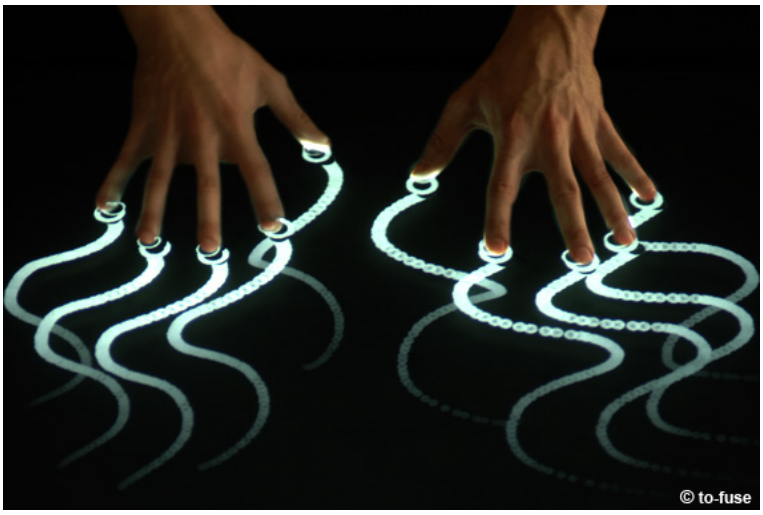
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

BACHELORARBEIT

Frühjahrssemester 2009

ANDREAS FISCHBACHER, MARCEL LENZ



Interaktiver Multi-Touch-Sales-Tisch

Hochschule für Technik Rapperswil
Abteilung Informatik
http://i.hsr.ch

Betreuer: Prof. Dr. Markus Stolze
Projektpartner: to-fuse

Februar 2009 - Juni 2009

[11. Juni 2009]

Andreas Fischbacher, Marcel Lenz: *Bachelorarbeit*, Interaktiver Multi-Touch-Sales-Tisch, © Februar 2009 - Juni 2009

BETREUER:
Prof. Dr. Markus Stolze

EXPERTE:
Markus Flückiger, Zühlke Engineering AG, Schlieren

GEGENLESER:
Prof. Dr. Lothar Müller

PROJEKTPARTNER:
Christian Iten und Emanuel Zgraggen, to-fuse, Zürich

ORT:
Rapperswil

VERSION:
Version 1.0, 11. Juni 2009

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

Teil I

EINFÜHRUNG

1 ABSTRACT

1.1 AUFGABENSTELLUNG

Verkaufsgespräche bei Mobiltelefonen laufen nicht immer erfolgreich ab. Dies ist sowohl für Kunden wie Verkäufer unbefriedigend. Teil des Problems ist, dass Verkäufer nicht immer genügend Detailwissen zu allen Produkten besitzen. Zudem kommt es vor, dass Verkäufer sich zu wenig Zeit nehmen, die Bedürfnisse von Kunden zu besprechen.

Durch den Einsatz von interaktiven Verkaufsunterstützungssystemen können diese Missstände behoben werden. Die in dieser Bachelorarbeit realisierte Applikation erweitert das Verkaufsgespräch um ein solches interaktives System mit dem Einsatz eines Multi-Touch-Tisches.

1.2 ZIEL DER ARBEIT

Der Multi-Touch-Tisch wird primär mit den Fingern bedient

Das Ziel der Arbeit ist die Entwicklung eines Applikationsprototyps für den Multi-Touch-Tisch der Hochschule für Technik Rapperswil. Diese Software unterstützt die Diskussion zwischen Kunde und Verkäufer. Sie ermöglicht den Vergleich verschiedener Smartphones und listet die weiteren Produkte auf. Die Applikation muss vollständig über den Multi-Touch-Tisch bedienbar sein.

Die Software muss die folgenden Hauptfunktionen erfüllen:

- Erkennung von auf dem Tisch platzierten und speziell präparierten Smartphones
- Lesen der Produkteigenschaften aus einer Datenbank
- attraktive Präsentation der Produkteigenschaften
- Bedienung des Tisches mit interaktiven Elementen

1.3 ERGEBNISSE

Smartphones werden mittels eines speziellen Musters erkannt. Die Darstellung der Daten erfolgt mit einer eigens dafür implementierten Tabelle und einem Produktrad.

Die für eine attraktive Präsentation benötigten Animationen werden mit einem Dispatcher aneinandergereiht. Die Realisation der Applikation erfolgte dabei unter der Verwendung von Java und OpenGL. Der Prototyp wurde mit dem multitouchfähigen Applikationsframework «Charger» der Firma to-fuse erstellt.

2 MANAGEMENT SUMMARY

2.1 EINLEITUNG

2.1.1 *Ausgangslage*

Die vorliegende Bachelorarbeit ist ein Forschungsprojekt des IFS[1] und to-fuse[2] unter der Leitung von Prof. Dr. Markus Stolze. Verwendet wurde die Multi-Touch-Plattform von to-fuse.

2.1.2 *Motivation*

Im Rahmen dieses Forschungsprojekts soll der Einsatz interaktiver Medien in Verkaufsshops zur Unterstützung von Beratungsgesprächen illustriert werden. Dabei ist lediglich eine rudimentäre Marschrichtung vorgegeben. Die Anforderungen wurden im Laufe der Arbeit konkretisiert.

2.1.3 *Stand der Technik*

Formfaktoren von 30 Zoll Bildschirmdiagonale und mehr sind nur bei spezialisierten Anbietern verfügbar

Die Verbreitung von multitouchfähigen Geräten erhöhte sich in letzter Zeit stark. Unter anderem ist dies auf den Boom des iPhone von Apple zurückzuführen. Hingegen sind grosse Geräte noch nicht allzu weit verbreitet. Neben individuellen Lösungen sticht der Microsoft Surface hervor. Es zeichnet sich ein Trend zu multitouchfähigen Applikationen ab, der durch die Multi-Touch-Unterstützung von Windows 7 begünstigt sein könnte.[3]

2.1.4 *Ist-Situation*

Multi-Touch-Tische sind derzeit noch wenig weit verbreitet. T-Mobile setzt dabei eine Applikation ein, die ein ähnliches Themengebiet abdeckt.[4]

Als Grundlage für die Arbeit stand der Multi-Touch-Tisch und das Charger Framework der Firma to-fuse zur Verfügung. Dieses Framework erkennt die Benutzereingaben und liefert grundlegende Darstellungsfunktionen. Zusätzlich zum Framework standen uns zur Entwicklungsunterstützung verschiedene Beispielapplikationen sowie das Entwicklerteam von to-fuse zur Verfügung. Diese behandelten jedoch nie den für diese Arbeit vorgesehenen Anwendungsbereich.

1 Institut für Software der HSR
2 http://www.to-fuse.ch/
3 Windows 7 ist ein von Microsoft entwickeltes Betriebssystem, das Ende 2009 erscheinen soll.
4 Ein Video dazu findet sich unter http://www.youtube.com/watch?gl=DE&hl=de&v=IIEtj1MJzaw

2.1.5 *Soll-Situation*

Anhand eines Szenarios sollen die Funktionen und der Einsatzbereich der Applikation verdeutlicht werden:

Peter ist auf der Suche nach einem neuen Smartphone. Er hat dabei noch kein konkretes Produkt vor Augen. In der Verkaufsstelle eines Mobilfunkanbieters gefällt Peter das Design eines Smartphones sofort. Er ist sich jedoch nicht sicher, ob dieses Produkt seine Anforderungen abdeckt.

Szenario

Peter platziert das Smartphone auf dem Multi-Touch-Tisch. Daraufhin werden ihm sofort die Produkteigenschaften mit den Bewertungen angezeigt. Um eine mögliche Kaufentscheidung zu erleichtern, kann Peter weitere Smartphones auf dem Tisch platzieren und dadurch die Produkteigenschaften direkt vergleichen. Für Peter sind die Musikfunktionen des Smartphones sehr wichtig. Deshalb priorisiert er diese Kategorie. Daraufhin zeigt ihm der Tisch weitere Smartphones an, welche seine Kriterien erfüllen. Beim Vergleich mit den anderen Produkten stellt Peter fest, dass sein ausgewähltes Smartphone exakt seinen Bedürfnissen entspricht.

Mit der Überzeugung, das richtige Produkt gekauft zu haben, verlässt Peter zufrieden die Verkaufsstelle.

Abbildung 1: Die Applikation in Aktion

2.1.6 *Herausforderungen*

Das Projekt liess sich grob in technische und gestalterische Herausforderungen teilen.

TECHNISCH Die Einarbeitung in eine neue Technologie stellte das grösste Entwicklungsrisiko dar. Vor allem die Entwicklung einer tabellarischen Darstellung war herausfordernd. Die Applikation wurde mit der Verwendung von Java und OpenGL umgesetzt.

GESTALTERISCH Die Interaktionsfähigkeiten bei der Bedienung eines Multi-Touch-Tisches unterscheiden sich stark von denjenigen einer gewöhnlichen Desktop-Applikation. Dementsprechend musste die Gestaltung auf diese neuartige Bedienung optimiert werden.

2.2 VORGEHEN

2.2.1 *Involvierte Personen*

Die folgenden Personen waren während des Projekts beteiligt:

- Andreas Fischbacher - Entwickler der Applikation
- Marcel Lenz - Entwickler der Applikation
- Prof. Dr. Markus Stolze - Betreuer der Arbeit, Dozent für Informatik an der Hochschule für Technik in Rapperswil
- Dipl. Des. Christian Iten, to-fuse
- Dipl. Ing. Inf. Emanuel Zgraggen, to-fuse

2.2.2 *Prozessmodell*

Das Projekt wurde mit dem Prozessmodell RUP[5] gemäss Larman [6] entwickelt. Für die Bestimmung der Anforderungen und des Softwaredesigns (Elaboration) wurden zwei Iterationsphasen und für die Entwicklung (Construction) vier Iterationsphasen geplant. Die Gesamtdauer des Projekts war auf 17 Wochen festgelegt.

Die Entwicklung wurde zudem stark vom User Centered Design[6] Prinzipien geprägt.

2.2.3 *Planung und Analyse*

Die Marschrichtung des Projekts war anfangs nur rudimentär vorgegeben. Somit bestand eine grosse Entfaltungsmöglichkeit für uns, dies machte jedoch die Projektplanung schwierig. Die Anforderungen mussten im Verlaufe des Projekts oftmals den neuen Gegebenheiten angepasst werden. Dies ist auf die Fokussierung auf die Benutzeroberfläche zurückzuführen. Gewichtige Änderungen an der Architektur mussten dabei jedoch nicht vollzogen werden.

2.2.4 *Implementation*

Anfänglich wurde die Erkennung eines Pinmusters erarbeitet. Schrittweise wurden danach die grafischen Elemente der Applikation entwickelt und getestet. Dazu gehörten unter anderem die Tabelle, die Anzeige der ähnlichen Produkte, das Produktrad und die diversen Animationen.

5 Der Rational Unified Process (RUP) ist ein objektorientiertes Vorgehensmodell zur Softwareentwicklung und ein kommerzielles Produkt der Firma Rational Software, die seit 2002 Teil des IBM Konzerns ist.
http://de.wikipedia.org/wiki/Rational_Unified_Process

6 Die nutzerorientierte Gestaltung zielt darauf ab, interaktive Produkte so zu gestalten, dass sie über eine hohe Gebrauchstauglichkeit (Usability) verfügen.
http://de.wikipedia.org/wiki/User_Centered_Design

2.2.5 *Qualitätssicherung*

Um die Stabilität der Applikation gewährleisten zu können, wurden diverse Qualitätsmassnahmen ergriffen. Zum einen wurden Hilfsmittel eingesetzt, die uns in der Codeanalyse unterstützten. Zum anderen wurden diverse Systemtests konzipiert, welche die Funktionalität der Applikation sicherstellten.

2.3 ERGEBNISSE

Eine funktionsfähige Java-Applikation liegt als Resultat vor. Diese Applikation erfüllt sämtliche geforderten Ziele. In der Benutzerdokumentation im Anhang werden die grafischen Elemente näher beschrieben.

Abbildung 2: Screenshot der Applikation mit Beschriftung der GUI-Elemente

2.3.1 *Erreichte Ziele*

Unter anderem wurden die folgenden Funktionalitäten implementiert:

- Mustererkennungsalgorithmus
- Datenbankanbindung
- tabellarische Darstellung
- Applikation komplett über Multi-Touch-Tisch bedienbar
- ansprechende grafische Animationen
- Vorschlag ähnlicher Produkte
- Anpassungsfähigkeit der Kategoriepräferenzen

Aus unserer Sicht entspricht das Ergebnis dem Geforderten. Auch der Industriepartner ist über das vorliegende Resultat hoch erfreut.

2.3.2 *Kosten*

Insgesamt hatten wir während den 17 Wochen einen Arbeitsaufwand von 806 Stunden. Daraus resultiert eine Abweichung von 6 Stunden, von den ursprünglich geplanten 800 Stunden. Insbesondere bei der Implementation der Benutzeroberfläche und der zahlreichen Animationen mussten wir deutlich mehr Zeit investieren als ursprünglich angenommen.

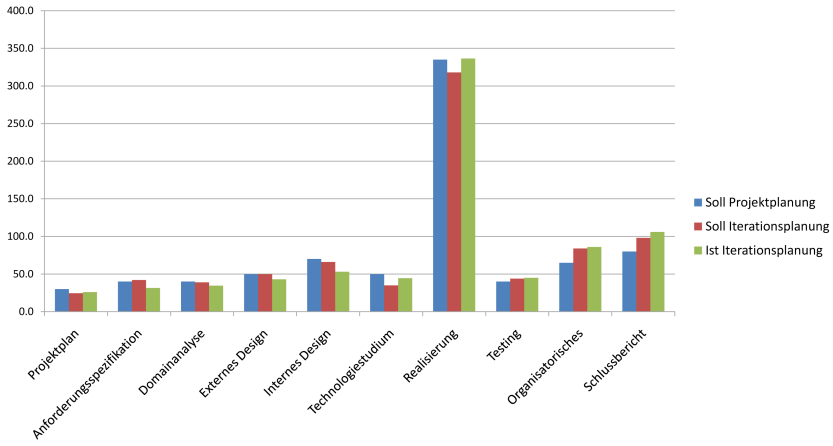


Abbildung 3: Diagramm Zeitaufwand pro Paket

2.4 AUSBLICK

2.4.1 *Lernpunkte*

Viele in der Theorie erlernte Methoden wurden praktisch angewendet. Insbesondere durch den Einsatz eines Papierprototyps konnte die grafische Oberfläche sehr schnell präzisiert werden. Die ungewohnte Interaktionsart erforderte ein Umdenken in der Gestaltung der Benutzeroberfläche. Die gesammelte Erfahrung ermöglicht eine neue Sichtweise auf ähnliche Systeme.

2.4.2 *Verbleibende Probleme*

Die Oberfläche des Multi-Touch-Tisches ist sehr lichtsensitiv. Bei Lichteinfall können aus diesem Grund die Objekte nur schwer oder gar nicht erkannt werden. Auch die Interaktionen mit den Fingern können dann teilweise nicht richtig interpretiert werden.

2.4.3 *Möglichkeiten zur Weiterentwicklung*

Ein mögliches Einsatzfeld der Applikation ist in einer Verkaufsstelle eines Mobilfunkanbieters zu finden. Die Applikation kann dementsprechend auf das Produktportfolio und das Corporate Design eines Anbieters angepasst werden. Insbesondere die Interaktion mit Gesten konnte im Rahmen dieser Arbeit nur begrenzt behandelt werden. Der Mustererkennungsalgorithmus, die Tabelle und der Eventdispatcher können in anderen Projekten mit dem Charger-Framework verwendet werden.

