist auch die jeweilige Geste von Interesse, die als Auslöser dient.

5.2.5 Weitere Varianten

Einige Gesten sind zu komplex oder schwer mit Worten umschreibar. Für solche Fälle gibt es weitere Varianten, die dafür Abhilfe schaffen können. So kann man mit Hilfe von *Keyframes* einen wichtigen Moment einer Geste bildhaft darstellen. Meist ist es nicht mehr als eine Skizze für folgende Situationen: Auslösung, Aktivierung und Aktualisierung. Je mehr das gestenbasierte System an Größe und Platz im physischen Raum einnimmt, desto mehr nehmen die architektonischen Elemente an Bedeutung zu. So betrachtet man Faktoren wie Augenhöhe, physische Erreichbarkeit, Höhe und Breite genauer. Mit dem Storyboard kann eine ganze Sequenz von Gesten betrachtet werden und findet auch Anwendung bei neuen oder komplexen Gesten. Erweitert werden Storyboards durch Erklärungen aus unterschiedlichen Sichten. Filme bzw. Animationen sind ein weiteres Mittel, um Gesten im System zu dokumentieren.

6 Umsetzung

Um eine geeignete Umsetzung für eine prototypische Applikation zur Informationsdarstellung im öffentlichen Bereich zu erreichen, sollten die vorher festgelegten Vorgaben (siehe Abschnitt 1.3), die theoretischen Konzepte (siehe Kapitel 2) sowie die sich immer stärker etablierenden gestenbasierenden Interaktionskonzepte (siehe Kapitel 4) berücksichtigt werden.

Die Wahl fiel auf die Entwicklungsumgebung *MT4j*, welches in der Programmiersprache Java umgesetzt wurde. Somit ist die Applikation aufgrund der von Java genutzten virtuellen Maschine betriebssystemunabhängig und ermöglicht eine unkomplizierte lauffähige Version auf verschiedenen Systemen. Es sei auch darauf hingewiesen, dass es keinen zu großen Geschwindigkeitsnachteil heutiger Applikationen in Java gibt.

6.1 Programmarchitektur

Im Folgenden wird ein Überblick über die Softwarearchitektur der prototypischen Applikation zur Informationsdarstellung im öffentlichen Umfeld gegeben, indem auf die einzelnen Klassen und wichtigen Quellcodestellen eingegangen wird. Die Klassendiagramme sind dem Anhang dieser Bachelorarbeit zu entnehmen.

6.1.1 Systemüberblick

Die Applikation nutzt neben dem Framework *MT4j* zum Parsen von XML-Dokumenten *Jdom*. Deren hierarchische Bedeutung lässt sich dem Paketdiagramm entnehmen:

Abbildung 6.1: einfaches Paketdiagramm der prototypischen Applikation

Die mit *MT4j* erstellte Applikation nutzt die von *TUIO* bereitgestellte Erkennung der Gesten. Die Bibliotheken von *Processing* helfen bei der Erstellung und Bearbeitung grafischer Objekte, besonders im Bereich der Texturen. Beide Bibliotheken sind in das Framework fest integriert, deren Nutzung ist aber nicht zwingend nötig.

Zum Parsen der XML-Dokumente wird die externe Bibliothek *Jdom* genutzt, wobei diese nur die Klasse für das Parsen der XML-Dokumente lädt.

6.2 Parsen von XML

Das Parsen der XML-Dokumenten in der Applikation geschieht durch die Nutzung der API Jdom. Dabei handelt es sich um eine für Java entwickelte *open source* API, welche auf der Baumstruktur basiert. Mit Hilfe von Jdom kann man XML-Dokumente parsen, erstellen oder bearbeiten. Diese Bachelorarbeit nutzt die zur Erstellungszeit aktuelle Version 1.1.1, welche seit Juli 2009 erhältlich ist [BM00].

Die Klasse, welche das Auslesen der XML-Dokumente erledigt, die die benötigten get-Methoden und Hilfsmethoden bereitstellt, ist die Klasse *Jdom*. Die benötigten Importe für Jdom sind alle in der frei erhältlichen *jdom.jar* zu finden. Jdom benutzt keinen eigenen nativen Parser, sondern es wird üblicherweise einer der etablierten SAX2-Parser benutzt[1]. Dafür wird ein Objekt des folgenden Typs verwendet:

```
org.jdom.input.SAXBuilder
```

Es wird der parameterlose Konstruktor genutzt. Die *build()*-Methode von SAXBuilder erstellt ein Dokument-Objekt aus einer der möglichen Quellarten.

- InputStream
- Reader
- URL
- File
- String[2]

[1] u.a. Crimson, Xerces
[2] muss eine System-ID beinhalten

Zur Fehlerüberwachung werden die *IOException* und die *JDOMException* genutzt. Ersteres geschieht, wenn es Probleme beim Lesen des Dokumentes gab, letzteres tritt bei einem Aufbaufehler des Dokumentes ein. Tritt keiner dieser Fehler auf, kann mit den Methoden der verschiedenen Jdom-Klassen durch das Dokument navigiert werden. Eine allgemeine Umsetzung zum Einlesen eines XML-Dokumentes mit Hilfe eines SAX-Parsers zeigt die folgende *try-catch-Anweisung*:

```
SAXBuilder saxBuildername = new SAXBuilder();
try {

    saxBuildername.build(sourcetype);
    //assignments if no exception is thrown

}
//well-formedess error
catch(JDOMException e) {

    //assignments

}
catch(IOException e) {

    //assignments

}
```

Dieses einfache Beispiel versucht die XML-Quelle zu öffnen und zu lesen. Es macht nicht mehr, als das XML-Dokument nach einem konformen Aufbau zu überprüfen und bei Bestehen kann die Jdom-Baumstruktur erstellt werden. Dieses entspricht einem Programm, welches XML-Dokumente validiert, wenn noch an den betreffenden Stellen der Anweisungen Ausgaben getätigt werden.

Die Methode *loadXML(String file)* nutzt eine vom Aufbau gleiche *try-catch-Anweisung*. Mit Hilfe des Parameters wird der Pfad und der Name des XML-Dokumentes übermittelt. Dies ist vonnöten, da zu bestimmten Programmzeitpunkten ein anderes XML-Dokument geöffnet werden muss.

Der Aufbau der XML-Dokumente war nicht vorgegeben und wurde deshalb frei gewählt. Aufgrund dessen wurde der XML-Dokumentaufbau einfach gehalten und beinhaltet eine Verschachtelung von Elementen sowie deren zugehöriger Textinhalt. Es gibt zwei Arten von XML-Dokumenten, welche sich in ihrem Aufbau unterscheiden. Eines ist das XML-Dokument, welches die n-beliebige Anzahl von Projekten beinhaltet und bei dem Start der Applikation gelesen wird. Es liefert folgende Informationen:

- **Name**: Der Name des XML-Dokumentes eines Projektes, welches bei einer Projektauswahl geladen werden soll.
- **Icon**: Der Name der Bilddatei, welche beim Programmstart in der Projektliste auftaucht.

Bei dem Anderen handelt es sich um die jeweiligen XML-Dokumente der Projekte selbst. Es lässt sich in zwei Abschnitte einteilen. In dem Einen werden die Metadaten des Projektes geliefert, in dem Anderen ist der Projektinhalt zu finden. Die Metadaten werden bei einer Projektauswahl des Benutzers gelesen und an dem betreffenden Ort der Applikation angezeigt. Der Projektinhalt besteht hauptsächlich aus Textform, der beim Öffnen des Projektes ausgegeben wird. Zusätzlich wird aus den möglichen Medienelementen eine Medienliste erstellt. Ein Medienelement beinhaltet den Namen des Mediums und den Namen des Icons, welcher für die Medienliste als Vorschaubild dient.

Das Lesen der Elementinhalte kann grundsätzlich über zwei Arten getätigt werden. Die einfachere Form bedient sich der direkten Anfrage mittels der *getChild()*-Methode. So kann man vom Root-Element beginnend, mittels mehrfacher *getChild()*-Aufrufe, den betreffenden Textinhalt mit *getChildText()* erreichen. Der Nachteil von *getChild()* ist, dass es nur das erste gesuchte Kindeselement liefert. Somit muss bekannt sein, dass kein weiteres Kindeselement mit dem gleichen Namen oder Namespace in dem XML-Dokument vorkommt. Es ist nur von den Metadatenelementen bekannt, dass sie nicht wiederholend auftreten. Jedes Projekt hat die gleichen Metadaten und diese können nicht mehrfach vorkommen. Alle Metadaten werden mit den *getChild()*- und *getChildText()*-Methoden ermittelt und in eine für die Applikation verarbeitbare Form weitergeleitet.

Eine exemplarische Nutzung kann diesem Beispiel entnommen werden:

```
SAXBuilder parser = new SAXBuilder();
Document doc = parser.build("filename.xml");
//Text of root-elements child
String first = doc.getRootElement().getChildText("elementname");
//Text of child of root-elements child
String second =
getRootElement().getChild("elementname").getChildText("elementname");
```

Für das Ermitteln der Textinhalte gibt es neben *getChildText()* noch *getChildTextTrim()* und *getChildTextNormalize()*. Der Unterschied liegt darin, wieviel von "white space[3]" aus dem XML-Dokument übernommen wird.

- getChildText(): übernimmt den Textinhalt, wie er in dem XML- Dokument vorliegt
- getChildTextTrim(): übernimmt nicht den vorausgehenden und nachfolgenden "white space" vom Textinhalt
- getChildTextNormalize(): verhält sich wie getChildTextTrim(), wobei alles zu einem Bereich zusammengefügt wird

Soll der Text des Elementes ermittelt werden, kann die Methode *getText()* genutzt werden. Diese Methode gibt es auch in der Version für getrimmten und normalisierten Text.