Teil II

TECHNISCHER BERICHT

[11. Juni 2009]

3 PROJEKTPLAN

3.1 PROJEKTORGANISATION

3.1.1 *Auftraggeber*

Mitwirkende Personen

HSR Hochschule für Technik Rapperswil
Institut für Software
Oberseestrasse 10
8640 Rapperswil

3.1.2 *Betreuer*

Prof. Dr. Markus Stolze
HSR Hochschule für Technik Rapperswil
Oberseestrasse 10
8640 Rapperswil

3.1.3 *Teammitglieder*

Andreas Fischbacher
Blattenstrasse 15
8717 Benken
andreas.fischbacher@hsr.ch

Marcel Lenz
Dattikonstrasse 5
8730 Uznach
marcel.lenz@hsr.ch

3.1.4 *Berater*

to-fuse
Christian Iten
Geroldstrasse 31/33
8005 Zürich

to-fuse
Emanuel Zgraggen
Geroldstrasse 31/33
8005 Zürich

3.1.5 *Veranwortlichkeiten*

Aufteilung der Aufgaben

Das Projekt wurde in verschiedene Aufgabenbereiche eingeteilt. Die jeweiligen Verantwortlichen stellen zugleich die Ansprechspartner für die internen sowie die externen Schnittstellen des jeweiligen Aufgabenbereichs dar. Fällt die verantwortliche Person aus, so übernimmt das andere Teammitglied den Aufgabenbereich.

MARCEL LENZ	ANDREAS FISCHBACHER
Projektleitung (intern)	Protokollierung
Design-Leiter	Coding-Leiter
Qualitätssicherung	Testing und Automatisierung

Tabelle 1: Verantwortlichkeiten

3.1.6 *Sitzungen*

Die neuen Iterationen werden nach dem Ende der alten Iteration geplant. Damit ist eine Anpassung an die sich ändernden Anforderungen sehr gut möglich. Jeweils zu Beginn einer Iteration wird eine Teamsitzung durchgeführt. Um die Qualität des Codes zu sichern, werden die Programmteile in wöchentlichen Code-Reviews analysiert. Wöchentlich finden Sitzungen mit dem Betreuer Markus Stolze statt, in denen das weitere Vorgehen besprochen wird.

3.2 MANAGEMENT-ABLÄUFE

3.2.1 *Projekt Kostenvoranschlag*

Das Projekt dauert vom 16.02.2009 bis zum 12.06.2009. Für die Bearbeitung wird ein Richtwert von 800 Arbeitsstunden angestrebt. Dies ergibt eine durchschnittliche Arbeitszeit von rund 23.5 Stunden pro Woche pro Teammitglied. Vom 09.04.2009 bis zum 15.04.2009 sind gemäss dem Jahreskalender der HSR die Frühlingsferien.

3.2.2 *Prozessmodell*

Das Projekt wird nach dem Prinzip des Rational Unified Process realisiert. Dies beinhaltet unter anderem

- das Festlegen von Phasen, die mittels Iterationen ein oder mehrmals durchschritten werden.
- das setzen von Meilensteinen, bei denen ein Review mit dem Projektbetreuer und eine Beurteilung statt findet.

Beschreibung gemäss Larman [6].

3.2.3 *Iterationsplanung*

Bei der Elaboration-Phase werden zwei Iterationen und bei der Construction-Phase vier Iterationen eingeplant. Für die Inception-Phase und die Transition-Phase werden je eine Iteration eingeplant. Nach dem Ende einer Iteration ist jeweils ein Meilensteintermin festgelegt. Diese Termine müssen zwingend eingehalten werden. Eine Ausnahme bildet hier die erste Iteration der Construction-Phase. Hier wird kein Meilenstein anvisiert.

Die Iterationsphasen des RUP im Überblick

INCEPTION In der Inception-Phase wird der Projektplan erarbeitet. Dieser beinhaltet unter anderem eine Risikoanalyse, die Aufteilung der Arbeitspakete und die geplanten Meilensteine. Das primäre Ziel in der Inception-Phase ist die Abschätzung der Projektdauer.

ELABORATION In der Elaboration-Phase wird sowohl das Domain Modell, wie auch das Datenbankmodell erstellt. Zusätzlich werden Use Cases erstellt, respektive die bereits vorhandenen erweitert und System-Sequenzdiagramme erzeugt. Das Ziel der Elaboration-Phase ist die theoretische Ausarbeitung der Systemarchitektur und die Erkennung von zu priorisierenden Systemkomponenten.

CONSTRUCTION In der Construction-Phase wird die Applikation entwickelt. Dies beinhaltet unter anderem die Programmierung der Programmlogik, der Benutzeroberfläche und der Datenhaltung. Ebenfalls werden Unit- und Systemtests in dieser Phase ausgeführt und ausgewertet.

TRANSITION In der Transition-Phase werden nur noch geringe Anpassungen vorgenommen. Der Hauptaspekt liegt bei der Optimierung des Produkts. Im Falle der Bachelorarbeit fällt zudem die Fertigstellung des Schlussberichts in diese Phase.

Projektwoche / Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Inception		MS1															
Elaboration 1			MS2	MS3													
Elaboration 2					MS4		MS5										
Construction 1																	
Construction 2											MS6						
Construction 3														MS7			
Construction 4																MS8	
Transition																	MS9

Abbildung 4: Iterationsplanung und Meilensteine

3.2.4 *Meilensteine*

	MEILENSTEIN	DATUM
MS1	Projektplanung	27.02.09
MS2	Anforderungsspezifikation	06.03.09
MS3	Domainanalyse	16.03.09
MS4	Externes Design	27.03.09
MS5	Internes Design, Prototyp	08.04.09
MS6	Alpha-Version	01.05.09
MS7	Beta-Version	22.05.09
MS8	Final-Version	05.06.09
MS9	Schlussbericht	12.06.09

Tabelle 2: Meilensteine

Erläuterung der verschiedenen Versionen

ALPHA-VERSION Die Alpha-Version beinhaltet die komplette Progammlogik. Zudem enthält sie eine primitive Benutzeroberfläche.

BETA-VERSION Bei der Beta-Version ist das GUI komplett implementiert. Zusätzlich werden die auf den Multi-Touch-Sales-Tisch gelegten Objekte erkannt.

FINAL-VERSION Unvollständigkeiten und Fehler aus der Beta Version sind behoben.

3.3 ARBEITSPAKETE

Bei der Planung der Pakete wurden bewusst Zeiträume für eintreffende Risiken offen gehalten und in der prognostizierten Arbeitsdauer einkalkuliert.

3.3.1 *P01 - Projektplan*

Im Projektplan werden sämtliche projektrelevanten Termine und die Arbeitspakete festgelegt.

P01 PROJEKTPLAN	30 STUNDEN

- Dokument «Projektplan» erstellen
- Projektorganisation festlegen und Iterationen / Meilensteine planen
- Arbeitspakete, Verantwortlichkeiten zuteilen
- Projektplan erweitern und korrigieren
- Protokollierung und Zeiterfassung

Tabelle 3: P01 - Aufgaben Projektplan

3.3.2 *P02 - Anforderungsspezifikation*

Sämtliche in der Anforderungsspezifikation festgehaltenen Anforderungen müssen für einen erfolgreichen Projektabschluss berücksichtigt werden.

P02 ANFORDERUNGSSPEZIFIKATION	40 STUNDEN

- Anforderungen definieren (25 Stunden)
  - Erfassen der funktionalen und nichtfunktionalen Anforderungen des Produkts
  - Erstellen des entsprechenden Dokuments
  - Überarbeiten der definierten Anforderungen
- Use Cases (15 Stunden)
  - Brief Use Cases aus den funktionalen Anforderungen erstellen
  - Beschreiben der wichtigsten Use Cases im Fully-Dressed-Format
  - Beschreiben aller Use Cases im Fully-Dressed-Format (in einer späteren Iteration)

Tabelle 4: P02 - Aufgaben Anforderungsspezifikation

3.3.3 *P03 - Domainanalyse*

In der Domainanalyse wird die Systemlogik festgelegt.

P03 DOMAINANALYSE	40 STUNDEN

- Domain- und Datenbankmodell (25 Stunden)
  - Erstellen des Domain Models mit Hilfe der Use Cases
  - Erstellen des Datenbankmodells
- System-Sequenzdiagramme (5 Stunden)
  - Erstellen der wichtigsten System-Sequenzdiagramme
- Domainanalyse-Dokument (10 Stunden)
  - Erstellen des Domainanalyse-Dokuments
  - Erweitern des Domainanalyse-Dokuments

Tabelle 5: P03 - Aufgaben Domainanalyse

3.3.4 *P04 - Externes Design*

Im Externen Design wird die Darstellung der Applikationsoberfläche skizzenhaft erarbeitet.

P04 EXTERNES DESIGN	50 STUNDEN

- Entwurf der Programmoberfläche
- Paper Prototype

Tabelle 6: P04 - Aufgaben Externes Design

3.3.5 *P05 - Internes Design*

Im internen Design werden die Erkenntnisse aus der Domainanalyse umgesetzt. Dabei wird die Systemarchitektur festgelegt.

P05 INTERNES DESIGN	70 STUNDEN

- Klassendiagramme (40 Stunden)
  - Klassendiagramm der Problem Domain erstellen
  - Klassendiagramme für alle restlichen Schichten erstellen
- Sequenzdiagramme (10 Stunden)
  - Sequenzdiagramme zur Veranschaulichung von komplexen Abläufen erstellen
- Architektur (10 Stunden)
  - Pakete definieren
  - Abhängigkeiten und Aufgaben der Pakete bzw. Schichten zuteilen
  - Schnittstellen definieren
- Designdokument (10 Stunden)
  - Erstellen des Designdokuments
  - Erweitern des Designdokuments

Tabelle 7: P03 - Aufgaben Internes Design

3.3.6 *P06 - Dokument- und Technologiestudium*

Das Paket Dokument- und Technologiestudium reserviert Zeit, die für die Einarbeitung in unbekannte Techniken benötigt wird.

P06 DOKUMENT- UND TECHNOLOGIESTUDIUM	50 STUNDEN

- Studium OpenGL
- Studium Framework Multi-Touch-Tisch
- Studium LaTeX
- Studium weiterer Technologien
- Skriptestudium

Tabelle 8: P06 - Aufgaben Technologiestudium

3.3.7 *P07 - Realisierung*

In der Realisierung erfolgt die Programmierung der geplanten Programmteile.

P07 REALISIERUNG	335 STUNDEN

- Problem Domain (50 Stunden)
  - Umsetzung der Programmlogik
- Datenhaltung (30 Stunden)
  - Realisierung der Datenhaltung
- GUI - Darstellung (80 Stunden)
  - Erstellen sämtlicher Oberflächenelemente
  - Implementierung einer benutzerfreundlichen Bedienung
  - Anordnung und Implementation der verschiedenen Panels
- GUI - Events und Observer (50 Stunden)
  - Reaktionen auf Eingaben
  - Updates der Benutzeroberfläche
- GUI - Animationen (30 Stunden)
  - Erstellung von Animationen
- Controller (30 Stunden)
  - Abstrahierung der Eingabemöglichkeiten
- Erweiterung Framework (15 Stunden)
  - Sich wiederholende Aufgaben direkt im Framework integrieren
- Erkennungsalgorithmen (50 Stunden)
  - Erkennung von Geräten anhand der Pin-Anordnung

Tabelle 9: P07 - Aufgaben Realisierung

3.3.8 *P08 - Testing*

Im Paket Testing finden sich sowohl Unit-Tests als auch System-Tests.

P08 TESTING	40 STUNDEN

- Unit-Tests (15 Stunden)
  - Erstellung der Unit-Tests
  - Ausführung der Unit-Tests
- System-Tests (25 Stunden)
  - Erstellung der System-Tests
  - Ausführung der System-Tests
  - Evaluierung der System-Tests

Tabelle 10: P08 - Aufgaben Testing

3.3.9 *P09 - Organisatorisches*

Sämtliche organisatorischen Aufgaben wie beispielsweise Sitzungen werden im Paket Organisatorisches zusammengefasst.

P09 ORGANISATORISCHES	65 STUNDEN

- Sitzungen (40 Stunden)
  - Teamsitzungen
  - Sitzungen mit Betreuer
- Code Reviews (15 Stunden)
  - Besprechung wichtiger Codefragmente
- Sonstige organisatorische Tätigkeiten (10 Stunden)

Tabelle 11: P09 - Aufgaben Organisatorisches

3.3.10 *P10 - Schlussbericht*

Der Schlussbericht entspricht der abzugebenden Bachelorarbeit.

P10 SCHLUSSBERICHT	80 STUNDEN

- Schlussbericht ergänzen und formatieren
- Rechtschreibungsprüfung

Tabelle 12: P10 - Schlussbericht

3.4 RISIKOANALYSE

Bei der Planung der Pakete wurden bewusst Zeiträume für eintreffende Risiken offen gehalten und in der prognostizierten Arbeitsdauer einkalkuliert. Neben dem Risiko wird zudem die Auftrittswahrscheinlichkeit (WSK) in der Risikoanalyse festgehalten.

	RISIKO	WSK	FOLGE	KOSTEN
R1	Versteckte Kundenanforderungen	mittel	Zusatzaufwand, Zeitverzögerung	Sehr hoch
R2	Framework nicht oder nur teilweise brauchbar	gering	Zusatzaufwand, Zeitverzögerung	Sehr hoch
R3	Nichterreichen der verlangten Leistungen	mittel	Erhöhte Komplexität	Hoch - sehr hoch
R4	Mitgliederausfall	mittel	Zusatzaufwand, Zeitverzögerung	Zeitabhängig, Gering bis hoch
R5	Überforderung der Mitarbeiter in Bezug auf ihr softwaretechnisches Können	gering	Zusatzaufwand, Zeitverzögerung	Sehr Hoch
R6	Vergoldung	gering	Erhöhte Komplexität	Hoch
R7	Geräteerkennung funktioniert nicht wie gewünscht	mittel	Zusatzaufwand, Zeitverzögerung	Sehr hoch
R8	Ausfall der Hardware	gering	Zeitverzögerung	Sehr hoch

Tabelle 13: Risikoanalyse

Massnahmen zur Risikominimierung

Um die Wahrscheinlichkeit eines Risikofalls zu minimieren, sind die folgenden Massnahmen zu befolgen:

R1 Das Risiko der versteckten Kundenanforderungen kann durch eine detailliert erfasste Anforderungsspezifikation verringert werden.

R2 Ist das Framework nicht oder nur teilweise brauchbar, muss entweder das Framework erweitert oder eine eigene Testumgebung festgelegt werden.

R3 Um die verlangten Leistungen zu erreichen, ist die Befolgung der Anforderungsspezifikation notwendig. Zudem kann durch die Absprache mit dem Kunden und durch Abschätzungen bei den Reviews das Risiko minimiert werden.

R4 Um einem Ausfall eines Teammitglieds entgegenzuwirken, ist eine grosszügige Zeitplanung von Nöten. Zusätzlich muss das andere Teammitglied während des Ausfalls eine Mehrarbeit verrichten, damit die geplanten Termine fristgerecht eingehalten werden können.

R5 Dem Risiko einer Überforderung der Mitarbeiter in Bezug auf ihr softwaretechnisches Können kann mit einer ausreichenden Zeit für das Technologiestudium entgegengewirkt werden. Zudem stehen kompetente Berater zur Verfügung.

R6 Eine strikte Einhaltung der Anforderungsspezifikation und das Setzen von Zielen minimieren das Risiko der Vergoldung.

R7 Bei Problemen wird auf komplexe Anordnungsstrukturen verzichtet. Zudem wird genügend Zeit für die Geräteerkennung eingeplant.

R8 Bei einem Ausfall der Hardware muss die Herstellerfirma des Multi-Touch-Tisches konsultiert werden.

3.4.1 *Eingetroffene Risiken*

Die folgenden Risiken sind während des Projektverlaufs eingetroffen:

RISIKO	MEHRAUFWAND	PROBLEMBESCHREIBUNG UND MASSNAHMEN
R2	rund 3h	Die transparenten Flächen von übereinandergelegten Texturen wurden oftmals nicht komplett angezeigt. Die Ursache hierfür waren nicht korrekt gesetzte OpenGL-Parameter. Dementsprechend musste das Framework ergänzt werden.
R7	rund 2h	Die Geräte konnten teilweise nur schwer erkannt werden. Dies lag hauptsächlich am Lichteinfall auf die Tischoberfläche.
R8	rund 5h	Die Kamera des Multi-Touch-Tisches war defekt. Mit dem Einsatz einer neuen Kamera wurde das Problem behoben.

Tabelle 14: Risikoanalyse

4 PROJEKTMONITORING

Nachfolgend finden sich verschiedene Diagramme, die den Zeitaufwand visualisieren. Die dazu verwendeten Daten entstammen dem Projektplan und dem auf der CD zu findenden Detailplan.

Falls nicht weiter beschrieben, bezeichnet die Y-Achse die Anzahl Stunden.

4.1 GESAMTAUFWAND

4.1.1 *Zeitaufwand pro Paket*

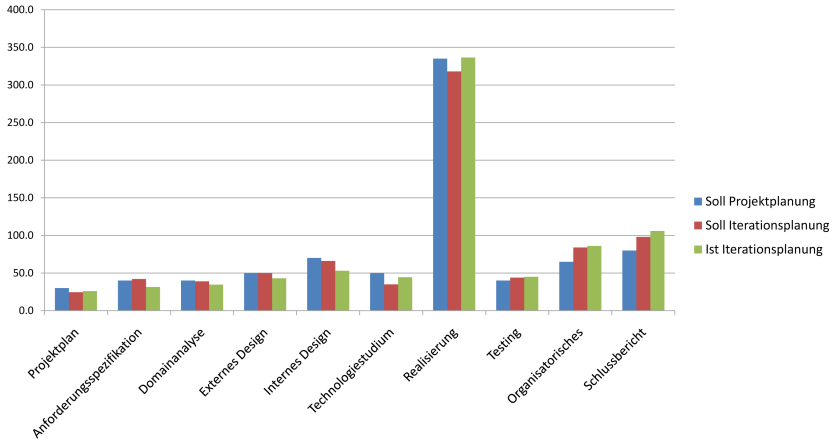


Abbildung 5: Diagramm Zeitaufwand pro Paket

Der Zeitaufwand der Pakete konnte grösstenteils eingehalten werden. Grosse Unterschiede zwischen der ursprünglichen Planung und dem wirklichen Resultat ergaben sich nur in den Paketen Internes Design, Organisatorisches und im Schlussbericht.

Die eingetragenen Werte basieren auf Erfahrungswerten

Beim Paket Internes Design setzten wir den Zeitaufwand durch einen Erfahrungswert aus einem früheren Projekt genügend hoch an. Wie sich herausstellte, wurde nicht die gesamte Zeit benötigt und wir hatten einen Zeitaufwandüberschuss von 17 Stunden.

Die benötigte Zeit für die Erstellung des Schlussberichts wurde von uns ein wenig unterschätzt. Der Aufwand um die einzelnen Dokumente für die Abgabe zu überarbeiten, zusammenzufügen und in einen Zusammenhang zu bringen verursachte bei uns einen Unterschied zur ursprünglichen Planung von 18 Stunden.

Beim Vergleich zwischen der «Soll-Iterationsplanung» und der «Ist-Iterationsplanung» ist der Unterschied zwischen dem geplanten Zeitaufwand der Pakete und dem tatsächlichen Zeitaufwand deutlich geringer ausgefallen. Hier gab es nur kleine Abweichungen, da jeweils

alle zwei Wochen der Zeitaufwand für die nächste Iteration festgelegt werden konnte. Einzig bei der Realisierung betrug der Unterschied rund 18 Stunden. Hierbei spielten für den zusätzlichen Zeitaufwand oft einige kurzfristig aufgetauchte Bugs eine Rolle.

4.1.2 *Prozentualer Anteil*

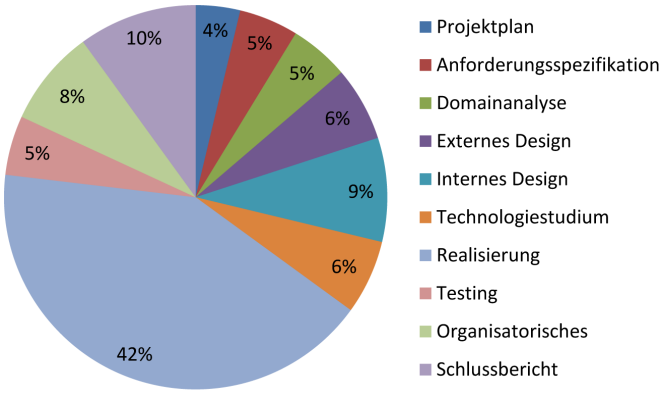


Abbildung 6: Diagramm Soll Projektplanung

Die Projektplanung sah vor, dass etwas mehr als 40% der Zeit für die Realisierung aufgewendet werden soll. Die restliche Zeit verteilt sich auf einige kleinere Pakete.

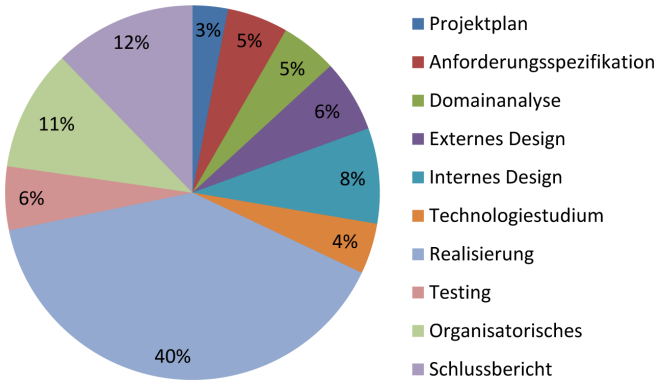


Abbildung 7: Diagramm Soll Iterationsplanung

Bei den Iterationsplanungen gab es meist nur Änderungen im ein- bis zweiprozentigen Bereich. Der einzige grössere Unterschied zeigt sich beim Paket Organisatorisches. Dies liegt vor allem daran, dass wir einige organisatorische Tätigkeiten zeitlich unterschätzt haben. Dazu gehört unter anderem die Gestaltung der Broschüre und des Plakats.

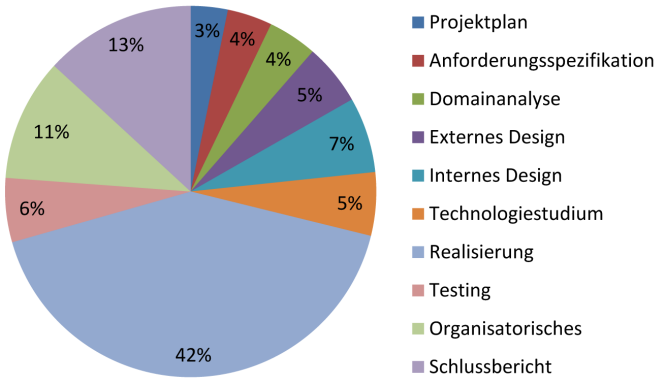


Abbildung 8: Diagramm Ist Iterationsplanung

Die schlussendlichen Resultate bei der Ausführung unterscheiden sich von der Planung nur gering und sind nicht weiter erwähnenswert.

4.2 ZEITAUFWAND REALISIERUNG

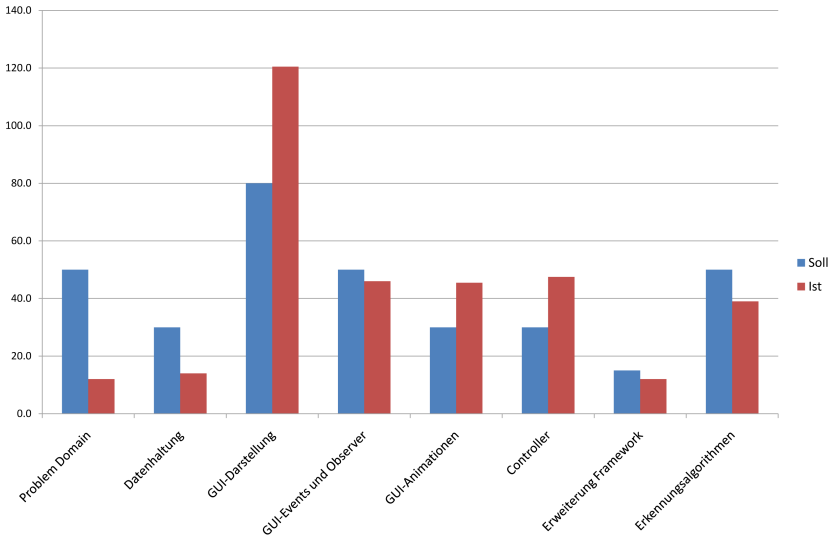


Abbildung 9: Zeitaufwand für das Paket Realisierung

Der Aufwand für die Realisierung konnte in der Projektplanung zwar einigermassen präzise eingeschätzt werden, innerhalb des Paketes sind jedoch grosse Differenzen vorhanden. Insbesondere die Bereiche Problem Domain, Datenhaltung und das GUI unterscheiden sich stark von der ursprünglichen Planung.

Zusätzlicher Fokus auf die Benutzeroberfläche

Dies ist mit dem zusätzlichen Fokus auf die Benutzeroberfläche zu erklären. Der Zeitaufwand für die Problem Domain und die Datenhaltung konnte durch den Einsatz von Hibernate und einer Reduktion der Anforderungen sehr gering gehalten werden. Dadurch konnte der Benutzeroberfläche und den Controllern zusätzliche Zeit gewidmet werden.

4.3 ZEITAUFWAND PRO WOCHE

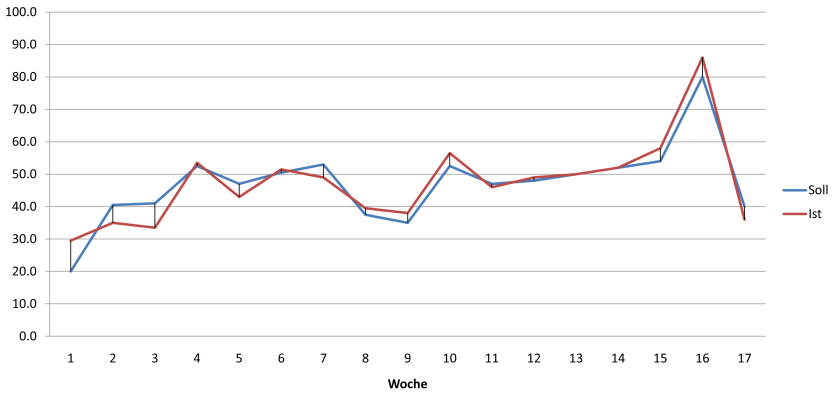


Abbildung 10: Diagramm Wochentotal

Anfangs wurden primär sämtliche Dokumentvorlagen erstellt und das vorgegebene Framework studiert. Mit diesen ersten Erkenntnissen konnte ein Projektplan erstellt werden. Der zeitliche Anstieg in der vierten Projektwoche ist auf die Konzeption der Programmlogik und die gleichzeitige Planung des externen Designs zurückzuführen. Das externe Design musste aufgrund einer Präsentation für die Swisscom frühzeitig forciert werden.

Schwankungen

Die Schwankungen in der achten und der neunten Woche sind auf die Frühlingsferien der HSR zurückzuführen. Der starke Anstieg in der 16. Woche ist mit dem Ende des offiziellen Schulbetriebs zu erklären. Dadurch konnten wir uns in dieser Woche voll und ganz der Bachelorarbeit widmen.

In der letzten Woche wurden die anstehenden Arbeiten frühzeitig ausgeführt um auf keine zeitlichen Probleme zu stossen. Deswegen mussten nebenbei nur noch einige kleinere organisatorischen Arbeiten erledigt werden.

4.4 ZEITAUFWAND PRO PERSON

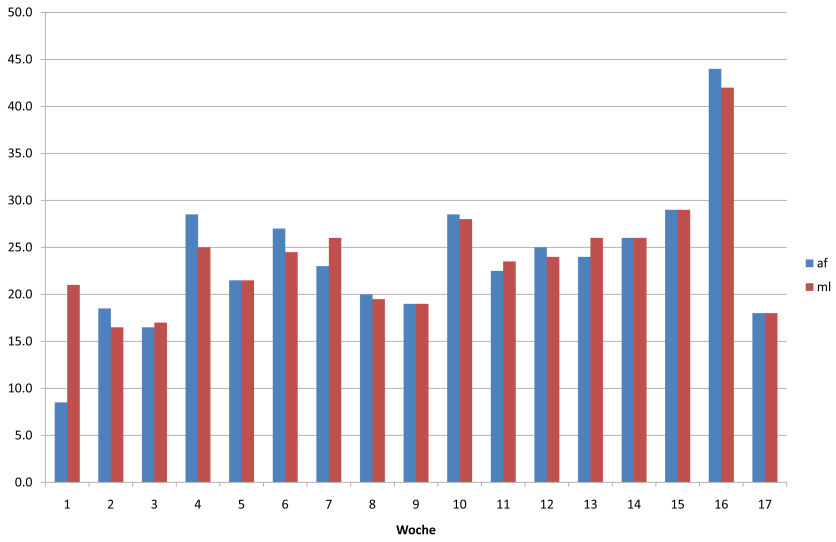


Abbildung 11: Diagramm Wochentotal pro Person

Der Unterschied zu Beginn des Projekts ist durch die Erstellung des TeX-Templates für den Bericht zu erklären. Im Verlaufe des Projekts wurden die Arbeiten gleichermassen aufgeteilt. Oftmals wurde auch die paarweise Programmierung eingesetzt.

4.5 ERFAHRUNGSBERICHT MARCEL LENZ

4.5.1 *Projektverlauf*

Während des Projekts waren wir vielen Anforderungsänderungen ausgesetzt

Aufgrund der doch eher vage formulierten Aufgabenstellung war ich am Anfang des Projekts nicht sicher, was mich erwartete. Durch die gute Instruktion des Betreuers erhielt unser Team jedoch sehr schnell eine Vision des Projekts und ab diesem Zeitpunkt konnte zielgerichtet gearbeitet werden. Die Projektplanung konnte aufgrund der Erfahrungen, die in der Studienarbeit gesammelt wurden, schnell bearbeitet werden. Die sich immer wieder ändernden Anforderungen resultierten in einer agilen Entwicklung. Dies hatte auch zur Folge, dass es einige grössere Abweichungen von der Projektplanung gab. Die Realisierung der Datenbankschicht und der Programmlogik ging aus meiner Sicht erstaunlich schnell voran. Insbesondere die Einbindung von Hibernate war in kürzester Zeit erledigt. Ebenfalls überrascht war ich, dass der Refactoring-Prozess deutlich kürzer als bei den vorangegangenen Projekten ausfiel. Dies ist meines Erachtens auf das paarweise Programmieren zurückzuführen.

Überrascht war ich von der Effizienz des Papierprototyps. Ein in der Theorie von meiner Seite eher belächelter Ansatz überzeugte mich in der Praxis sehr.

4.5.2 *Arbeit im Team*

Eine gute Zusammenarbeit mit dem Team ist in einem Projekt sehr wichtig. Aufgrund der Tatsache, dass Andreas Fischbacher und ich kompromissbereit waren, konnten alle auftretenden Probleme schnell und einfach bei einer Aussprache gelöst werden. Dadurch kam es auch nie zu Unstimmigkeiten im Team. Durch das paarweise Programmieren konnten Fehler direkt und einfach analysiert und verbessert werden.

4.5.3 *Arbeit mit den Betreuern*

Durch die wöchentlichen Teamsitzungen waren während des Projektverlaufs selten offene Fragen vorhanden. Bei der Planung und Umsetzung des externen Designs konnten durch die Inputs des Betreuers Prof. Dr. Markus Stolze sowie durch Christian Iten und Emanuel Zgraggen viele Probleme umgangen werden.