Ist es nicht bekannt, wie oft die Elemente im XML-Dokument vorkommen, ist eine rekursive Navigation durch das komplette Dokument die einzige Option. Dazu wird die *getContent()*-Methode benutzt. Diese gibt eine Liste aller Kinder[4] des Elementes und nicht nur Elemente wieder. Indem man *getContent()* auf Kindeselementen anwendet, kann eine Ebene tiefer in der Elementstruktur des Dokumentes nach weiteren Knoten gesucht werden. Es wird *getContent()* rekursiv angewendet, damit durch das gesamte Dokument navigiert werden kann. Um welchen Knotentyp es sich gerade handelt, muss ermittelt werden. Erst nach dessen Identifizierung kann man entsprechende Anweisungen ausführen lassen. Die Identifizierung geschieht mit Hilfe des *instanceof*-Operator.

[3] darunter zählen u. a. Leerzeichen, Zeilenumbrüche
[4] z. B. Textknoten, Kommentare, Anweisungen usw.

Üblicherweise wird hierfür eine Methode erstellt, bei der das Startelement als Parameter übergeben wird. Von diesem Element ausgehend ruft sich die Methode für jedes weiter gefundene Kindeselement neu auf.

```
public static void navigate(Element element) {

    List content = element.getContent();
    Iterator iterator = content.iterator();
    while (iterator.hasNext()) {
        Object o = iterator.next();
        if (o instanceof Element) {
            Element child = (Element) o;
            navigate(child);
        }
    }

}
```

Auf diesem Grundgerüst baut sich dann die jeweils spezifische "Suchmethode" auf. Im Fall der Applikation liegt das Interesse im Finden des jeweils zugehörigen Textes von Elementen und dessen Zwischenspeicherung. Ist ein solcher Textknoten gefunden, können die abzuarbeitenden Anweisungen erteilt werden.

```
if (o instanceof Text) {//assignments}
```

Die Klasse *Jdom* weist zwei solcher "Suchmethoden" auf, um XML-Dokumente auf diese Weise nach bestimmten Textknoten zu durchsuchen sowie den gefundenen Text zwischenzuspeichern. Eine der Methoden ist *public void loadProjectList(Element element) {}*, deren Aufgabe darin besteht, das XML-Dokument mit der Sammlung aller vorhandenen Projekte auszulesen. Mit den zugehörigen *get*-Methoden kann auf den zwischengespeicherten Text zugegriffen werden.

- `public String[] getProjectListName() {}`

  Die Größe des Feldes entspricht der Anzahl der gefundenen Projekte in dem XML-Dokument. Die Reihenfolge der eingetragenen Medien in dem Projekt-XML-Dokument ist von oben nach unten eingelesen.

- `public String[] getProjectListIcon() {}`

  Das Feld gleicht dem von *getProjectListName()*, es enthält nur die Namen der Bilddateien. Dieses Bild erscheint dann bei der Applikation als Icon in der Projektliste. Ebenso ist die Position des Bildnamens im Feld dieselbe wie der dazugehörige Projektname bei *getProjectListName()*.

Bei der anderen Methode handelt sich es um *public void loadProjectContent(Element element) {}*, welche bei dem jeweiligen XML-Dokument eines Projektes angewendet wird. Die zugehörigen *get*-Methoden sind:

- `public String[] getProjectListMedia() {}`

  Dieses Feld liefert die Vorschaubildnamen, welche zur Darstellung der Medienliste genutzt werden. Die Reihenfolge entspricht den von oben nach unten eingetragenen Medien aus dem Projekt-XML-Dokument.

- `public String[] getProjectMedia() {}`

  Das Feld beinhaltet die Mediennamen. Falls sich der Benutzer entscheidet, ein Medium genauer zu betrachten, wird der zugehörige Medienname zu dessen Darstellung genutzt.

- `public String getProjectContent() {}`

  Der String beinhaltet den zugehörigen Text des Projektes, welcher bei der Projektöffnung dargestellt wird.

Das Zwischenspeichern der durch das Parsen ermittelten Daten geschieht in Feld-Listen vom Typ String. Lediglich für den textuellen Inhalt eines Projektes wird ein String-Buffer benutzt. Die Umwandlung der Listen in ein Feld geschieht bei dem Aufruf der zugehörigen *get*-Methode, gleiches gilt für die Umwandlung des String-Buffers zu einem String[5]. Bei der Erstellung des Feldes aus einer Liste wird eine *for each*[6]-Schleife angewandt.

```
String[] array = new String[arrayList.size()];
int i = 0;
for (String a : arrayList) {

    array[i] = a;
    i++;

}
```

[5] geschieht einfach mit *toString() bei Zuweisung*
[6] auch enhanced-loop genannt

6.3 Nutzung der Entwicklungsumgebung MT4j

Eine Applikation, welche mit der Entwicklungsumgebung *MT4j* erstellt wird, besteht immer aus mindestens einer Szene. Die Darstellung der Szene wird durch die Nutzung der abstrakten Klasse *AbstractScene* ermöglicht. Diese liefert alle wichtigen Grundfunktionen, wenn sich die eigene Klasse von ihr ableitet. Somit leitet sich die Klasse *MainScene* von *AbstractScene* ab, um Zugriff zu erhalten.

Damit eine Szene benutzt werden kann, müssen sich zwei Methoden in der Klasse befinden. Wenn eine direkte Benutzung nicht vonnöten ist, müssen sie mindestens definiert werden und brauchen dann keine Anweisungen zu beinhalten.

public void init()

Diese Methode wird bei jeder Szenenänderung aufgerufen. Präziser ausgedrückt: Sie wird vor der jeweiligen Szene, welche sich ändert, als die gerade aktive Szene aufgerufen. Die Methode kann u.a. dazu benutzt werden, bestimmte Einstellungen vornehmen zu lassen. Im Fall der eigenen Applikation wird sie zur Registrierung des Listeners für die Tastatur genutzt.

public void shutDown()

Der Aufruf dieser Methode erfolgt durch die aktuelle Szene, wenn eine Änderung zu einer neuen Szene bevorsteht. Da die eigene Applikation einen Listener für die Tastatur in der Methode *init()* registriert, muss dieser hier wieder abgemeldet werden.

6.3.1 Aufbau der Klasse Mainscene

Der Konstruktor der Klasse weist zwei Parameter auf, die *MTApplication* und den Szenennamen. Innerhalb des Konstruktors werden die wichtigen Einstellungen der Szenen vorgenommen. Auch dient der Konstruktor zur Initialisierung und Definition aller Szenekomponenten.

Die Entwicklungsumgebung benutzt *OpenGL* zur beschleunigten Darstellung der grafischen Elemente und Effekte. Ohne Unterstützung wird die Applikation schlecht bis gar nicht laufen. Während eine Szene aufgebaut wird, sollte auf *OpenGL*-Unterstützung geprüft werden. Ist das nicht der Fall, wird eine entsprechende Fehlermeldung ausgegeben.

```
if (!MT4jSettings.getInstance().isOpenGlMode()) {
    System.err.println(this.getName() + ïs only usable with
    the OpenGL renderer.");
}
```

Weitere Einstellungen sind:

- Löschfarbe bei Änderung einer Szene
- Registrierung einer Anzeige für den Cursor[7]
- Verhinderung von Anzeigefehlern[8]
- Einstellung der Hintergrundfarbe der Szene

Nach den Szeneeinstellungen folgt die Erstellung der darauffolgenden sichtbaren Komponenten, welche später näher behandelt werden. Neben dem Konstruktor, *init()* und *shutDown()* gibt es noch einige Methoden, welche die Klasse komplettieren.

- `public void keyEvent(KeyEvent e) {}`

  Mit dieser Methode wird der Listener für die Tastatureingaben erstellt. Wenn er registriert wurde, reagiert die Applikation auf Eingaben. In der Methode wird definiert, auf welche Tasten und wie reagiert werden soll.

- `private void addTapProcessor(MTListCell cell) {}`

  Dies ist ein weiterer Listener, der auf bestimmte Eingaben innerhalb der Projektliste reagiert und entsprechende Szeneänderungen bewirkt.

[7] dient für den Benutzer als visuelle Bestätigung
[8] Applikation ist nur 2D, Entwicklungsumgebung ermöglicht 3D und ohne Angabe der Tiefe kann es zu Fehlern kommen

- `private void addTapProcessorMedia(MTListCell cell) {}`

  Der Listener für die Medienliste, dessen Funktionsweise dem der Projektliste ähnlich ist, wird hier definiert.

- `private void fillMetaField(String projectXML) {}`

  Die Methode sorgt bei ihrem Aufruf für das Füllen der Metadatenkomponente, indem die Metadaten aus dem ausgewählten Projekt-XML-Dokument ausgelesen werden.

- `private void fillLabelField(String cellName, int clickedCell, int maxCell) {}`

  Bei Aufruf der Methode wird die Labelkomponente mit Daten gefüllt.

- `private void createMediaContent() {}`

  Vorhandene Mediendaten des Projektes werden in einer Medienliste zusammengestellt.

- `private String getPathToIcons() {}`

  Diese Hilfsmethode liefert den kompletten Pfad, um die Übersicht im Quellcode zu erleichtern. Sonst müsste für jedes Laden einer Bilddatei der Pfad angegeben werden.