4.5.4 *Fazit*

Mit dem Resultat bin ich sehr zufrieden. Aus meiner Sicht konnte ich an einem äusserst interessanten Projekt arbeiten. In der Zukunft werden wohl noch viel mehr verschiedene Interaktionsmöglichkeiten verwendet. Im Rahmen dieser Arbeit konnte ich zumindest eine dieser neuen Technologien ausgiebig testen und erste Erfahrungen auf diesem Gebiet sammeln.

4.6 ERFAHRUNGSBERICHT ANDREAS FISCHBACHER

4.6.1 *Projektverlauf*

Zu Beginn der Arbeit war für mich noch unklar, wie gross der Projektumfang sein wird. Vor allem die evtl. auftretenden Probleme bei der Implementation der Benutzeroberfläche konnten sehr schwer eingeschätzt werden, da die Elemente für die Oberfläche selbst erstellt werden mussten und wir bisher noch nie eine reine touchbasierte Benutzeroberfläche erstellt hatten. Um diese Unsicherheiten zu eliminieren, haben wir zu Beginn für uns ein kleines Programm anhand des Spieles Pong erstellt. Nach einer Besprechung mit den to-fuse-Mitarbeitern über den Programmcode war uns dann klar, wie wir zu verfahren hatten. Danach konnten die Anforderungen spezifiziert werden. Über den Papierprototypen konnten wir die Oberfläche und deren Bedienung durch den Benutzer evaluieren. Der Papierprototyp stellte sich als überaus nützlich heraus und erfüllte seinen Zweck voll und ganz. Die Realisierung nahmen wir mit paarweisem Programmieren in Angriff. Durch diese Vorgehensweise konnte die Studienarbeit bereits erfolgreich abgeschlossen werden und half uns auch bei der Kontrolle des Codes. Während der Realisierung stellten wir überrascht fest, dass vor allem die Datenschicht sehr schnell implementiert war. Dabei half uns der Einsatz von Hibernate sehr.

4.6.2 *Arbeit im Team*

Während des ganzen Projektes fand ich die Zusammenarbeit mit Marcel Lenz sehr angenehm. Wir konnten jederzeit bei unterschiedlichen Meinungen eine konstruktive Diskussion über Pro und Contra der verschiedenen Ansichten führen. Die zusätzliche Bereitschaft beider Teammitglieder bei Termindruck mehr Zeit z.B. an Wochenenden zu investieren wirkte sich positiv auf unsere gute Stimmung aus und spornte uns zusätzlich an.

4.6.3 *Arbeit mit den Betreuern*

Sehr positiv fand ich die wöchentlichen Teamsitzungen mit dem Betreuer, in denen jeweils der Stand des Projektes bekanntgegeben wurde. Die Teilnahme der to-fuse-Mitarbeiter zu Beginn des Projektes nützte uns dabei zusätzlich. Bei diesen Sitzungen gaben uns die to-fuse-Mitarbeiter Erklärungen und eine kleine Einführung, wodurch wir uns relativ schnell einen Einblick in die Programmierumgebung für den Tisch verschaffen und damit arbeiten konnten. Bei der Bedienung und der Positionierung der Benutzerelemente wurden wir zudem durch Prof. Dr. Markus Stolze sehr kompetent beraten. Wir konnten jederzeit auf die Mithilfe der beteiligten Personen zählen, was uns sehr freute.

4.6.4 *Fazit*

Meines Erachtens lieferten wir ein gutes Endprodukt ab, welches die Anforderungen abdeckt und auch zukünftig flexibel erweitert werden kann. Die Erstellung dieser Multi-Touch-Anwendung ermöglichte uns einen Einblick in ein Programmiergebiet, in welchem man nicht alle Tage arbeitet. Auch die Anwendung eines Papierprototyps machte mir den Sinn und Zweck dieses Werkzeugs klarer. Das gelernte Wissen aus der Studienarbeit konnte zudem vertieft werden.

5 ANFORDERUNGSSPEZIFIKATION

5.1 STAND DER TECHNIK

5.1.1 *Was ist Multi-Touch?*

Ein multitouchfähiges Gerät besitzt eine Oberfläche, bei der mehrere aktive Kontaktpunkte zur gleichen Zeit erkannt werden. Das Gerät kann Berührungsinformationen räumlich und zeitlich zuordnen. Celentano and Minuto [2]

5.1.2 *Geräteübersicht*

Es befinden sich verschiedene multitouchfähige Geräte auf dem Markt oder in der Entwicklung. In diesem Abschnitt werden nur die wichtigsten Geräte behandelt. Eine detailliertere Liste ist im Anhang zu finden.

TO-FUSE MULTI-TOUCH PLATTFORM Der patentierte Multi-Touch-Screen der Firma to-fuse beruht auf einer Rückprojektion, kombiniert mit einem Infrarot Kamerasystem. Gegenüber anderen Methoden wie zum Beispiel FTIR, zeichnet er sich durch eine hohe Touch-Sensibilität sowie eine hohe Intoleranz gegenüber Umgebungslicht aus. Da das System nicht Druck- sondern Berührungsempfindlich ist, erlaubt es auch die Erkennung von Gegenständen, welche auf dem Screen platziert sind.[1]

MULTI-TOUCH VON JEFF HAN Dieses Gerät ist eines der ersten und berühmtesten Multi-Touch-Geräte. Die Funktionsweise des Gerätes basiert auf der frustrated total internal reflection (FTIR). Han [4].

MICROSOFT SURFACE Surface ist ein von Microsoft produzierter Multi-Touch-Tisch. Dabei wird eine kamerabasierte Berührungserkennung mit Infrarot-Beleuchtung und 5 Kameras eingesetzt. Janssen [5]

APPLE IPHONE Beim Apple iPhone handelt es sich um ein Smartphone. Es lässt sich dadurch nicht direkt mit den zuvor aufgelisteten Geräten, welche eine grosse Berührungsoberfläche aufweisen, vergleichen. Durch die weltweite Verbreitung des Geräts konnten jedoch die Möglichkeiten einer Multi-Touch-Bedienung aufgezeigt werden.

5.1.3 *Zukunft*

Das von Microsoft entwickelte Betriebssystem Windows 7 (Erscheinungsdatum gegen Ende 2009) wird Multi-Touch vollumfänglich unterstützen. Es ist davon auszugehen, dass die Verbreitung von Multi-Touch-Geräten weiter ansteigen wird.

1 Zitat von Christian Iten

Ein möglicher Trend ist auch die Verwendung von sehr kleinen Geräten. Patrick Baudisch und Gerry Chu zeigen in ihrem Artikel «Back-of-Device Interaction Allows Creating Very Small Touch Devices» die Möglichkeiten solcher Geräte auf. Baudisch and Chu [1]

In einer von Microsoft veröffentlichten Zukunftsvision für das Jahr 2019 sind einige Multi-Touch-Interaktionen zu sehen. [2]

5.2 SZENARIO

Ein Kunde will ein neues Smartphone[3] erwerben. Er hat dabei noch kein Präferenzprodukt ausgewählt. Allerdings muss das Smartphone gewisse Kriterien erfüllen, um seinen Ansprüchen gerecht zu werden. Bei der Kaufberatung muss ihn demnach ein Verkäufer unterstützen. Dabei unterscheidet sich der Wissensstand zwischen dem Kunden und dem Verkäufer oftmals drastisch. Schnell verliert der Kunde den Überblick über die vielen technischen Eigenschaften. Der Kunde will für den Preis möglichst viel Leistung erhalten. Dies steht oftmals im Kontrast mit dem Ziel des Verkäufers. Dieser will dem Kunden ein möglichst teures Gerät verkaufen. Dadurch kann das Verkaufsgespräch sowohl für den Kunden, wie auch für den Verkäufer unbefriedigend enden.

Vision

Bei den im Szenario beschriebenen Problemen setzt der Multi-Touch-Sales-Tisch an. Er bringt den Kunden und den Verkäufer auf die gleiche Wissensebene. Die Gliederung der technischen Eigenschaften in bewertbare Kategorien dient der transparenten und übersichtlichen Darstellung der Gerätedetails. Verschiedene Smartphones können direkt und nach einheitlichen Kriterien verglichen werden. Die Gefahr, dass der Käufer ein für seine Bedürfnisse überdimensioniertes Gerät ersteht, wird deutlich verringert. Im Gegenzug ermöglicht der Multi-Touch-Sales-Tisch auch einem unerfahrenen Verkäufer eine einfache und kompetente Beratung.

5.3 DAS VERKAUFSGESPRÄCH

5.3.1 *Aufbau des Gesprächs*

James E. Dion [3] listet in seinem Buch «Retail Selling Ain't Brain Surgery, It's Twice As Hard» die folgenden neun Schritte für ein erfolgreiches Verkaufsgespräch aus der Sicht des Verkäufers auf:

1. The Greeting
2. Needs Determination
3. Product Knowledge
4. Suggestion Selling
5. Trading Up
6. Answering Objections

2 Video unter http://www.youtube.com/watch?v=rxVS5nYFnkA
3 Ein Smartphone vereint den Leistungsumfang eines Mobiltelefons mit dem eines Personal Digital Assistants (PDA).
http://de.wikipedia.org/wiki/Smartphone

7. The Close
8. Maximizing the Last Moment
9. After Sales Service

Mit dem Einsatz des Multi-Touch-Sales-Tisches sollen insbesondere die Punkte drei bis sechs unterstützt werden.

5.3.2 *Verbesserungspotential beim Verkaufsgespräch*

Die Punkte drei bis sechs können wie folgt durch den Multi-Touch-Sales-Tisch unterstützt werden:

3. PRODUCT KNOWLEDGE Damit der Verkäufer Produktkenntnisse vermitteln kann, muss er die Produkte kennen. Dies ist leider in der Praxis oftmals nur eingeschränkt der Fall. Der Multi-Touch-Sales-Tisch hilft hierbei mit der transparenten Darstellung der Produktinformationen.

«Most Customers respect and enjoy Suggestion Selling.» Dion [3]

4. SUGGESTION SELLING Der Multi-Touch-Sales-Tisch zeigt Alternativen auf, die dem gewählten Produkt ähnlich sind. Dies erleichtert dem Verkäufer das Anbieten von Alternativen. Er muss hierfür die Produkte nicht detailliert kennen, da die Applikation ihm die passenden Geräte anzeigt.

«If you show the best product at the beginning of the sale, and let the Customer see the difference between the good, the better, and the best products, she most often will choose the better or the best one.»Dion [3]

5. TRADING UP Mit dem Multi-Touch-Sales-Tisch können die Produkte direkt verglichen werden. Dadurch kann das beste Produkt leichter ermittelt werden. Selbstverständlich ist dabei der Verkauf des besten Geräts immer noch vom Geschick des Verkäufers abhängig.

6. ANSWERING OBJECTIONS Beim Mobiltelefonverkauf sind Einwände oftmals technischer Natur. Mit dem Multi-Touch-Sales-Tisch können aufgrund der transparenten Darstellung der Produktdaten diese Einwände gezielt beantwortet werden.

5.4 ALLGEMEINE BESCHREIBUNG

5.4.1 *Programmübersicht*

Übersicht der Möglichkeiten

Ein Käufer will bei seinen Kaufentscheiden unterstützt werden. Der Verkäufer, welcher ihm behilflich ist, möchte dem Kunden möglichst schnell einen Überblick über die Smartphones und deren Eigenschaften geben.

Die zu erarbeitende Applikation erkennt die speziell präparierten Smartphones und zeigt die dazu verfügbaren Informationen an. Sind mehrere Smartphones auf dem Tisch, können diese direkt miteinander verglichen werden. Dies ermöglicht dem Kunden, das Smartphone schnell und einfach anhand der äusseren (Design, Gewicht, Grösse, etc.) wie auch der inneren (Speicherkapazität, Stromverbrauch, etc.) Werte zu beurteilen. Ist der Kunde unsicher welches Smartphone er kaufen möchte, so werden ihm ähnliche Smartphones vorgeschlagen.

Die in dieser Arbeit behandelte Applikation kann auch auf andere

Produkte übertragen werden, was allerdings nicht Teil dieses Projektes ist.

Das Systemziel

SYSTEMZIEL Das System zielt darauf ab, den Verkäufer sowie auch den Käufer auf eine gemeinsame Wissensebene während des Verkaufsgesprächs zu bringen. Dabei soll das System dem Käufer eine Vergleichsmöglichkeit bieten und dem Verkäufer Fakten liefern, welche die Argumente für das gewählte Smartphone untermauern. Das System muss beim Verkaufsgespräch unterstützend wirken und darf dabei keinen störenden Einfluss nehmen.

Das Projektziel

PROJEKTZIEL Das Ziel des Projekts ist die Entwicklung eines stabilen Prototyps einer verkaufsunterstützenden Applikation, welche durch den multitouchfähigen Tisch eine einfache Bedienmöglichkeit zur Verfügung stellt. Dabei wird sowohl dem Verkäufer als auch dem Käufer durch eine einmalige Vorführung die Bedienung klar. Die Applikation bietet dabei eine Vergleichsmöglichkeit für verschiedene Smartphones mit einer detaillierten Anzeige und eine Empfehlung für weitere Smartphones anhand einer vom Kunden bevorzugten Kategoriereihenfolge.

5.4.2 *Projektvoraussetzungen*

MULTI-TOUCH-TISCH Ein Multi-Touch-Tisch kann im Gegensatz zu einem normalen druckempfindlichen Display mehrere Druckpunkte gleichzeitig auswerten. Für die Arbeit steht ein solcher Tisch zur Verfügung.

FRAMEWORK Um die Programmierung einer Applikation zu erleichtern, wurde für den Multi-Touch-Tisch ein Framework von to-fuse entwickelt. Dieses Framework ermittelt unter anderem die Druckpunkte und stellt deren Daten über eine Schnittstelle zur Verfügung.

5.5 FUNKTIONALE ANFORDERUNGEN

5.5.1 *Bedienung Multi-Touch-Tisch*

Unterstützte Gesten

GESTENERKENNUNG Bei der Bedienung des Multi-Touch-Tisches werden zwei verschiedene Gesten unterstützt: der Flick und das Verschieben. Der Flick wird dabei durch die Berührung des Objekts und einem schnellen Zug in eine Richtung ausgelöst. Beim Verschieben wird das Objekt berührt und danach herumgezogen bis man die gewünschte Position erreicht hat. Danach wird das Objekt wieder losgelassen.

OBJEKTERKENNUNG Die Erkennung von Smartphones wird durch den Gebrauch von Füssen an den Produkten realisiert. Durch die unterschiedlichen Anordnungen der Füsse kann ein Algorithmus die Smartphones identifizieren.

5.5.2 *Datenhaltung*

Die Daten werden in einer Datenbank gespeichert

Die Daten der Smartphones werden in einer Datenbank abgelegt. Die Verwaltung dieser Daten kann hierbei direkt über die Datenbank erfolgen. Im Rahmen dieser Arbeit muss keine Verwaltungsadministration erstellt werden.

5.5.3 *Produktvergleich*

Der Produktvergleich wird über eine Werteskala realisiert. Die technischen Eigenschaften werden dabei kategorisiert. Beispielsweise sind in der Kategorie «Foto» die Details der integrierten Kamera enthalten. Um dem Kunden eine Entscheidungshilfe beim Kauf eines Smartphones zu geben, wird ein Wert pro Kategorie angezeigt. Anhand dieses Werts können die verschiedenen Kategorien klar bewertet werden. Dabei gilt zu beachten, dass für den Produktvergleich keine Kundenprofile und Verkäuferprofile vorhanden sind.

5.5.4 *Produktempfehlung*

Die Empfehlung eines Smartphones wird über einen Abgleich mit den vom Kunden präferierten Kategorien abgegeben. Dabei werden alle vorhandenen Produkte anhand ihres Gesamtresultats vom Besten zum Schlechtesten dargestellt.

5.6 NICHTFUNKTIONALE PRODUKTANFORDERUNGEN

5.6.1 *Bedienbarkeit*

Die Standards von Multi-Touch-Applikationen werden eingesetzt.

Die Applikation wird ausschliesslich über den Multi-Touch-Tisch bedient. Deshalb müssen die grafischen Elemente der Applikation so gross dargestellt werden, dass die Bedienung mittels Finger ohne Probleme möglich ist.

Dies bedingt, dass die Elemente untereinander mit einem ausreichend grossen Abstand ausgestattet sind, was Fehleingaben verhindert.

Bei der Bedienung der Elemente werden die bereits vorhandenen Standards von Multi-Touch-Anwendungen berücksichtigt.

5.6.2 *Zuverlässigkeit*

Als Endresultat muss ein stabiler Prototyp vorliegen. Der Prototyp muss hierfür die folgenden fünf Anwendungsfälle ohne Absturz sowie mit guter Interaktivität und Attraktivität durchführen können:

- Ein Smartphone wird auf dem Tisch platziert.
- Die weiteren Smartphones werden anhand der Kategoriepräferenz angezeigt.
- Zwei Smartphones können verglichen werden.
- Die Präferenz der Kategorien kann verändert werden.
- Ein Smartphone wird vom Tisch entfernt.

Die Interaktion mit der benutzten Datenbank und der Tischoberfläche muss gewährleistet sein.
Damit die grafische Performanz der Applikation hoch gehalten werden kann, wird Java bindings for OpenGL (JOGL[4]) eingesetzt.

5.6.3 *Wartbarkeit*

Die Wartbarkeit der Produktdaten wird über eine Datenbank gewährleistet. Die Anpassung der Produktdaten über den Multi-Touch-Tisch ist nicht Teil dieser Arbeit.

5.6.4 *Implementierungsbedingungen*

Für die Umsetzung des Projektes wird Java in der Version 1.6 eingesetzt. Als Programmierbasis dient das von to-fuse zur Verfügung gestellte Charger-Framework. Dieses wird von uns im für die Bachelorarbeit benötigten Masse erweitert. Das Framework beinhaltet Java bindings for OpenGL (JOGL).

5.6.5 *Hardware*

Es wird der Multi-Touch-Tisch mit dem dazugehörigen Rechner verwendet.

5.6.6 *Software*

Als Betriebssystem wird Windows XP eingesetzt. Die Applikation wird durch die Verwendung von Java 1.6 und OpenGL unabhängig vom Betriebssystem lauffähig sein. Als Datenbanksystem wird MySQL[5] eingesetzt.

4 Das JOGL-Projekt stellt eine Schnittstelle zur OpenGL-API bereit. OpenGL wird für die performante Darstellung von Grafiken und Animationen verwendet.
https://jogl.dev.java.net/
5 http://www.mysql.com/

5.7 ANWENDUNGSFÄLLE

AUSGANGSLAGE Alle Produkte (Beispiel Smartphone) sind neben dem Multi-Touch-Tisch positioniert.

5.7.1 *Use Case Diagramm*

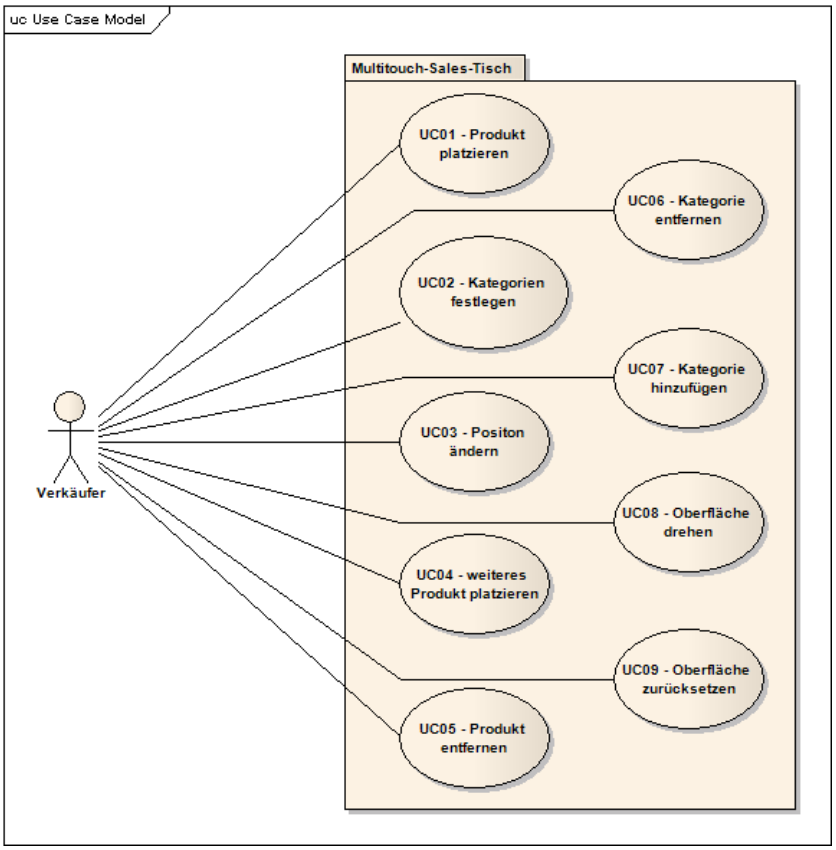


Abbildung 12: Use Case Diagramm

5.7.2 *Brief Use Cases*

UC01 Der Verkäufer platziert ein Produkt auf der leeren Tischoberfläche. Das System erkennt das Produkt und zeigt die entsprechenden Produktinformationen in der Vergleichstabelle an. Das System rangiert die weiteren Produkte nach der Kategoriereihenfolge des erkannten Produktes und zeigt diese an. *Produkt platzieren*

UC02 Der Verkäufer legt die Reihenfolge der Produktkategorien fest. Aus diesen präferierten Eigenschaften erstellt das System eine neue Rangierung der weiteren Produkte. Das System passt demnach die im UC01 beschriebenen Alternativvorschläge automatisch an. *Kategorien festlegen*

UC03 Der Verkäufer verschiebt das angezeigte Produkt auf die Position eines anderen Produktes in der Vergleichstabelle. Das System tauscht darauf die Positionen der Produkte. *Position ändern*

weiteres Produkt platzieren

UC04 Der Verkäufer platziert ein weiteres Produkt auf der Tischoberfläche. Das System zeigt die Informationen des Produkts an der Position in der Tabelle an.

Produkt entfernen

UC05 Der Verkäufer will ein Produkt aus der Vergleichstabelle entfernen (Beispielsweise nach dem Entfernen des Smartphones). Das System blendet das Produkt aus und stellt die Tabelle ohne dieses Produkt dar.

Kategorie entfernen

UC06 Der Verkäufer will eine Kategorie aus der Vergleichstabelle entfernen. Das System blendet diese Kategorie aus und stellt die Tabelle ohne diese Kategorie dar.

Kategorie hinzufügen

UC07 Der Verkäufer will eine zusätzliche Kategorie der Vergleichstabelle hinzufügen. Der Verkäufer selektiert die gewünschte Kategorie, worauf das System diese Kategorie einblendet und die Tabelle mit der zusätzlichen Kategorie darstellt.

Oberfläche drehen

UC08 Der Verkäufer will die Oberfläche drehen, damit der Kunde die Produkteigenschaften besser lesen kann. Der Verkäufer wählt die gewünschte Aktion, worauf das System die Oberfläche um 180 Grad dreht.

Oberfläche zurücksetzen

UC09 Der Verkäufer hat das Verkaufsgespräch beendet und will nun die Oberfläche auf den Anfangszustand zurücksetzen. Der Verkäufer wählt die gewünschte Aktion, worauf das System die Oberfläche zurücksetzt.

5.7.3 *Fully dressed Use Cases*

Die Fully dressed Use Cases beschreiben die Abläufe auf Seiten des Systems sowie des Benutzers ausführlich.

UC01 - Produkt platzieren

- Motivation: Kunde will Details über ein Smartphone erhalten.
- Auslöser: Verkäufer
- Vorbedingungen: keine
- Input: Smartphone mit Erkennungsmuster
- Output: Details des Smartphones in Vergleichstabelle

BENUTZERABSICHT	SYSTEMAKTION
1. Benutzer platziert Smartphone auf dem Tisch	
	2. System durchläuft einen Erkennungsalgorithmus
	3. Produkt wurde erkannt. System zeigt Produktinformationen in Vergleichstabelle an
	4. System rangiert weitere Smartphones nach der Kategoriepräferenz des Produkts
	5. System zeigt weitere Produkte nach deren Rangierung an
	F.1 System erkennt Produkt nicht. Der Vorgang wird abgebrochen und nichts wird angezeigt

Tabelle 15: Ablauf und Fehlerfälle UC01

UC02 - Kategorien festlegen

- Motivation: Kunde will seine bevorzugten Kategorien des Smartphones festlegen.
- Auslöser: Verkäufer
- Vorbedingungen: Smartphone wurde bereits erkannt
- Input: Eingaben des Verkäufers
- Output: geänderte Reihenfolge der weiteren Produkte und neue Anordnung der Kategorien

BENUTZERABSICHT	SYSTEMAKTION
1. Benutzer ändert die Wichtigkeit der Kategorien	
	2. System rangiert die weiteren Produkte passend zur neuen Gewichtung
	3. System zeigt neue Rangierung der Produkte an

Tabelle 16: Ablauf und Fehlerfälle UC02

UC03 - Position ändern

- Motivation: Kunde will ein angezeigtes Produkt auf eine andere Position in der Vergleichstabelle verschieben.
- Auslöser: Verkäufer
- Vorbedingungen: Vergleichstabelle wird mit mindestens zwei Produkten angezeigt
- Input: Verschieben des Produktes
- Output: vertauschte Produkte in der Vergleichstabelle

BENUTZERABSICHT	SYSTEMAKTION
1. Benutzer ändert die Position des Produktes auf der Vergleichstabelle	
	2. System überprüft neue Position
	3. System verschiebt Produkt an neue Position
	4. System verschiebt das vorher angezeigte Produkt an Position des Eingabeproduktes

Tabelle 17: Ablauf und Fehlerfälle UC03

UC04 - weiteres Produkt platzieren

- Motivation: Kunde will Details über ein weiteres Smartphone erhalten.
- Auslöser: Verkäufer
- Vorbedingungen: Es befindet sich bereits mindestens ein Smartphone auf dem Tisch.
- Input: Smartphone mit Erkennungsmuster
- Output: Details des Smartphones

BENUTZERABSICHT	SYSTEMAKTION
1. Benutzer platziert Smartphone auf dem Tisch	
	2. System durchläuft einen Erkennungsalgorithmus
	3. Produkt wurde erkannt. System zeigt Produktinformationen in der Vergleichstabelle an
	F.1 System erkennt Produkt nicht. Der Vorgang wird abgebrochen und nichts angezeigt

Tabelle 18: Ablauf und Fehlerfälle UC04

UC05 - Produkt entfernen

- Motivation: Kunde will ein Produkt aus der Vergleichstabelle entfernen.
- Auslöser: Verkäufer
- Vorbedingungen: Die Vergleichstabelle wird angezeigt.
- Input: gewähltes Produkt
- Output: Vergleichstabelle wird ohne das gewählte Produkt angezeigt

BENUTZERABSICHT	SYSTEMAKTION
1. Benutzer entfernt Produkt von der Vergleichstabelle	
	2. System stellt Vergleichstabelle ohne das gewählte Produkt dar

Tabelle 19: Ablauf und Fehlerfälle UC05

UC06 - Kategorie entfernen

- Motivation: Kunde will eine Kategorie aus der Vergleichstabelle entfernen.
- Auslöser: Verkäufer
- Vorbedingungen: Die Vergleichstabelle wird angezeigt.
- Input: gewählte Kategorie
- Output: Vergleichstabelle wird ohne die gewählte Kategorie angezeigt

BENUTZERABSICHT	SYSTEMAKTION
1. Benutzer entfernt Kategorie von der Vergleichstabelle	
	2. System stellt Vergleichstabelle ohne die gewählte Kategorie dar
	3. System berechnet Rangierung der weiteren Produkte neu
	4. System zeigt die entfernte Kategorie am Rand an

Tabelle 20: Ablauf und Fehlerfälle UC06

UC07 - Kategorie hinzufügen

- Motivation: Kunde will eine Kategorie der Vergleichstabelle hinzufügen.
- Auslöser: Verkäufer
- Vorbedingungen:
  - Die Vergleichstabelle wird angezeigt.
  - Es sind Kategorien vorhanden, die in der Vergleichstabelle nicht angezeigt werden.
- Input: gewählte Kategorie
- Output: Vergleichstabelle wird mit der gewählten Kategorie angezeigt

BENUTZERABSICHT	SYSTEMAKTION
1. Benutzer fügt Kategorie der Vergleichstabelle hinzu	
	2. System stellt Vergleichstabelle mit der gewählten Kategorie dar
	3. System berechnet Rangierung der weiteren Produkte neu
	4. System entfernt die Kategorie am Rand

Tabelle 21: Ablauf und Fehlerfälle UC07

UC08 - Oberfläche drehen

- Motivation: Kunde will sich die Vergleichstabelle ansehen. Er steht jedoch auf der anderen Seite des Tisches und sieht so alles auf dem Kopf.
- Auslöser: Verkäufer
- Vorbedingungen: keine
- Input: keine
- Output: Oberfläche wird um 180 Grad gedreht angezeigt

BENUTZERABSICHT	SYSTEMAKTION
1. Benutzer wählt die Aktion «drehen» aus	
	2. System dreht die Oberfläche um 180 Grad

Tabelle 22: Ablauf und Fehlerfälle UC08

UC09 - Oberfläche zurücksetzen

- Motivation: Der Verkäufer hat das Verkaufsgespräch beendet und möchte die Oberfläche auf den Anfangszustand zurücksetzen.
- Auslöser: Verkäufer
- Vorbedingungen: keine
- Input: keine
- Output: Oberfläche wurde zurückgesetzt

BENUTZERABSICHT	SYSTEMAKTION
1. Benutzer wählt die Aktion «reset» aus	
	2. System setzt die Oberfläche auf den Anfangszustand zurück

Tabelle 23: Ablauf und Fehlerfälle UC09

6 EXTERNES DESIGN

6.1 VORAUSSETZUNGEN

Eine multitouchfähige Applikation unterscheidet sich grundsätzlich von einer normalen Desktopapplikation. Die wichtigsten Charakteristiken und die Probleme, die bei solchen Applikationen auftreten können, werden in diesem Abschnitt kurz beschrieben.

6.1.1 *Charakteristiken einer Touch-Applikation*

Gemäss den «Windows Vista User Experience Touch Guidelines» [7] besitzt eine touchfähige Applikation die folgenden Charakteristiken:

- Natürlich und Intuitiv
  Jeder kann mit den Fingern Dinge berühren.
- Weniger aufdringlich
  Der Gebrauch von Berührungen lenkt weit weniger ab, als eine Tastatureingabe oder ein Mausklick.
- Direkt und einfach
  Berührungen ermöglichen das Gefühl einer direkten Interaktion mit den Objekten auf dem Bildschirm. Im Gegensatz dazu erfordert eine Maus koordinative Bewegungen, die nicht direkt auf dem Bildschirm stattfinden.
- Verringerte Präzision
  Der Benutzer kann die Objekte nicht derart präzise anvisieren wie mit einer Maus oder einem Stift. Dadurch darf man vom Benutzer nicht erwarten, dass er kleine Objekte berührt oder manipuliert.

6.1.2 *Probleme einer Touch-Applikation*

Fünf Probleme die bei Touch-Applikationen auftreten können

Bei der Konstruktion des externen Designs musste insbesondere die verringerte Präzision berücksichtigt werden. Microsoft listet in den «Windows Vista User Experience Touch Guidelines» [7] die folgenden fünf Probleme auf, welche beim Einsatz einer touchfähigen Applikation auftreten können:

- Kleine Bedienelemente sind schwierig zu gebrauchen.
- Müssen auf dem Bildschirm weite Strecken zurückgelegt werden, kann dies sehr ermüdend wirken.
- Hover[1] kann nicht erkannt werden.
- Texteingabe und Textselektion sind schwierig.
- Kleine Ziele in der Nähe des Bildschirmrandes können sehr schwer erreichbar sein.