- `private String getPathToXML() {}`

  Eine weitere Hilfsmethode, welche das Liefern des XML-Dokumentpfades erleichtert, dient zur Quellcodevereinfachung.

6.3.2 Die Komponenten der Applikation

Wie schon vorher angesprochen, besteht eine Applikation aus mindestens einer Szene. Jede Szene enthält ihren eigenen "canvas", einen sogenannten Szenegraphen. Wie bei jedem grafischen Benutzerinterface fügen sich die zu zeichnenden Komponenten in solch eine hierarchisch aufgebaute Struktur zusammen. Jede Komponente kann dabei beliebig ihr zugeordnete Kindeselemente besitzen, aber nur eine Elternkomponente. Die Klasse *MTComponent* der Entwicklungsumgebung bildet die Basisklasse aller Komponenten. Diese Komponenten sind nicht sichtbar. Indem ihre Methode zum Zeichnen überschrieben wird, kann mittels *processing's* Syntax und Methoden zum Zeichnen Sichtbares erstellt werden. Die eigene Applikation wird selber nur auf schon fertige geometrische Grundformen zurückgreifen, die sich von *MTComponent* ableiten.

Bei der Zuordnung von Komponenten in eine Szene muss auch dessen Position geklärt werden. Da es sich um eine 3D-Umgebung handelt, gibt es 3 Koordinaten.

Des Weiteren gibt es nicht nur einen Koordinatenraum, sondern insgesamt drei an der Zahl. Einen kurzen Überblick über deren Funktionsweise soll ihr Zusammenspiel aufzeigen.

Global Space

Dies ist die Umgebung, in der die Komponenten aus der Sicht der Kamera[9] erscheinen. Teilen sich alle Komponenten denselben Vorfahren[10], so teilen sie sich auch denselben globalen Koordinatenraum.

Local Space

Darunter versteht sich des Komponenten eigenes dreidimensionales Koordinatensystem ohne weitere Transformationen[11]. Dimensionen, Punkte und Positionen einer erstellten Komponente sind immer relativ zu diesem lokalen Koordinatensystem und somit zu dem komponenteneigenen lokalen Ursprung.

Parent-Relative Space

Wird die komponenteneigene lokale Transformation zum lokalen Koordinatenraum hinzugefügt, spricht man vom *Parent-Realtive Space* der Komponente.

Um einen groben Überblick über die nötigen Schritte zur Zeichnung einer Komponente aufzuzeigen, dient folgende Transformationsabfolge[12] von der Entwicklungsumgebung:

- Zeichenbefehle der Komponente (nicht transformierter lokaler Koordinatenraum)
- Hinzufügen der komponenteneigenen lokalen Transformation zu dem, was gezeichnet werden soll (*Parent-Realtive Space*)
- Hinzufügen aller elterneigenen Transformationen zum ”canvas” der Szene (globaler Koordinatenraum)
- Hinzufügen der Kamera und Transformation der Perspektive (Bildschirm sichtbarer Raum)

[9] unter Kamerasicht versteht sich der Beobachtungswinkel des Benutzers auf die aktuelle Szene
[10] bei einer Szene üblich, da alle zum ”canvas” der Szene gehören
[11] translate (bewegen), skalieren, rotieren
[12] auch Transformation Chain genannt

Die benutzten Komponententypen benötigen verschiedene Transformationen bei der Erstellung und während ihrer Positionierung. Liegt ein solcher Fall vor, so wird darauf gesondert eingegangen.

6.3.2.1 Erstellung von Schrift

Da es sich bei der Entwicklungsumgebung um eine 3D-fähige Umgebung handelt, werden Fonts und damit auch der Text nicht direkt gerendert. Es gibt zwei zur Verfügung stehende Methoden, um dieses zu bewerkstelligen. Dabei handelt es sich einmal um Fonts auf Basis von Vektoren/Geometrie oder auf Basis von Bitmap/Texturen.

Vector Fonts

Vektor Fonts werden mit Hilfe geometrischer Primitiven[13] erstellt. Dadurch lassen sich folgende Eigenschaften bezüglich der Vor- und Nachteile abbilden:

Tabelle 6.1: Vor- und Nachteile von Vektor-basierter Schrift

Vorteile	**Nachteile**
Größe des Fonts hat nur geringen Einfluss auf dessen Speicherauslastung	allgemeine Speicherauslastung, um geometrische Informationen zu speichern
Rotation, Skalierung, Bewegung von Text ohne Einbüßen der Textschärfe	lange Textpassagen erhöht deren Zeit zum rendern
Modifizierung[14] von Text	lange Ladezeit zur Erstellung und Speicherung[15]
Hinzufügen von Kollisionserkennung bei Textbuchstaben	anti-aliasing sorgt für Artefaktbildung bei zu kleiner Schriftgröße

Bitmap Fonts

Für jeden Schriftbuchstaben wird ein Bild erstellt, um es dann auf einem Viereck als Textur abzubilden.

Tabelle 6.2: Vor- und Nachteile von Bitmap-basierter Schrift

Vorteile	**Nachteile**
durch Rendern von nur einem Viereck je Buchstabe ergibt sich eine gute Performance	Skalierung oder Rotation des Textes kann zu unerwünschten Fehlern führen
gute Lesbarkeit bei kleinen Schriftgrößen	große Schriftgrößen führen zu starker Auslastung des Texturspeichers

[13] Bezeichnung für einfache geometrische Grundformen wie Linien, Dreiecke
[14] Geometrieabänderung der Buchstaben
[15] am besten während Applikationsstart

Das Laden einer Schrift wird mit der Klasse *FontManager* und deren Methode *createFont()* realisiert. Beim Laden der Schrift kann die Schrift, Schriftgröße, Farbe und anti-aliasing bestimmt werden. Nachstehende Schriften werden unterstützt:

- alle zu findenden Systemschriften durch Java (Umsetzung als Bitmap-Schrift)
- *True Type Fonts (*.ttf)* aus einer Datei (Umsetzung als Vektor-Schrift, Bitmap-Schrift, Standardvektor)
- *Scaleable Vector Graphics (*.svg)* Schriftdateien (Umsetzung als Vektor-Schrift)

Um eine Vektor-Schrift erstellen zu können, muss die **.ttf*- oder **.svg*-Datei im /data oder /src/data Ordner vorliegen.

```
FontManager.getInstance().createFont(getMTApplication(),
"arial.ttf", 24);
```

In der Applikation wird eine weiße Schriftfarbe genutzt, die Schrift selber wird aus der Datei calibri.ttf bezogen.

```
IFont fontCalibri =
    FontManager.getInstance().createFont(getMTApplication(),
    "calibri.ttf",
        50, //Font size
        new MTColor(255,255,255), //Font fill color
        new MTColor(255,255,255)); //Font outline color
```

Der Einsatz von Schrift im Bereich des Labels erfordert eine kleine Schriftgröße. Da solche Schriftgrößen auf Vektorbasis unschöne Effekte hervorrufen, wird die Nutzung der Bitmap-Schrift erzwungen. Damit das Erstellen einer Bitmap-Schrift aus einer **.ttf*-Datei funktioniert, muss vorher dies geschehen:

```
FontManager.getInstance().registerFontFactory(".ttf", new
BitmapFontFactory());
```

6.3.2.2 Textkomponenten

Nachdem eine Schrift erfolgreich erstellt wurde, fehlt nur noch der Bereich, in welchem sie erscheint. Genauer gesagt muss die Schrift einer Komponente zugeordnet werden, welche die Fähigkeit hat, Schrift zu rendern. Die Entwicklungsumgebung beinhaltet zwei solcher Komponenten.

Text Field

Durch Nutzung der *MTTextField* Klasse wird die Darstellung von einzeiligem Text ermöglicht. Zeilenumbrüche werden ignoriert. Sollte der Text den Bereich des Textfeldes überschreiten, so wird der Text linksführend gescrollt.

```
MTTextField(float x, float y, float width, float height, IFont
font, PApplet applet);
```

Text Area

Benötigt man das Rendern von mehrzeiliger Schrift, so leistet die Klasse *MTTextArea* hierfür Hilfe.

```
MTTextArea(float x, float y, float width, float height, IFont
font, processing.core.PApplet applet);
```

6.3.2.3 Geometrische Komponenten

Der Aufbau einiger Elemente der Applikation, wie u.a. das Metadatenfeld, wurde mit Hilfe von geometrischen Grundformen erstellt. Diese leiten sich von der Klasse *AbstractShape*[16] ab. Die Sammlung der verschiedenen Grundformen besteht aus:

- MTEllipse: zur Erstellung simpler elliptischer Formen
- MTLine: Klasse zum Zeichnen einfacher Linienabschnitte
- MTPolygon: Repräsentiert ein ebenes, konvexes Polygon
- MTRectangle: zur Erstellung eines einfachen Vierecks
- MTRoundRectangle: Darstellung eines Vierecks mit abgerundeten Ecken
- MTCube: zur Erstellung eines Würfels (3D)
- MTSphere: Klasse zum Erstellen kugelförmiger Körper (3D)
- MTTriangleMesh: Klasse zum Zeichnen von Verbänden aus Dreiecken (3D)
- MTTriangle: Dreiecksklasse, welche als Verbund genutzt wird (3D)

Weiterführend erlangt man Zugriff auf nachstehende Funktionen, die jede Grundform mit sich bringt:

- Einstellung der Position
- Einstellung der Erscheinung und des Stils
- *get*-Methoden, zur Ermittlung des Mittelpunktes und der grundformeigenen Dimension
- Verwaltung der Vertices
- *Bounding Shape*[17] für Sichtbarkeit, Mittelpunkterrechnung, "Greifen", Ausdehnung
- OpenGL Hardware Beschleunigung
- Einstellung von Standard Multi-Touch Eingabeverarbeiter

[16] weiterführend würde man von AbstractVisibleComponent zu MTComponent gelangen!
[17] allgemein weniger komplexe Struktur, als die der echten Struktur und wird aus Performancegründen oft genutzt

Indem ein Objekt mittels einer der Grundklassen erzeugt wird, kann eine geometrische Grundkomponente schnell erzeugt werden.

```
MTRectangle rectangle = new MTRectangle(float x, float y, float
width, float height, processing.core.PApplet applet);
```

Erstellt man ein Feld aus mehreren Vertices und übergibt sie einem Objekt von *MTPolygon*, baut man sich sein eigenes Objekt. Des Weiteren ist es möglich, für jeden Vertex eine Farbe zuzuordnen[18]. Ein geschlossenes Objekt wird erzielt, indem der erste definierte Vertex im Feld ebenso noch als letzter vorkommt[19]. Im Abschnitt *Erzeugung der MTCell* wird diese Variante benutzt. Ein einfaches Beispiel für ein Rechteck mit der Kantenlänge 10 wird hier gegeben:

```
Vertex[] vertices = new Vertex[] {
    new Vertex(0, 0);
    new Vertex(0, 10);
    new Vertex(10, 10);
    new Vertex(10, 0);
    new Vertex(0, 0);
};
```

Nachdem das Feld mit dem Vertex für das Rechteck erstellt wurde, muss noch die Zuordnung zu einem Polygonobjekt geschehen:

```
MTPolygon p = new MTPolygon(vertices, getMTApplication());
```

Eine Reihe von Methoden können benutzt werden, um das Aussehen, Verhalten oder Anderes abzuändern. Dazu muss nur über das erstellte Objekt die betreffende Methode aufgerufen werden, der aufzurufende Werttyp ist der API zu entnehmen. Die Applikation nutzt im Allgemeinen anstatt einer Füllfarbe eigens erstellte Texturen für die geometrischen Objekte. Dazu muss zuerst ein *PImage*-Objekt aus der Processing Umgebung erstellt werden, welches die Bilddatei lädt. Danach kann das *PImage*-Objekt dem geometrischen Objekt als Textur zugewiesen werden.

```
MTRectangle rectangle = new MTRectangle(0, 0, 10, 10,
getMTApplication());
PImage image = mtApplication.loadImage("Image.jpg");
rectangle.setTexture(image);
```

[18] bei unterschiedlichen Farben erreicht man einen Gradient-Effekt
[19] damit ein Vektor auch zwischen Start- und Endpunkt entsteht

Die Positionierung der geometrischen Objekte in einer Szene geschieht mit Hilfe derer Mittelpunkte. Genauer handelt es sich um den Mittelpunkt, welcher durch eine *Bounding Shape* erstellt wurde. Dies sollte bei einer Positionierung bedacht werden. Die Positionierungsmethoden funktionieren nach dem am Anfang des Abschnittes *6.3.2* erläuterten Prinzip der Koordinatenräume.

```
public void setPositionRelativeToParent(Vector3D pos) {}
public void setPositionGlobal(Vector3D pos) {}
```

6.3.2.4 Erstellung einer Liste mit Hilfe von MTList und MTListCell

Bei der *MTList* handelt es sich um eine Listenkomponente, die sich vertikal aufbaut. Indem man ihr mehrere *MTListCell* Komponenten zuweist, wird eine Liste mit mehreren Elementen erstellt. Die Applikation erstellt zwei solcher Listen, einmal als Vorschaufunktion aller zur Verfügung stehenden Projekte und zum anderen zur Vorschau aller gefundenen Medien eines Projektes.

Die Listenerstellung muss unter Berücksichtigung der späteren Rotation geschehen, da die Applikation horizontale Vorschaulisten nutzt. Die Reihenfolge der Erstellung kann dieser Abfolge entnommen werden:

- Erstellung der Liste:
  Die Größe: Iconhöhe definiert Listenbreite, Iconbreite * sichtbare Zellenanzahl definiert Listenhöhe
  Einstellung weiterer Eigenschaften
- Parsen des jeweiligen XML-Dokumentes, deren Daten angezeigt werden
- Bestimmung der Zellenanzahl
- Erstellung eines Vierecks, welches die Bilddatei als Textur lädt und dem jeweiligen Zellenelement zugeordnet wird (je Zelle)
- Zellenelement der Liste als Element zuordnen
- Listener für Benutzereingabe dem Zellenelement zuordnen
- Liste der Szene übergeben
- Rotation der Liste vornehmen
- Globale Positionierung der Liste

Eine Liste wird mit Hilfe der Klasse *MTList* erstellt. Die Definition der Größe der Liste ist einerseits von der Anzahl der gleichzeitig anzuzeigenden Zellen, wie auch von deren Ausrichtung abhängig. Da die Liste für die Applikation horizontal ist, wird die Zellenanzahl bei der Listenhöhe berücksichtigt! Das heißt, dass nach der Rotation die alte Listenbreite die neue Listenhöhe ist, die alte Listenhöhe ist die neue Listenbreite. Da auch die Größe der jeweiligen Vorschaubilder (Icons) eine Rolle spielt, definiert deren Größe auch die Dimension der Liste. Die Applikation nutzt Listen, deren Größe die gleichzeitige Anzeige von 5 Zellenelementen (Vorschauicons) zulässt.

```
int listWidth = iconHeight;
int listHeight = iconWidth;
MTList list = new MTList(0, 0, listWidth, listHeight * 5,
getMTApplication());
```

Die Anzahl der Zellenelemente in den Vorschaulisten ist abhängig von der gefundenen Projektanzahl bzw. der gefundenen Medien. Dazu wird die Länge des betreffenden Feldes der *get*-Methode aus der Klasse *Jdom* genutzt. Die Erstellung eines Zellenelementes, dessen Zuordnung zur Liste sowie dessen Listener für die Benutzereingabe getätigt wird, geschieht in einer Zählschleife. Die Endbedingung ist dabei die Länge des besagten Feldes. Kommendes Beispiel zeigt die Erstellung eines Zellenelementes für die Projektliste:

```
PImage icon = getMTApplication().loadImage(this.getPathToIcons() +
dom.getProjectListIcon()[index]);
Vertex[] vertices = new Vertex[] {

    //creating vertex for a rectangle, size of loaded icon

}
MTListCell cell = new MTListCell(iconHeight, iconWidth,
getMTApplication());
MTPolygon p = new MTPolygon(vertices, getMTApplication());
p.setTexture(icon);
cell.addChild(p);
list.addListElement(index, cell);
addTapProcessor(cell);
```

Erst nachdem alle Zellenelemente erzeugt wurden, kann die Liste der Szene übergeben werden.

```
this.getCanvas().addChild(projectList);
```

Nun, nachdem die Liste mit ihren Zellenelementen der Szene zugeordnet wurde, kann die Rotation und globale Positionierung erfolgen.

```
list.rotateZ(list.getCenterPointLocal(), -90,
TransformSpace.LOCAL);
list.setPositionGlobal(new Vector3D(getMTApplication().getWidth()
- desiredPosition, getMTApplication().getHeight() -
desiredPosition));
```

6.3.3 Die Listener zur Erkennung von Benutzereingaben

Standardgemäß besitzen Komponenten, welche sich aus der Klasse *AbstractShape* ableiten, schon aktivierte Listener. Diese sind sogenannte Standard-Listener für die Eingabeverarbeitung, die für folgende Eingaben schon erkennbare Gesten nutzen:

- Drag
- Skalierung
- Rotation

Möchte man die Standard-Listener für Eingabeverarbeitung von einer Komponente entfernen, so muss man die Registrierung rückgängig machen. Man kann auch alle bisherigen Aktiven deaktivieren. Dies wird in der Applikation bei Komponenten vorgenommen, welche nie ihre Position, Größe oder Ausrichtung ändern sollen. Sie sollen auch auf keine anderen Eingabevarianten reagieren, wie es z.B. beim Metadatenfeld der Fall ist. Solche Komponenten dienen nur zur Darstellung von Informationen.

```
//Variante zur Deaktivierung einer Eingabevariante
component.unregisterInputProcessor(inputProcessor);
//Deaktivierung aller bisher aktivierten Eingabevarianten
component.unregisterAllInputProcessors();
```

Um eine Komponente Multi-Touch-Gesten bekannt zu machen, muss ein Komponenteneingabeverarbeiter (Listener) für die jeweilige Komponente registriert werden. Dieser verarbeitet alle bei der Komponente auftretenden Eingaben. Bei der Verarbeitung und Analyse kann entsprechend bestimmter Muster oder Verhalten vorgegangen werden. Wenn eine Multi-Touch-Geste vorliegt, wird ein Gestenereignis zur Komponente zurückgesendet.

7 Fazit

Die ersten Kapitel legten den Rahmen der Arbeit fest und klärten über die theoretischen Grundlagen auf. Nachdem ein eigenes Konzept auf deren Basis erstellt und umgesetzt wurde, wird in den nächsten Unterkapiteln auf deren Auswertung eingegangen. Es wird eine allgemeine Beurteilung über Nutzung von gestenbasierter Interaktion geleistet. Danach werden die erzielten Ergebnisse mit Hilfe des Frameworks *MT4j* betrachtet und auf entstandene Komplikationen eingegangen. Das Parsen von XML- Dokumenten erhält eine eigene und kurze Bewertung, da es als Hilfsmittel zum Auslesen von Informationen genutzt wurde. Als Abschluss erfolgt ein Ausblick zu zukünftigen und aktuellen Nutzungen oder möglichen Änderungen.