1 Hover bezeichnet das Schweben über einem Bereich. Wenn ein Mauszeiger über einem Bedienelement liegt, kann dieses Element dadurch einen Effekt anzeigen.

6.1.3 *Spezialfall kleine Flächen*

Fitts Law besagt, dass die benötigte Zeit um mit einem Ziel zu interagieren abhängig von dessen Grösse und der Distanz ist. Je kleiner und weiter entfernt das Ziel ist, desto grösser wird der Zeitaufwand einer Interaktion. Aufgrund der Grösse der Fingerkuppen können kleine Bedienelemente nicht präzise benutzt werden.

Minimalgrösse eines Bedienelements

Gemäss Microsoft [7] ist eine Grösse von 23x23 Pixel eine gute minimale Grösse eines Interaktionselements für jede Art von Eingabegeräten. Im Vergleich dazu, ist das in Abbildung 13 dargestellte Bedienelement viel zu klein für eine effiziente Nutzung in einer touchfähigen Applikation.

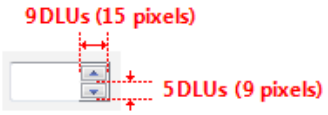


Abbildung 13: Beispiel für ein zu kleines Bedienelement (©Microsoft)

6.2 GESTALTUNG DES EXTERNEN DESIGNS

Bei der Gestaltung des externen Designs wurde nach dem folgenden Muster vorgegangen:

1. Erstellung einer Oberflächenskizze
2. Digitalisierung der Oberflächenskizze mithilfe eines Grafikprogrammes
3. Erstellung eines Papier-Prototypen

Nach jedem Punkt wurde das externe Design mit dem Betreuer der Arbeit besprochen und die gewonnenen Erkenntnisse wurden in der nächsten Iteration berücksichtigt.

6.2.1 *Oberflächenskizze*

Mit Bleistift und Papier wurde in einem ersten Schritt eine rudimentäre Skizze der Benutzeroberfläche erarbeitet. Sie enthält dabei lediglich die Kernelemente des Oberflächenlayouts. Die Skizze wurde erstellt, um allen am Projekt beteiligten Personen einen Einblick zu ermöglichen.

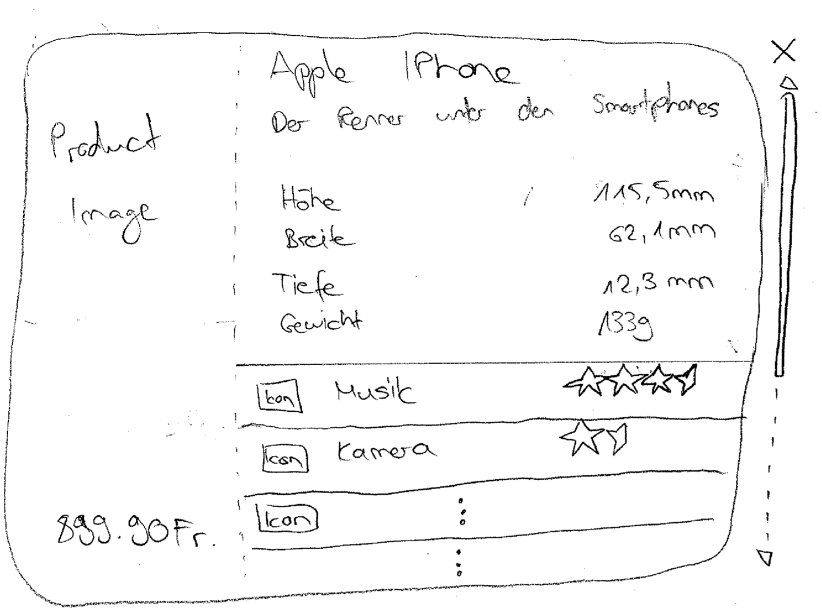


Abbildung 14: Handskizze der Einzelansicht eines Smartphones

Einzelansicht

Wird ein Smartphone auf den Multi-Touch-Sales-Tisch gelegt, wird das in der Abbildung 14 dargestellte Fenster angezeigt. Die Bewertung der Kategorien wird nicht mit einem numerischen Wert, sondern mit einer Anzeige von Sternen dargestellt. Umso besser das Produkt in einer Kategorie ist, desto mehr Sterne werden angezeigt. Die Anzeige von fünf Sternen entspricht dem Maximalwert in einer Kategorie.

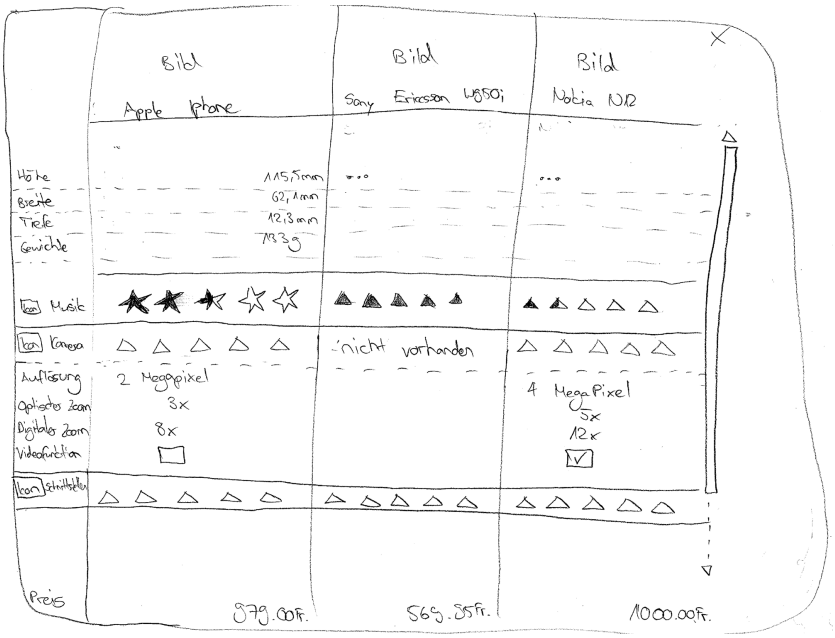


Abbildung 15: Handskizze der Vergleichsansicht von drei Smartphones

Vergleichsansicht

Sollen mehrere Smartphones verglichen werden, kommt die in der Abbildung 15 skizzierte Tabelle zum Zug. Die Kategorien sind hierbei aufklappbar und geben einen detaillierten Einblick in die technischen Eigenschaften des Gerätes.

Kritik an der Oberflächenskizze

Beim Review der Oberflächenskizze fiel insbesondere der Scrollbalken am rechten Rand negativ auf. Zudem sind die in der Skizze statisch festgehaltenen Attribute wie Höhe, Breite, Tiefe und Gewicht nicht zwingend hervorzuheben.

6.2.2 *Digitalisierte Oberflächenskizze*

Die Digitalisierung der Oberflächenskizze wurde schnell vorangetrieben, um einige Bildschirmfotos für einen Prospekt anzufertigen. Während bei der Erstellung der handskizzierten Oberfläche noch wenige Überlegungen zur Interaktion mit den Bedienelementen gemacht wurden, wurden diese Überlegungen bei der digitalisierten Oberflächenskizze miteinbezogen.

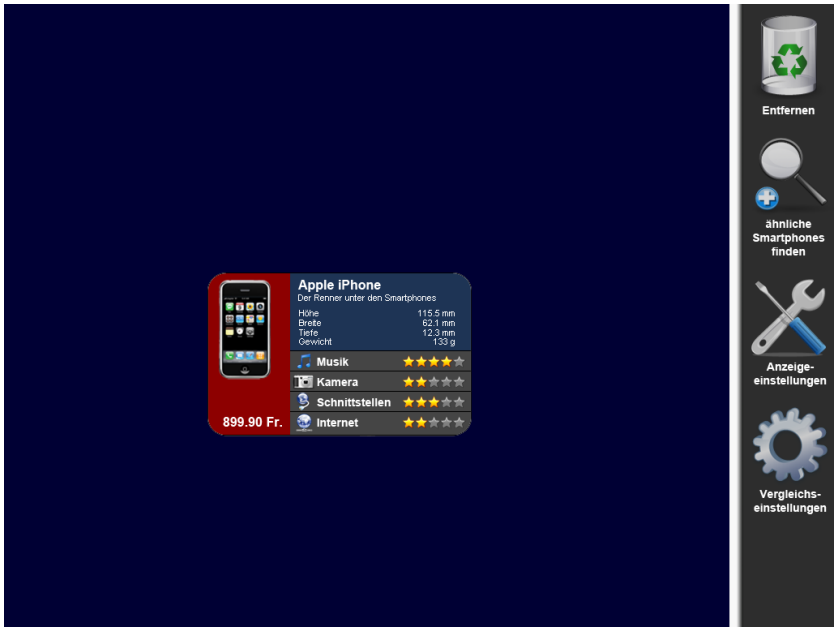


Abbildung 16: Digitalisierte Einzelansicht eines Smartphones

Einzelansicht

Um mit den Bedienelementen zu interagieren, wurde eine Seitenleiste am rechten Rand eingeführt (siehe Abbildung 16). Diese unterstützt die benötigten Kommandos, die für die Bedienung der Applikation zur Verfügung stehen. Befinden sich mehrere Smartphones auf dem Tisch kann die Vergleichsansicht angezeigt werden, indem zwei Einzelansichten ineinander verschoben werden.

Kritik an der digitalisierten Oberflächenskizze

Beim Review der digitalisierten Einzelansicht stach die Seitenleiste sofort als grosser Negativpunkt ins Auge. Während des Reviews wurden verschiedene Alternativen diskutiert, um die vier Kommandos ohne den Bedarf einer Seitenleiste zu ermöglichen. Dies wird durch den Einsatz von diversen Gesten ermöglicht. Als Beispiel kann hierbei eine Kategorie durch einen Flick[2] entfernt werden und muss nicht extra in den Papierkorb geschoben werden.

2 Ein Flick ist eine kurze Ziehbewegung.

Abbildung 17: Digitalisierte Vergleichsansicht von drei Smartphones

Für die in der Abbildung 17 dargestellte Vergleichstabelle wurde die Handskizze eins zu eins - ohne den Scrollbalken - umgesetzt.

Kritik an der digitalisierten Oberflächenskizze

Beim Review der digitalisierten Vergleichsansicht fiel uns auf, dass die Einzelansicht nicht zwingend notwendig ist. Nach Absprache mit dem Betreuer entschieden wir uns, beide Varianten für den Papierprototypen umzusetzen. Dies ermöglichte uns eine einfache und detaillierte Beurteilung der beiden Varianten.

Beim Mobiltelefonverkauf werden die Geräte oftmals in Kombination mit einem Abonnement verkauft. Die Anzeige der verschiedenen Preise, die je nach Abonnementstyp für den Gerätekauf anfallen, fehlt derzeit noch in der Vergleichstabelle.

6.2.3 *Papierprototyp*

Mit dem Papierprototypen wurden die verschiedenen Bedienungsarten getestet. Dadurch konnten wir die Oberfläche an die Bedienungsanforderungen des Benutzers anpassen.

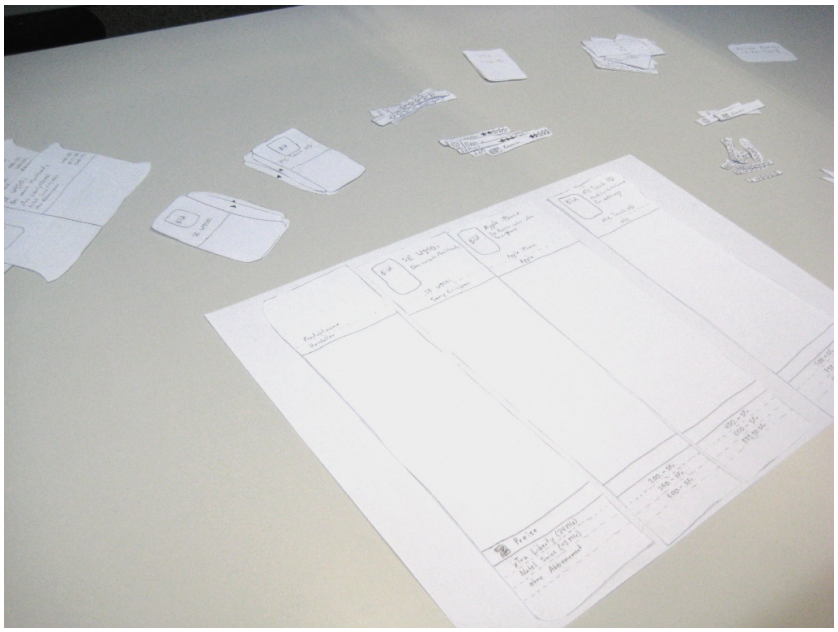


Abbildung 18: Übersicht der Papierprototyp-Elemente

Positionieren des Smartphones

Der Kunde oder Verkäufer (nachfolgend als Benutzer bezeichnet) positioniert ein Smartphone auf dem Tisch. Dabei werden nach der Erkennung Kurzinformationen zu dem Produkt und zusätzlich zwei an der Seite angeordnete, ausziehbare Klappen mit ähnlichen Smartphones angezeigt.

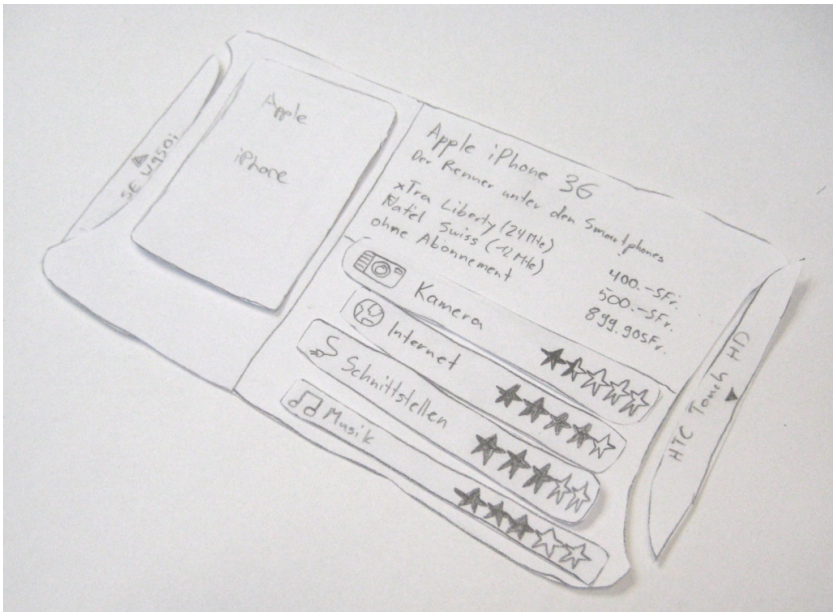


Abbildung 19: Einzelansicht nach dem Positionieren des Smartphones

Ausziehbare Klappen

Der Benutzer will die ähnlichen Smartphones mit Kurzinformationen sehen und zieht dazu die Klappen heraus. Dadurch kann er die Bewertung der Smartphones in den Kategorien vergleichen.