Einige der in der Konzeptphase umzusetzenden Punkte, die sowohl das Design wie auch die zu nutzenden Gestenformen betreffen, konnten nicht komplett berücksichtigt werden. So wurde auf die Verwendung eines Startbildschirmes und des Ruhezustandes verzichtet, da sie sich einerseits für eine prototypische Applikation als zu weitreichende Funktionen herausstellten und andererseits für den eigentlichen Funktionsumfang der Applikation wenig beitrugen. Eine weitere Einsparung wurde bei einer optionalen Geste, der *Spin to Scroll* Geste, für das Navigieren durch die Listen getätigt. Die Erstellung eigener Gestenformen, deren Erkennungseinstellung und Integration in das genutzte Framework hätte einen zu langwierigen Umsetzungsprozess hervorgerufen. Somit zeigt sich, dass während der Konzeptionsphase nicht alle gedachten Ansätze umgesetzt werden konnten.

7.1 Auswertung der gestenbasierten Interaktion

Ein wichtiger Aspekt, welcher bei der Konzeption und Umsetzung gestenbasierter Interaktion bedacht werden muss, ist die Kenntnis des Benutzers über solche Interaktionsformen. Gängige Hilfsmittel wie Maus und Tastatur benötigten auch ihre Entwicklungszeit und heutige Benutzer wuchsen mit deren Einsatz auf. Genauso wird es sich bei der gestenbasierten Interaktion verhalten. Damit ein mehrheitlicher Anteil der möglichen Benutzergruppe mit dieser Interaktionsform arbeiten kann, muss eine Vereinheitlichung erstellt werden.

Die Gestaltung und Umsetzung unterschiedlicher Gestenformen sind sehr umfangreich. Daher werden gleiche Gesten weiterverwendet oder abgeändert. Es gibt bisher keine Konvention für Einsatzgebiete und Funktion der Gesten, vielmehr erstellen unterschiedliche Interessengruppen abgeänderte Gestenformen. Dass Lizenzierungen von bestimmten Gestenformen vorkommen können, ist bisher nicht absehbar. Es ist aber gänzlich ungeklärt, ob und wie man den Einsatz von Gesten patentrechtlich umsetzen kann.

Gerade im öffentlichen Umfeld muss der Benutzer schon von Weitem angesprochen werden. Er muss auf die Entfernung den Zweck des Systems und dessen Benutzung mittels gestenbasierter Interaktion erkennen können. Dazu wird im Regelfall ein Startbildschirm genutzt, welcher mit einfacher Symbolik oder auch wenigen Worten darauf Rückschluss gibt. Es können auch physische Objekte zur Informationsübermittlung einfließen.

Eine Einsatzbeschränkung liegt definitiv im Fall von Manipulation oder Eingabe von Daten vor. Zeitliche Beschränkung entsteht durch die Positionierung der Oberfläche und bildet den Kompromiss zur Lesbarkeit und angenehmen Nutzung für den Benutzer. Dies muss schon in der Konzeptionsphase bedacht werden. Immer mehr Ideen und deren Umsetzungen greifen deshalb als Unterstützung auf andere Interaktionsformen zurück. So kann mittels einer Tastatur eine bessere Eingabe von Informationen durch den Benutzer geschaffen werden. Auch sprachbasierte Interaktionen können die Bedienung mit einem Computerbasierten System komplettieren. Es ist davon auszugehen, dass die gestenbasierte Interaktion nicht nur als eigenständige Interaktionsform vorkommt, sondern vielmehr mit anderen Interaktionsformen kombiniert wird.

Alte bekannte Regelungen für das Design und die Umsetzung von grafischen Oberflächen sind nicht komplett übernehmbar. So muss die Größe interaktiver Elemente angepasst werden und bestimmte Menüführungen, die nicht sinnvoll sind, abgeändert werden. Allgemein entstehen dadurch weniger komplexe grafische Oberflächen mit schnell erkennbarer Menüführung und Zugriff zu Informationen. Darin liegt eine der großen Stärken, welche mit intuitiver gestenbasierter Interaktion intensiviert wird, und der Benutzer muss sich nicht mehr bewusst

mit der Funktionsweise des Computerbasierten Systems beschäftigen. Der Schwerpunkt verlagert sich zu dem eigentlichen Zweck, dem das Computerbasierten System dient.

7.2 Auswertung der Umsetzung mit Hilfe von MT4j

Das Framework *MT4j* ist eins von mehreren, die auf *TUIO* aufbauen. Zur Ausführung der Applikation muss nur eine Javaumgebung vorhanden sein, was eine hohe Kompatibilität mit anderen Betriebssystemen und Rechnersystemen bietet. Das gewählte Format ist ein selbstausführendes Jar-Archiv, welches sich über die Konsole starten und somit mögliche Fehlerausgaben einsehen lässt.

Es gibt keine abgeschlossenen Frameworks für die Erstellung einer Applikation mit gestenbasierter Interaktion. Vielmehr ist der Faktor der Aktualisierung des Frameworks wichtiger. Einige erhielten seit geraumer Zeit keine Erweiterung, was bei dem gewählten Framework nicht der Fall ist, da der Entwicklungsprozess weiter voranschreitet.

Eine Großzahl von üblichen Gesten ist implementiert, eigene oder spezielle müssen aber selbst kreiert werden. Da dazu auf *TUIO* direkt zugegriffen werden müsste, konnte aufgrund der Komplexität in der Bachelorarbeit nicht eingegangen werden. Alle nötigen grafischen Grundobjekte zur Erstellung der grafischen Oberfläche sind einfach umsetzbar und vorhanden. Für spezielle Nutzungsvarianten fehlten die nötigen Funktionen. So können Schriften erstellt werden, doch eine spätere Manipulation der Eigenschaften[1] ist nicht verfügbar. Objekte, welche Schriften rendern, können nur jeweils eine Schrift auf einmal darstellen, weswegen sich insgesamt die Nutzung von Text auf ein Minimum beschränkt. Von den Erstellern des Frameworks ist eine Verbesserung geplant, war aber mit der genutzten Version noch nicht umgesetzt. Eine Integration von Videodaten ist nur mit einem weiteren Framework zur Darstellung von Videos möglich, welches eine gesonderte Installation benötigt. Aufgrund dessen wurde darauf verzichtet, um die Ausführung plattformunabhängig, ohne eine nötige Installation und Einrichtung erhalten zu können.

Das erzeugte Jar-Archiv beinhaltet das komplette Framework und zusätzliche Dateien. Damit die Ausnutzung des Festplattenspeichers gering gehalten wird, wurden manuell unwichtige Dateien nachträglich entfernt. Es müssen im Ordner des Jar-Archives die *dll*- und Bibliotheks-Dateien vorliegen. Vorteilhaft ist die Settings-Datei, mit dessen Hilfe globale Applikationseinstellungen wie Auflösung und Vollbild nachträglich abänderbar sind.

[1] Größe, Farbe, Typ (Vektor oder Bitmap)

Für die erstellte prototypische Applikation wurde nur ein kleiner Teil der Möglichkeiten ausgeschöpft. Dies lässt sich schon allein darin erkennen, dass mit dem Framework 3D-Applikationen darstellbar sind, sowohl bezüglich der grafischen Objekte[2] wie auch der Erkennung von Gesten. Eine Kameraperspektive aus Sicht des Benutzers gehört auch zu einer der Funktionen, welche bei einer 2D-Applikation wenig genutzt wird, aber für 3D-Applikationen elementar ist.

7.3 Auswertung des Parsens von XML-Dokumenten mit SAX und Jdom

Die Nutzung von XML-Dokumenten erwies sich als eine gute Lösung zur Speicherung von Informationen und deren zugehörigen Mediendaten. Von den vielen Varianten des Auslesens der XML-Dokumente wurde *SAX* und dessen Zugriff mit *Jdom* genutzt, da es sich für eine Erweiterung zur Datenmanipulation und komplexeren XML-Dokumenten eignet. Ein größerer interner Speicherbedarf durch *Jdom* wird dadurch umgangen, dass jedes Projekt in einem eigenen XML-Dokument vorzufinden ist. Dadurch ist eine maximale Dokumentengröße absehbar und das Parsen während der Laufzeit der Applikation führt zu einer geringen Bearbeitungszeit. Mittels eines XML-Dokumentes zur gesamten Auflistung der ladbaren Projektdokumente wird ein einfaches Hinzufügen oder Entfernen von Projekten zugelassen.

Es ist wichtig, dass sämtliche XML-Dokumente in einer validen Form vorliegen, sonst ist deren Parsen nicht möglich und somit der gesamte Zugriff auf die Datenelemente des jeweiligen XML-Dokumentes nicht möglich. Die Applikation wird beim Parsen durch das Abfangen von möglichen Fehlern lauffähig bleiben. Die Erstellung neuer XML-Dokumente oder deren Abänderung wird nicht vorher überprüft, sondern lediglich erst während des Öffnens durch den SAX-Parser während der Laufzeit der Applikation. Es muss somit von der jeweiligen Person sichergestellt werden, welche Änderungen an den XML-Dokumenten vorgenommen werden, sodass sie noch valide sind.

[2] vorgefertigte, eigene oder aus einer externen 3D-Modelldatei geladene

7.4 Ausblick

Der Einsatz von gestenbasierter Interaktion sowie Multi-Touch werden sich in Zukunft etablieren. Schon erste unterschiedliche Testläufe zeigten eine positive Akzeptanz in Einkaufsstraßen. Weitere Ansätze verfolgen die Nutzung in anderen öffentlichen Bereichen, u.a. die Umgebung einer Bar oder ähnlichen Aufenthaltsräumen. In einer Arbeitsfeldumgebung gibt es verschiedene Ideenansätze, wovon die meisten die Idee des gemeinsamen Arbeitens verfolgen. Seit wenigen Jahren entwickeln sich auch spezialisierte Einsatzfelder und Orte, wie der Fall eines interaktiven Labortisches zeigt. Im privaten Umfeld wird der Einsatz besonders zu Unterhaltungszwecken oder im mobilen Bereich eingesetzt werden. Seit mehreren Jahren gibt es schon bekannte Umsetzungen in mobilen Geräten, deren stetige Verbesserung sich einem wichtigen Punkt nähern wird. Dieser wird sein, dass sich die gestenbasierte Interaktion als vorteilhafter zu den konventionellen Eingaben verhält, wenn es sich um eine einfache Kommunikationsschnittstelle zwischen Mensch und Computerbasiertem System handelt.

Die Ansichten der letzten Jahre verliefen von der hardwaretechnischen Umsetzung, den einzelnen Vor- und Nachteilen und neuen Gestenformen zu dem eigentlichen und wichtigen Aspekt einer jeden Technologie: Ihrem für uns Menschen nutzbringenden Einsatz und Vereinfachung des alltäglichen Lebensablaufes. So gehen aktuellere Betrachtungen auf Aspekte zur Unterstützung und Einrichtung der zwischenmenschlichen Kommunikation durch das benutzte Computerbasierte System ein. Somit ist es das Ziel, dass sich der Mensch weniger mit der Funktionsweise des Computerbasierten Systems beschäftigen muss, sondern sich auf die Nutzung zur Erfüllung der von ihm gestellten Aufgaben konzentrieren kann.