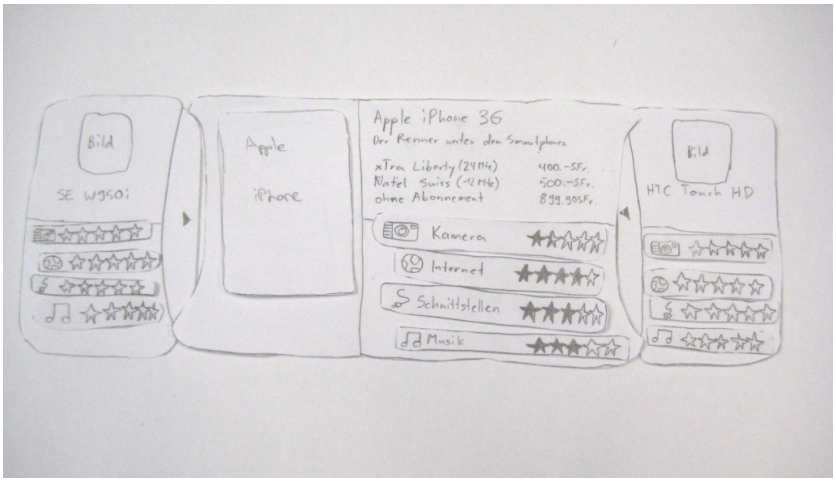


Abbildung 20: Einzelansicht nach dem Ausziehen der Klappen

Öffnen der Vergleichsansicht

Die Klappen werden danach vom Benutzer wieder eingezogen und er möchte die vollumfänglichen Informationen zum Produkt sehen. Dazu berührt er die Informationen des Smartphones zweimal schnell nacheinander, worauf die Vergleichsansicht mit dem gewählten Smartphone erscheint.

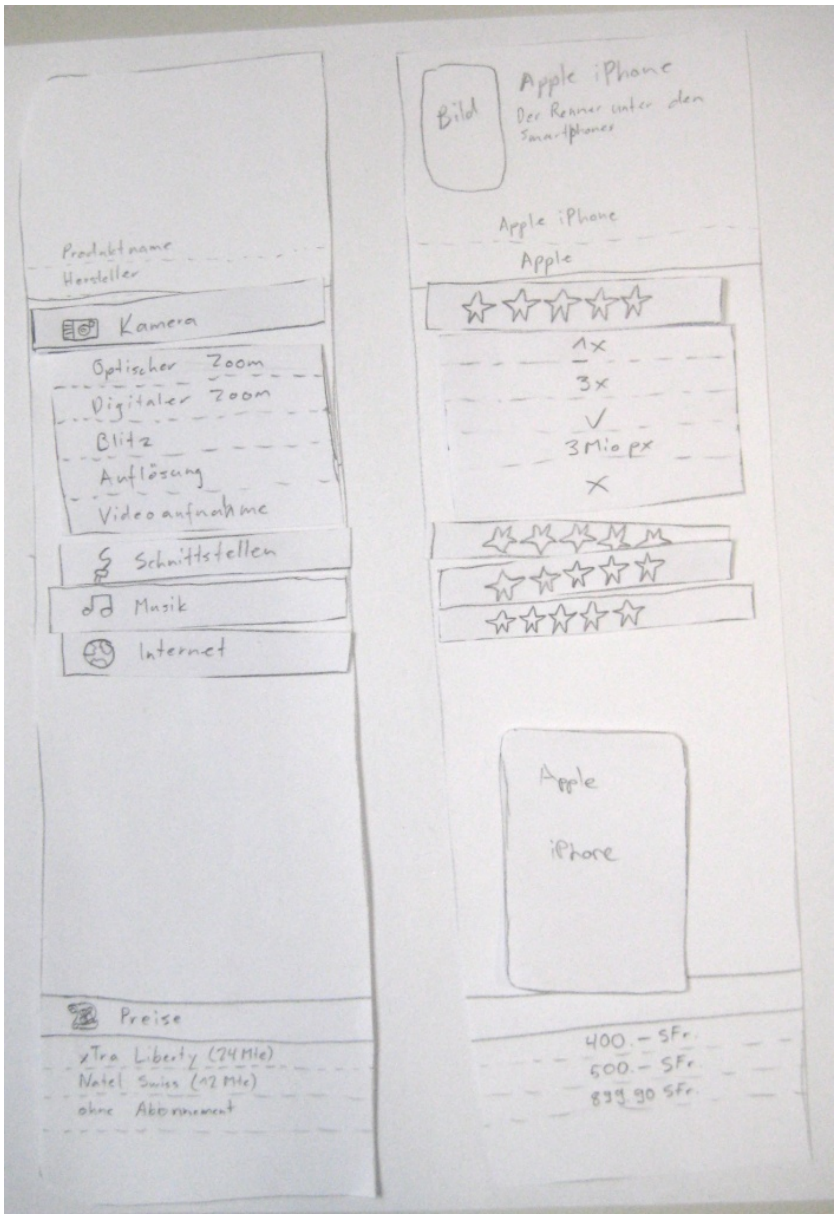


Abbildung 21: Vergleichsansicht nach dem Öffnen durch die Einzelansicht

Anzeigen von Vergleichsprodukten bei Vergleichsansicht

Auch bei der Vergleichsansicht möchte der Benutzer weitere Smartphones anzeigen lassen. Durch das Auseinanderziehen der links und rechts vom Smartphone angezeigten Zwischenräume werden zwei ähnliche Smartphones angezeigt.

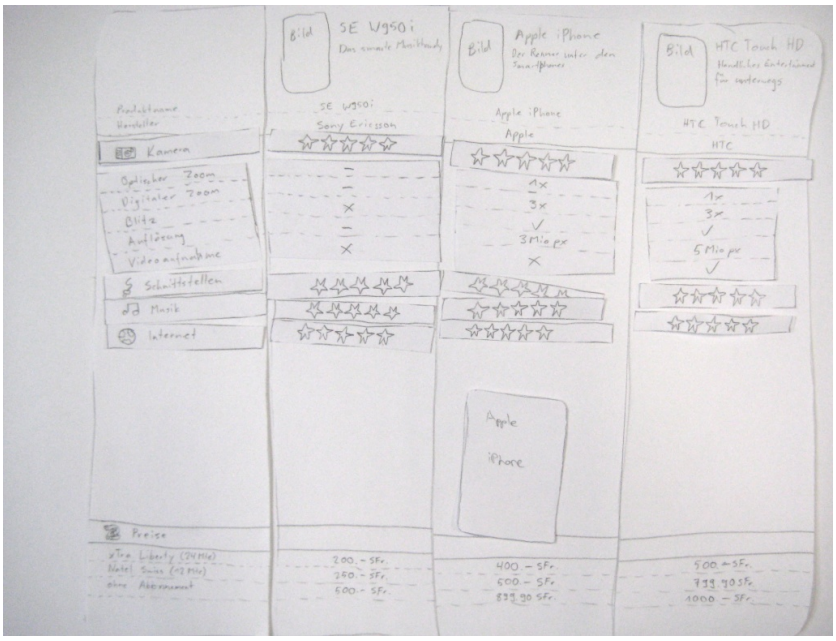


Abbildung 22: Vergleichsansicht mit Vergleichsprodukten

Vergleichskriterien ändern

Möchte der Benutzer die Prioritäten der Kategorien ändern, so kann er dies durch das Hoch- und Runterschieben der Kategorietitel erreichen.

Attribute einer Kategorie anzeigen/ausblenden

Durch die Berührung einer Kategorie kann der Benutzer diese aufklappen und die darin enthaltenen Attribute anzeigen lassen. Bei der Berührung einer anderen Kategorie wird die bisher vollständig dargestellte Kategorie wieder eingeklappt und die neu ausgewählte Kategorie mit ihren Attributen angezeigt.

Entfernen von Kategorien

Da den Benutzer die Kategorien Musik und Internet nicht interessieren, möchte er diese in der Vergleichsansicht nicht sehen. Dazu zieht er den Kategorientitel nach links an den Rand. Die Kategorie wird dadurch nur noch mit dem Titel am Rand angezeigt, während die Sterne und die Attribute nicht mehr angezeigt werden.

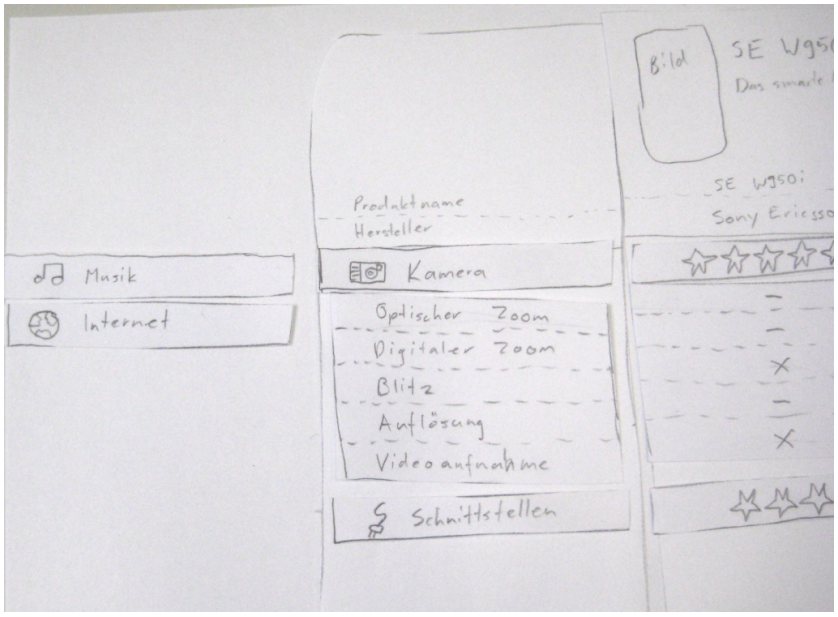


Abbildung 23: Entfernen von Kategorien

Der Benutzer könnte die Kategorie wieder anzeigen lassen, indem er die Kategorie unter- (tiefere Priorität) oder oberhalb (höhere Priorität) eines anderen Kategorientitels in die Tabelle zieht.

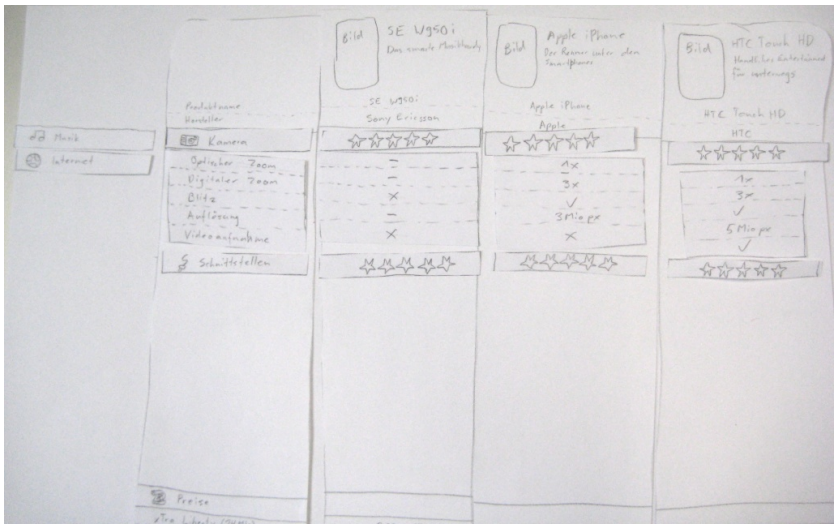


Abbildung 24: Übersicht mit entfernten Kategorien

Vergleichsprodukt wechseln

Will der Benutzer ein anderes vorgeschlagenes Smartphone anzeigen lassen, so kann er bei diesem Smartphone mit Flick nach links (oder nach rechts beim rechten Produkt) das Smartphone ändern. Zu beachten ist, dass die positionierten Smartphones nicht mit einem Flick gewechselt werden können.

Smartphones als Favorit ablegen

Möchte der Benutzer ein anderes Smartphone anzeigen lassen und das bisher angezeigte Smartphone soll später wieder zur Verfügung stehen, so kann er dieses gleich wie bei der Kategorie nach links ziehen. Durch Hereinziehen auf die gewünschte Spalte wird das Smartphone dann wieder angezeigt.

zusätzliches Smartphone auf dem Tisch

Wird vom Benutzer ein zusätzliches Smartphone auf den Tisch gelegt, so wird es je nach Positionierung links oder rechts vom bisherigen Smartphone angezeigt und das vorher in dieser Spalte dargestellte Smartphone als Favorit abgelegt.

Kritik und Ideen beim Papierprototypen

Ein Hauptpunkt bei der Auswertung des Tests war die Einzelansicht. Nach einiger Überlegung kamen wir zum Schluss, dass die Einzelansicht vorerst nicht realisiert wird. Der Grund war, dass die Einzelansicht keinen grossen Mehrwert bringt, jedoch einiges an Aufwand braucht.

Als Ersatz dient nun eine leere Ansicht, bei der ein Feld angezeigt wird, wo der Benutzer das Smartphone positionieren kann. Danach startet der Ablauf bei der Abbildung 21.

Die Startreihenfolge der Kategorien war ein weiterer Kritikpunkt. Hierbei entschieden wir, dass das auf dem Tisch positionierte Smartphone die Reihenfolge festlegt. Dabei wird die beste Kategorie des positionierten Smartphones zuoberst angezeigt.

Es wurden auch Ideen für Effekte gesammelt, da diese als Benutzerrückmeldung dienen. Folgende Effekte wurden dabei ausgewählt:

- Für ziehbare Elemente: Wenn der Benutzer drauf bleibt, wackelt das Element kurz.
- Für das Wechseln der Smartphones durch Flick: Die Anzeige des Smartphones rollt weiter und die nächste Smartphone-Anzeige rollt rein. War es das letzte vorgeschlagene Smartphone, so rollt die Anzeige ein wenig weiter und rollt dann wieder zurück.

6.3 FINALE BENUTZEROBERFLÄCHE

6.3.1 *Änderungen gegenüber dem Papierprototypen*

Drehrad

Im Verlaufe der Entwicklung wurde der Wechsel zwischen den vorgeschlagenen Smartphones mittels eines Flicks wieder verworfen. Im Gegenzug werden die Smartphones in einem Drehrad am rechten Rand der Oberfläche positioniert. Dasselbe Rad wird auch für die verdrängten Smartphones verwendet. Dieses wird jedoch am linken Rand positioniert.

Abbildung 25: Drehrad

Aktive Kategorie

Die aktive Kategorie wird zusätzlich eingefärbt. Damit kann diese besser von den restlichen Kategorien unterschieden werden.

Display	★☆☆☆☆	★★★☆☆	★★★★☆
Auflösung	240 x 320 Pixel	640 x 360 Pixel	320 x 480 Pixel
Handschrifterkennung	✗	✓	✗
Multitouch	✗	✗	✓
Stifteingabe	✗	✓	✗
Touchscreen	✗	✓	✓

Abbildung 26: Aktive Kategorie

Kontrollbuttons

Die Darstellung der Applikation kann gedreht und zurückgesetzt werden. Ursprünglich war geplant, diese Aktionen durch Gesten auszulösen. Aus zeitlichen Gründen wurde diese Idee verworfen und es wurden am linken unteren Rand zwei Buttons eingefügt.