A Anhang

A.1 Skizzen zu Gestenformen

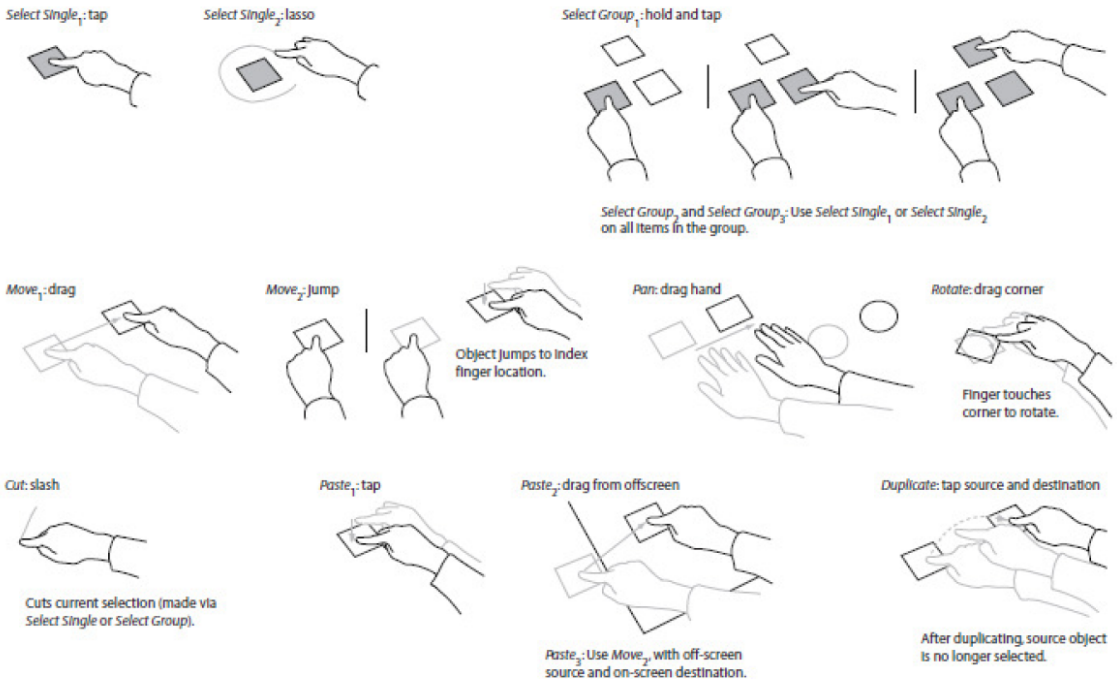

Abbildung A.1: Skizzen zu Grundformen von Gesten Teil 1, Wobbrock Seite 1089

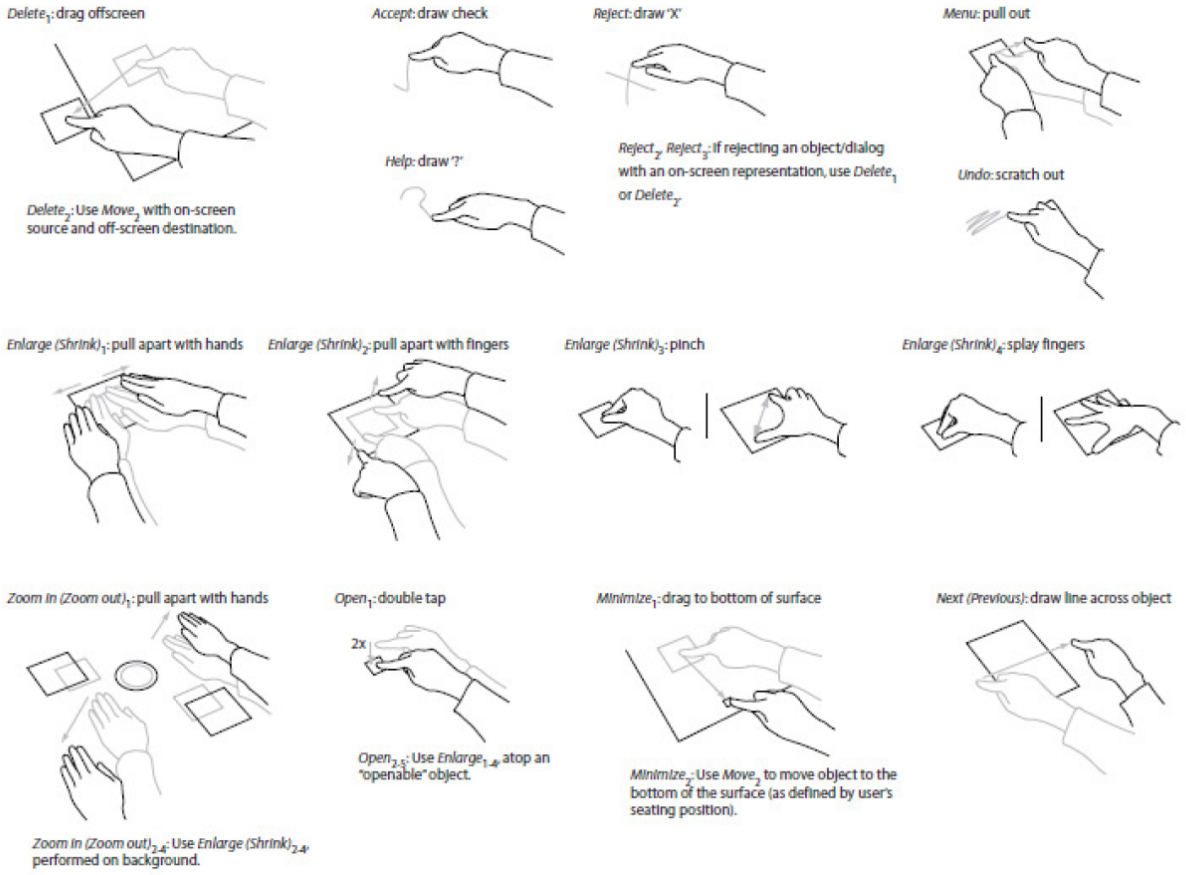

Abbildung A.2: Skizzen zu Grundformen von Gesten Teil 2

A.2 Klassendiagramme

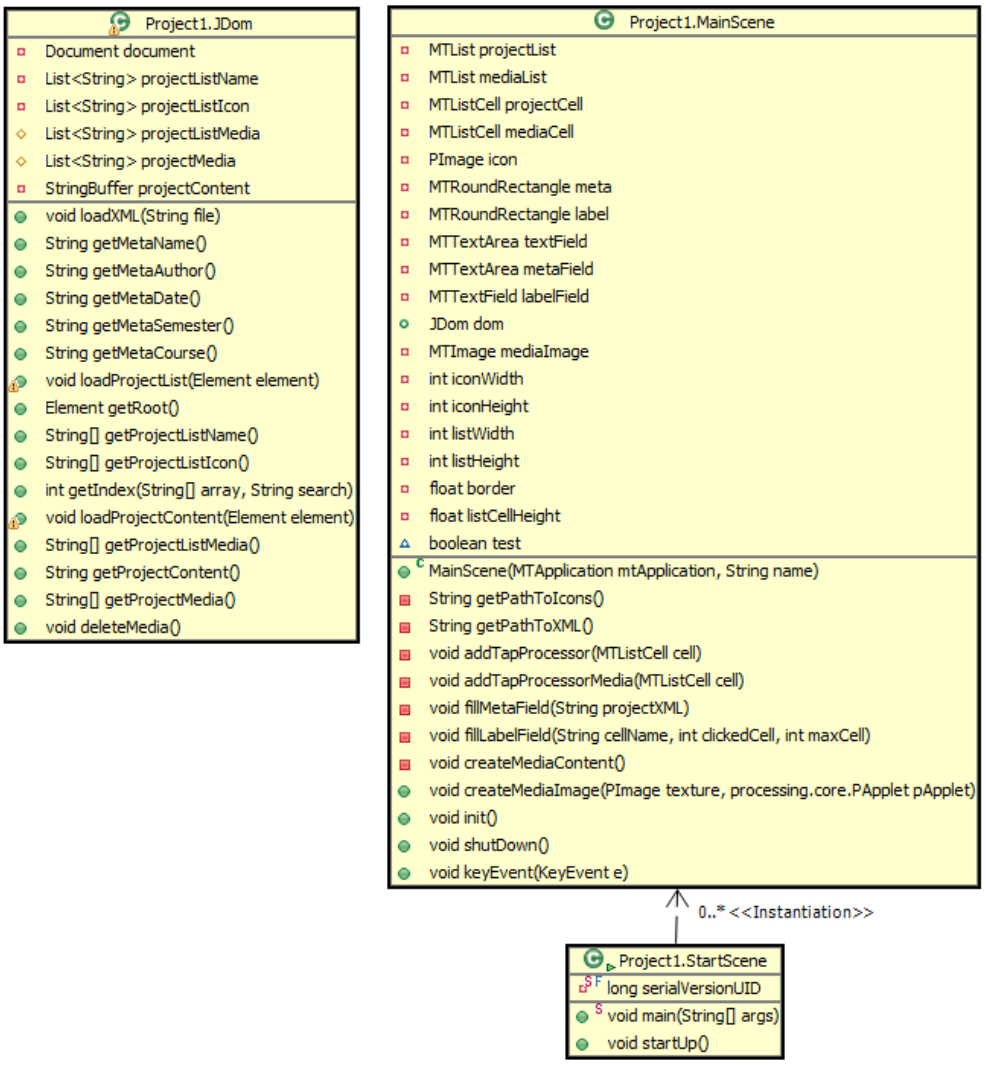

Abbildung A.3: Übersicht zu den Klassen der Applikation, Wobbrock Seite 1089

A.3 ungekürzte Sequenzdiagramme

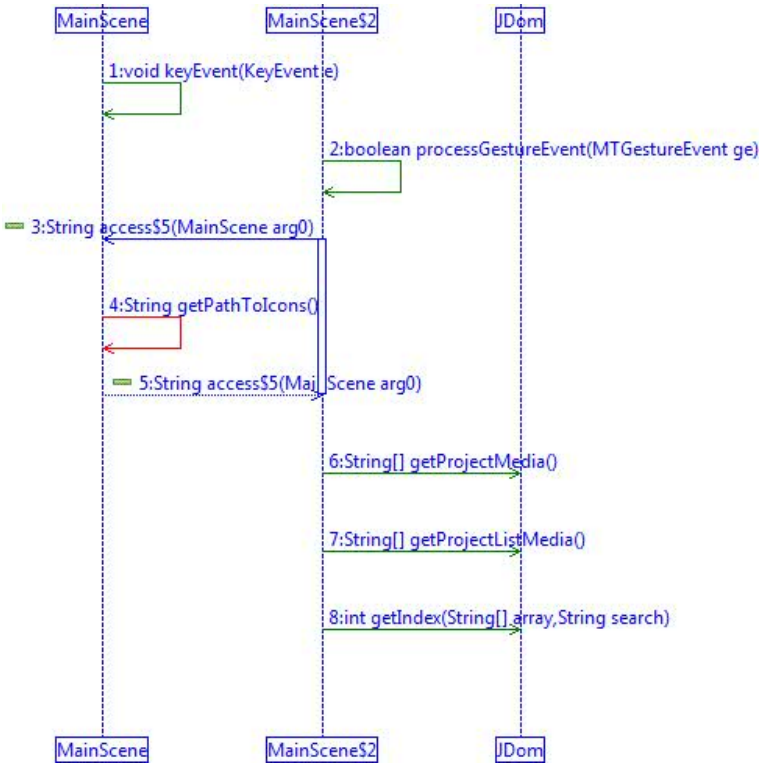

Abbildung A.4: Sequenzdiagramm: Öffnen eines Media-Dokumentes

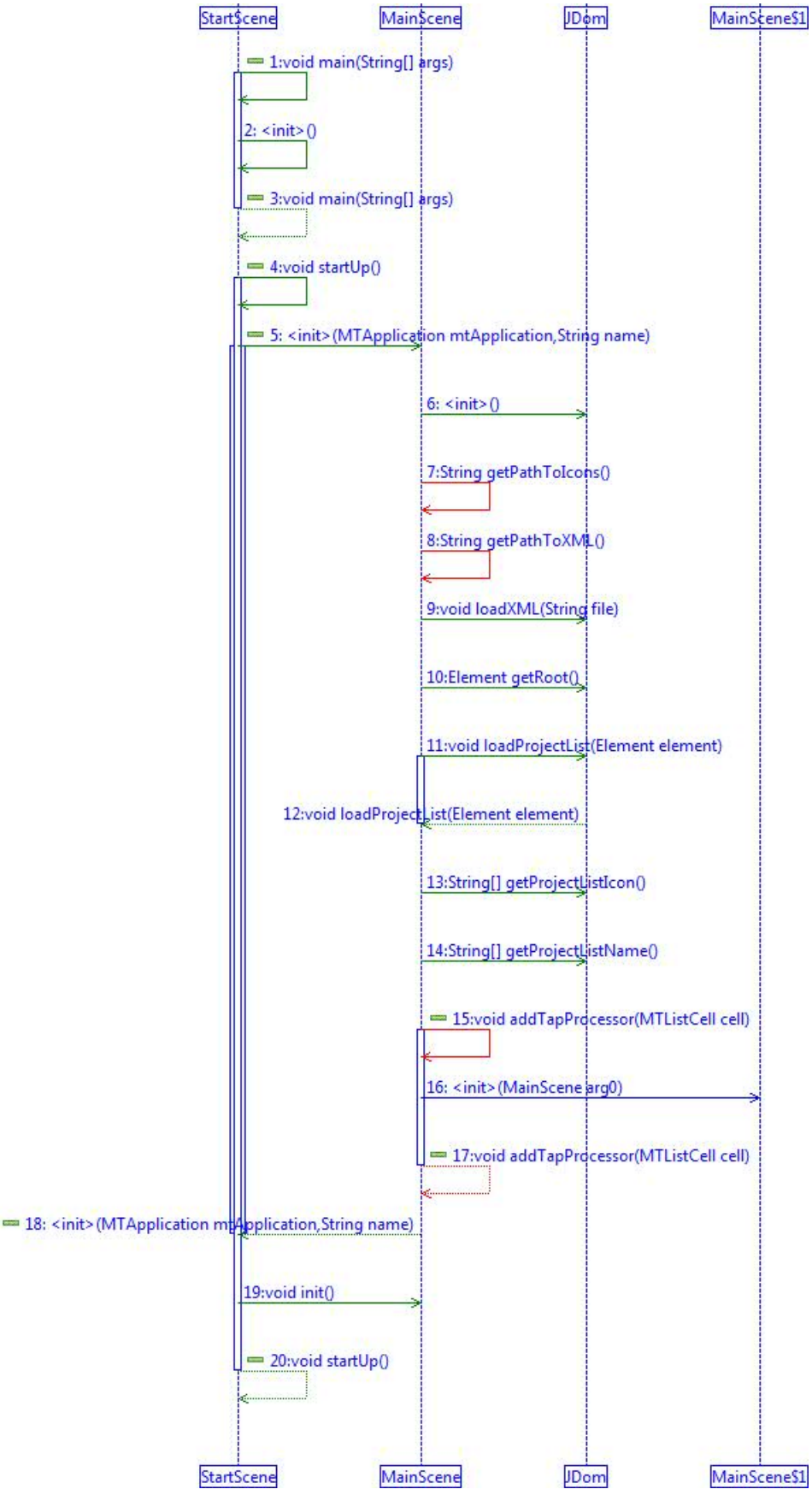

Abbildung A.5: Sequenzdiagramm: Start der Applikation

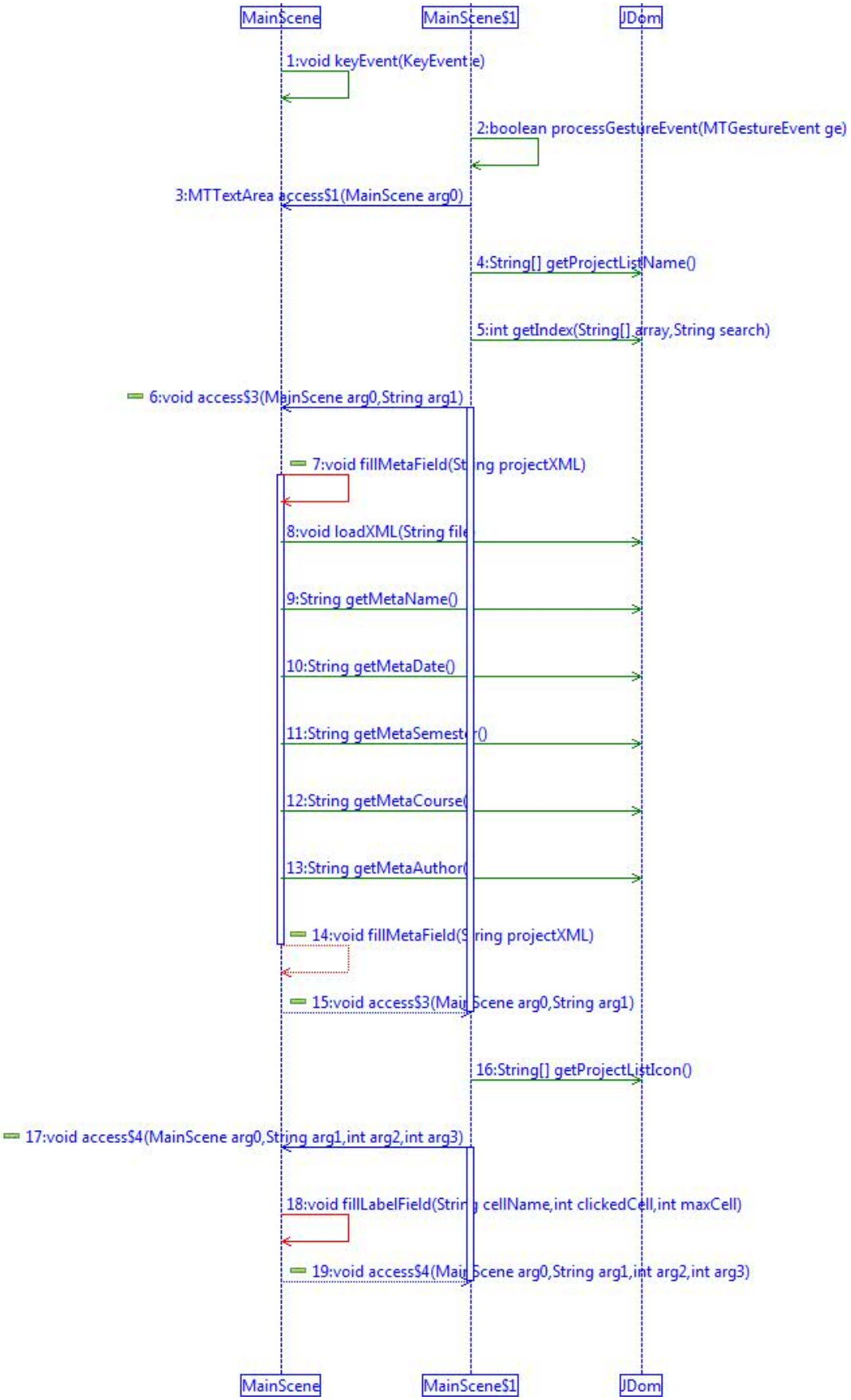

Abbildung A.6: Sequenzdiagramm: Selektion eines Projektes

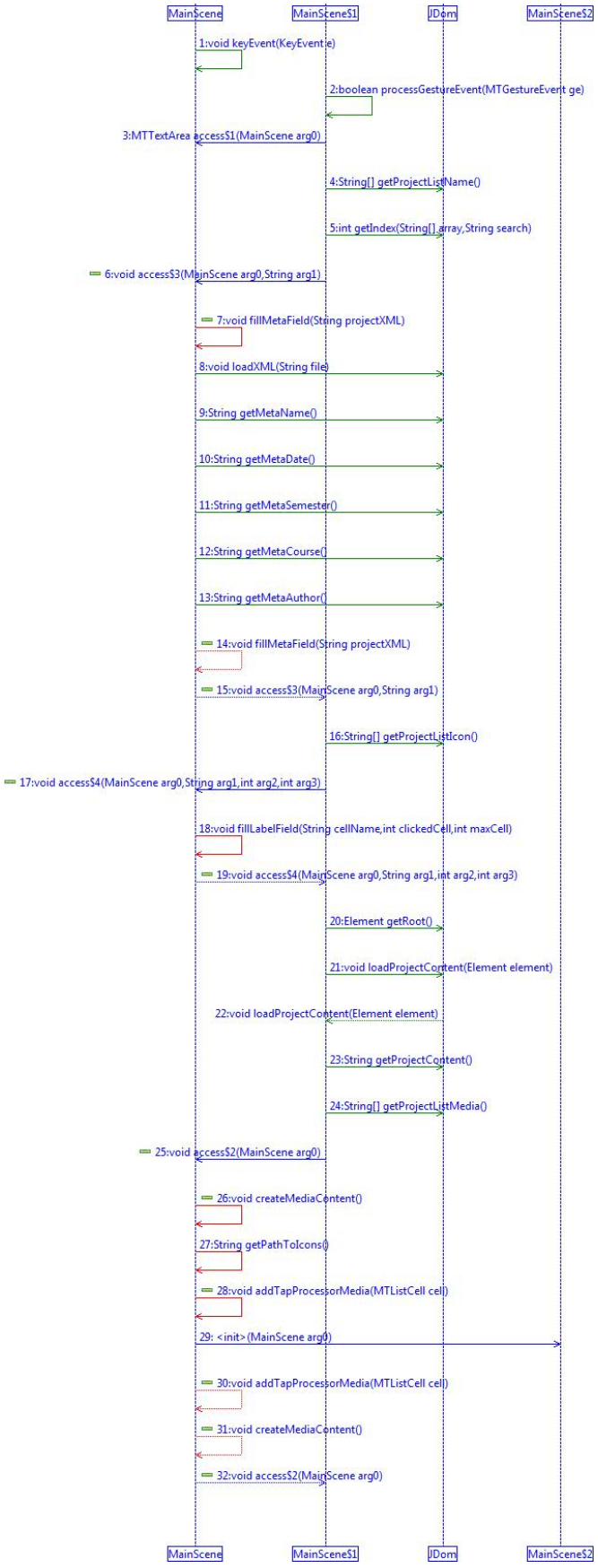

Abbildung A.7: Sequenzdiagramm: Öffnen eines Projektes

Literaturverzeichnis

[AT09] A. Teiche, C. Yanc C. Moore D. Solms G. Çetin J. Riggio N. Ramseyer P. D'Intino L. Muller R. Khoshabeh R. Bedi M. T. Bintahir T. Hansen T. Roth S. S. A. K. Rai R. A. K. Rai: *Multi- Touch Technologies.* 2009

[BA92] B. Ahlstroem, T. M. S. Lenman L. S. Lenman: *Overcoming Touchscreen User Fatigue By Workplace Design.* 1992

[BM00] B. McLaughlin, J. H.: *JDOM Project, http://www.jdom.org/.* since 2000

[Bux07] Buxton, B.: *Multi-Touch Systems that I Have Known and Loved.* 2007

[Bux10] Buxton, B.: *Invited Paper: A Touching Story: A Personal Perspective on the History of Touch Interfaces Past and Future.* 2010

[CF07] C. Forlines, C. Shen R. B. D. Wigdor W. D. Wigdor: *Direct-Touch vs. Mouse-Input for Tabletop Displays.* 2007

[Ehr02] Ehret, B. D.: *Learning Where To Look: Location Learning in Graphical User Interfaces.* 2002

[Has04] Hassenzahl, M.: *Funology: From Usability to Enjoyment (Human Computer Interaction), Kapitel 3: The thing and I: understanding the relationship between user and product.* 2004

[Hat01] Hatscher, M.: *Joy of Use: Determinanten der Freude bei der Software Nutzung.* 2001

[HS02] H. Sharp, J. P. Y. Rogers R. Y. Rogers: *Interaction Design: Beyond Human-Computer Interaction.* 2002

[JOW07] J. O. Wobbrock, Y. L. A. D. Wilson W. A. D. Wilson: *Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes.* 2007

[JS09] J. Schoening, P. O. A. Krueger K. A. Krueger: *Multi-touch is Dead, Long live Multi-touch.* 2009

[MH03] M. Hassenzahl, F. K. M. Burmester B. M. Burmester: *AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer QualitÃ€t.* 2003

[MK05] M. Kaltenbrunner, R. Bencina E. C. T. Bovermann B. T. Bovermann: *TUIO: A Protocol for Table-Top Tangible User Interfaces.* 2005

[PBB04] P. B. Brandtzæg, J. H. A. Følstad F. A. Følstad: *Funology: From Usability to Enjoyment (Human Computer Interaction), Kapitel 5: Enjoyment: Lessons from Karasek.* 2004

[PW04] P. Wright, L. M. J. McCarthy M. J. McCarthy: *Funology: From Usability to Enjoyment (Human Computer Interaction), Kapitel 4: Making sense of experience.* 2004

[RK90] R. Karasek, R. T.: *Healthy Work: Stress, Productivity, and the Reconstruction of Working Life.* 1990

[RMR01] R. M. Reimann, J. F.: *What we are, what we do, & what we need to know.* 2001

[Saf08] Saffer, Dan: *Designing Gestural Interfaces: Touchscreens and Interactive Devices.* 2008

[Saf09] Saffer, Dan: *Designing for Interaction: Creating Innovative Applications and Devices (Voices That Matter).* 2009

[SD10] S. Diefenbach, K. Kloeckner C. Nass A. M. M. Hassenzahl H. M. Hassenzahl: *Ein Interaktionsvokabular: Dimensionen zur Beschreibung der Aesthetik von Interaktion.* 2010

[Str04] Stribl, I.: *Joy of Use technischer Produkte.* 2004

[UL10] U. Laufs, J. Z. C. Ruff R. C. Ruff: *MT4j- A Cross-platform Multi-touch Development Framework.* 2010

Abbildungsverzeichnis

Tabellenverzeichnis