Abbildung 27: Kontrollbuttons

6.3.2 *Bildschirmauszug der finalen Benutzeroberfläche*

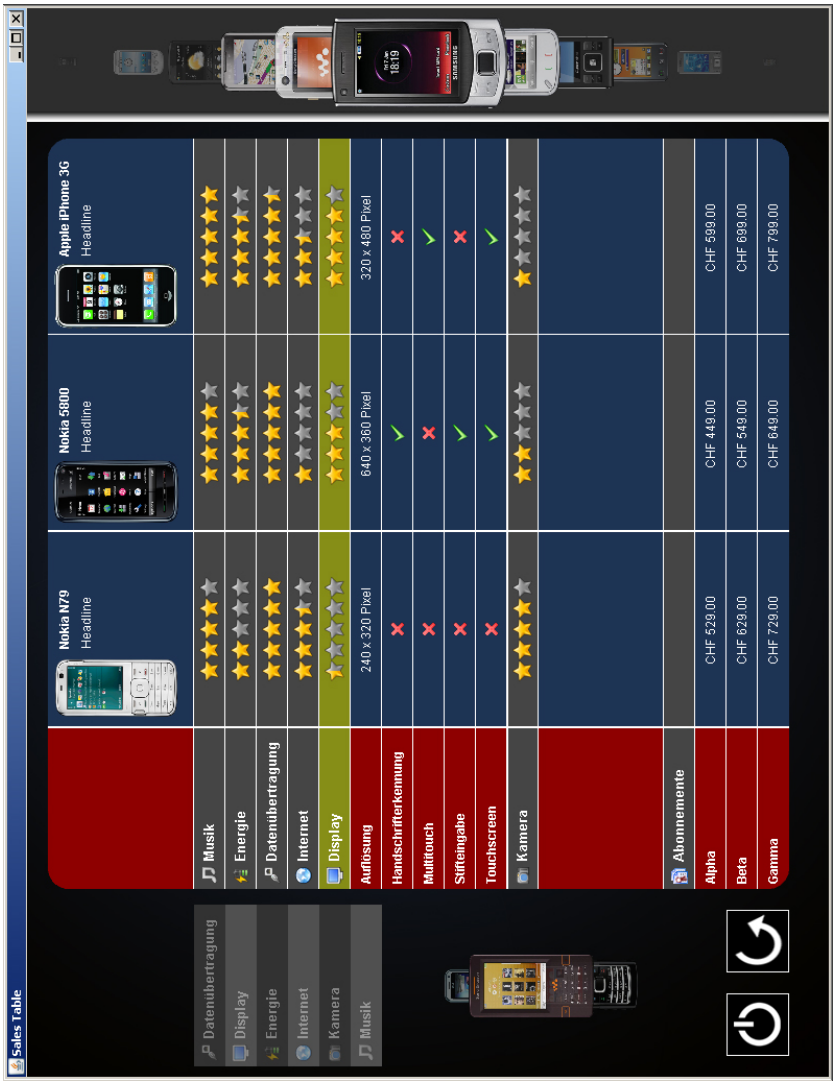


Abbildung 28: Screenshot der Applikation

7 DOMAINANALYSE

Die Domainanalyse wurde nach Larman [6] durchgeführt.

7.1 DOMAINMODELL

7.1.1 *Strukturdiagramm*

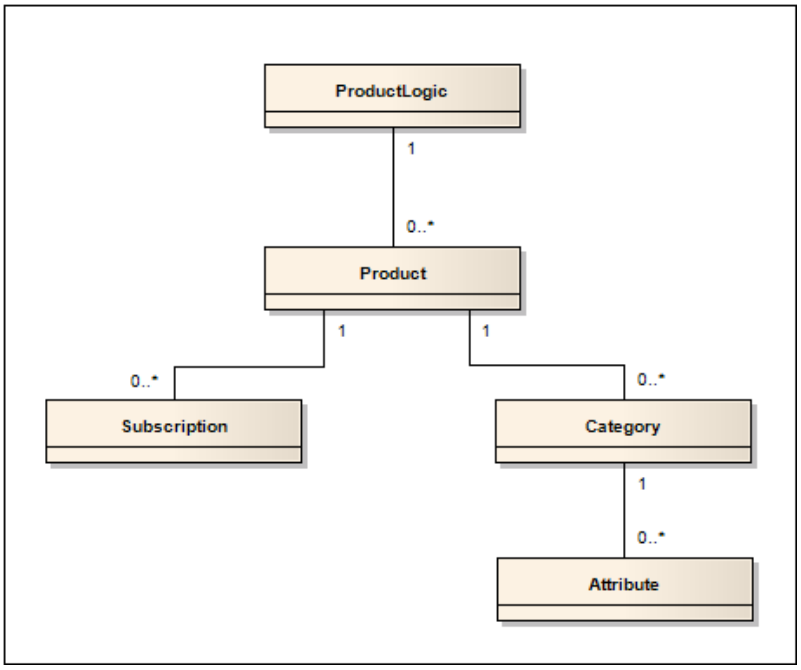


Abbildung 29: Domain Model

7.1.2 *Konzeptbeschreibung*

Das Domain Model wurde entworfen um die Datenschicht zu vereinfachen. Dadurch wird die Kommunikation mit der Benutzeroberfläche deutlich vereinfacht.

Kurzbeschreibung der Elemente des Domain Models

PRODUCTLOGIC Die Klasse ProductLogic bildet die Schnittstelle zu den weiteren Schichten. Sie enthält die anzeigbaren Smartphones und figuriert als Facade. Die oberen Schichten kommunizieren über diese Klasse mit dem restlichen Modell.
Die ProductLogic besitzt eine Beziehung zu der Klasse Product (den Smartphones).

PRODUCT Bei der Klasse Product handelt es sich um das Produkt selbst. Über diese Klasse können die dazugehörigen Kategorien und Abonnemente ausgelesen werden.
Die Klasse Product besitzt je eine Beziehung zu den Abonnementen (Subscription) und den Kategorien (Category).

SUBSCRIPTION In der Klasse Subscription wird das Abonnement für ein Smartphone abgebildet. Sie bildet ein reines Datenobjekt und dient zur einfacheren Verwaltung der Abonnemente.

CATEGORY Die Klasse Category dient zur Unterteilung der verschiedenen Attribute. Sie enthält eine Punktzahl, welche uns zu Vergleichszwecken dient. Die Category besitzt eine Beziehung zu der Klasse Attribute.

ATTRIBUTE Bei der Klasse Attribute handelt es sich um ein reines Datenobjekt, welches jeweils eines der variablen Attribute abbildet. Sie wurde zur einfacheren Verwaltung eingeführt.

7.2 DATENBANKMODELL

Bei der Realisierung der Datenhaltung wird auf ein Datenbanksystem zurückgegriffen. Als Alternative stand die Speicherung der Daten in XML-Dateien oder die direkte Speicherung in Java-Objekten zur Verfügung. Eine einfach erweiterbare Struktur gab den Ausschlag für die Wahl des Datenbanksystems.

7.2.1 *Strukturdiagramm*

Das in Abbildung 30 dargestellte Strukturdiagramm stellt den Aufbau der Datenbank dar. Die im Modell enthaltenen Tabellen werden im Rahmen dieses Kapitels näher erläutert. Um die Datenstruktur zu verdeutlichen, wurde für jede Tabelle mindestens ein Beispieleintrag erstellt.

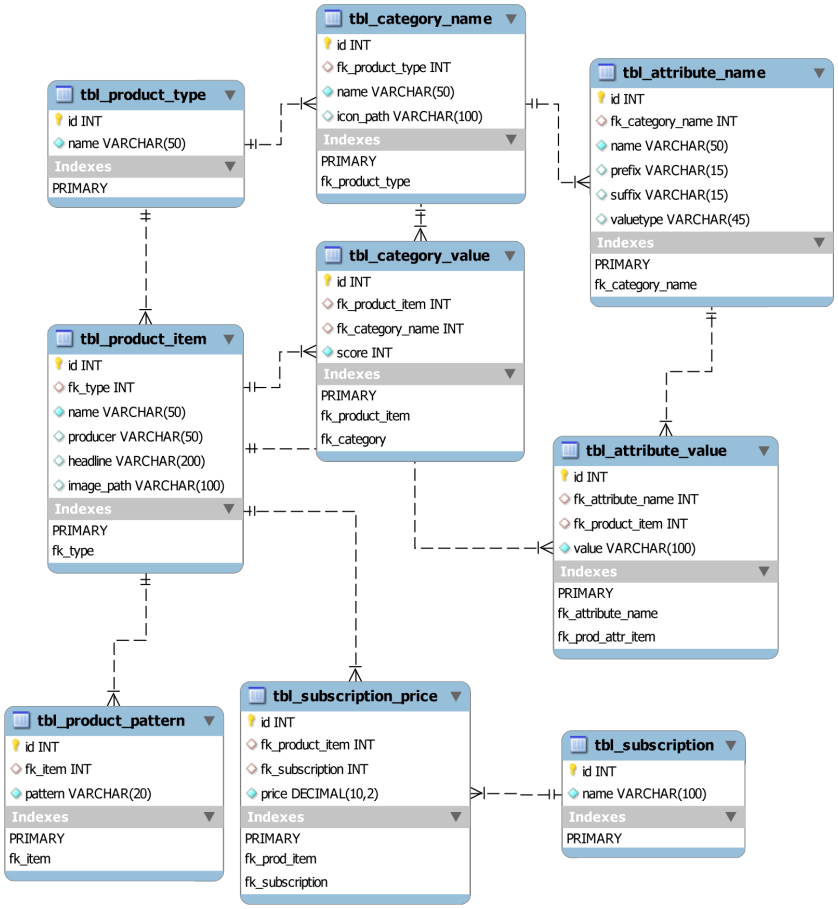


Abbildung 30: Strukturdiagramm Datenbankmodell

7.2.2 *Tabelle Product Type*

Der Produkttyp ermöglicht die Unterscheidung verschiedener Gerätetypen. So können beispielsweise Digitalkameras wie auch Smartphones unterschiedliche Kategorien und Attribute aufweisen. Da im Rahmen dieser Arbeit lediglich mit Smartphones gearbeitet wird, ist diese Tabelle nicht zwingend erforderlich. Aus Gründen der leichten Erweiterbarkeit wurde sie dennoch dem Datenbankmodell hinzugefügt.

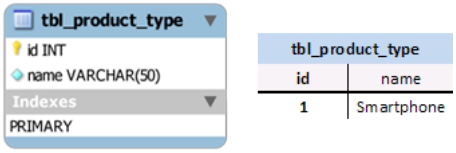


Abbildung 31: Tabelle tbl_product_type mit Beispiel

7.2.3 *Tabelle Category Name*

Jedem Produkttyp sind verschiedene Kategorien zugeordnet. Diese Kategorien ermöglichen eine Gliederung der Geräteeigenschaften (Attribute). In dieser Tabelle wird zusätzlich über den Pfad zum Bild ein Icon definiert.

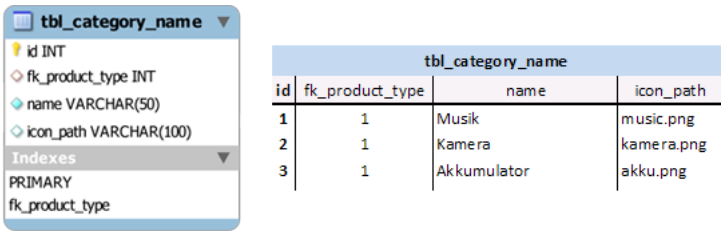


Abbildung 32: Tabelle tbl_category_name mit Beispiel

7.2.4 *Tabelle Attribute Name*

Jeder Kategorie sind verschiedene Attribute zugeordnet. In dieser Tabelle sind sämtliche Namen aller Attribute vermerkt. Dies ermöglicht eine dynamische Struktur der Attribute und der Verwaltungsaufwand einer statischen Attributverwaltung entfällt. Über das Feld «valuetype» wird der Typ des Attributes festgelegt, was die Unterscheidung in der Programmlogik vereinfacht.

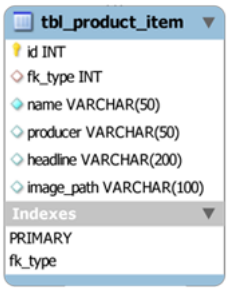
tbl_attribute_name

id	fk_category_name	name	prefix	suffix	valuetype
1	1	Unterstützt Playlists			Boolean
2	2	Digitaler Zoom		x	Boolean
3	2	Optischer Zoom		x	Boolean
4	3	max. Bereitschaftszeit		h	Float

Abbildung 33: Tabelle tbl_attribute_name mit Beispiel

7.2.5 *Tabelle Product Item*

In dieser Tabelle werden sämtliche Produkte gespeichert. Sie besitzen eine Referenz zu ihrem Produkttyp, um die Verbindung zu den Attributen der entsprechenden Kategorien herzustellen. Neben dem Produktnamen wird auch der Produkthersteller und ein Werbespruch (headline) gespeichert. Prinzipiell wäre es möglich diese Attribute in die Tabelle «Attribute Name» auszulagern. Da diese Attribute jedoch bei jedem Produkt vorhanden sein können, wurden sie direkt in die Tabelle «Product Item» integriert. Zusätzlich wird die Abbildung des Produktes als Pfad zur Bilddatei gespeichert.



tbl_product_item

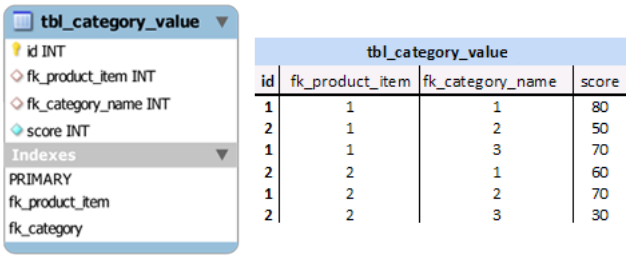
id	fk_type	name	producer	headline	image_path
1	1	iPhone 3G	Apple	Headline	iphone.png
2	2	w950i	Sony Ericsson	Headline	w950i.png

Abbildung 34: Tabelle tbl_product_item mit Beispiel

7.2.6 *Tabelle Category Value*

Jeder Produktkategorie wird eine Bewertung (score) zugewiesen. Ursprünglich war eine Berechnung der Bewertung anhand der Kategorieattribute geplant. Da in dieser Arbeit der Fokus jedoch auf der Gestaltung der Benutzeroberfläche und nicht auf der Entwicklung komplexer Bewertungsalgorithmen liegt, wurde nach Absprache mit dem Betreuer die Bewertung als statischer Wert in die Datenbankstruktur integriert.

Durch die Verknüpfung des «Product Item» und des «Category Name» kann die Bewertung eindeutig zugewiesen werden.



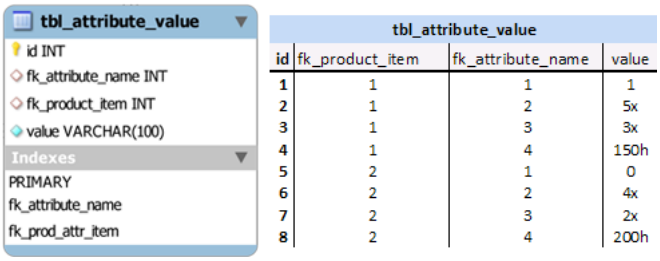
tbl_category_value

id	fk_product_item	fk_category_name	score
1	1	1	80
2	1	2	50
1	1	3	70
2	2	1	60
1	2	2	70
2	2	3	30

Abbildung 35: Tabelle tbl_category_value mit Beispiel

7.2.7 *Tabelle Attribute Value*

Jedem Attribut kann exakt ein Wert pro Produkt zugewiesen werden. Die Zuweisung erfolgt dabei ähnlich wie bei der Tabelle «Category Value» über eine Indextabelle. Zu beachten ist, dass sämtliche Werte als VARCHAR[1] abgespeichert werden. Der Typ des Attributes wird in der Tabelle «Attribute Name» festgelegt.



tbl_attribute_value

id	fk_product_item	fk_attribute_name	value
1	1	1	1
2	1	2	5x
3	1	3	3x
4	1	4	150h
5	2	1	0
6	2	2	4x
7	2	3	2x
8	2	4	200h

Abbildung 36: Tabelle tbl_attribute_value mit Beispiel

1 Ein Varchar oder Variable Character Field ist eine Menge von Zeichen unbestimmter Länge
http://en.wikipedia.org/wiki/Varchar

7.2.8 *Tabelle Subscription*

Mobiltelefone können oftmals nur erworben werden, wenn gleichzeitig ein Abonnement abgeschlossen wird. Da sich je nach Abonnement die Preise des Produkts unterscheiden, müssen sämtliche Abonnements gespeichert werden.

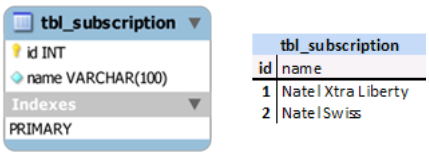


Abbildung 37: Tabelle tbl_subscription mit Beispiel

7.2.9 *Tabelle Subscription Price*

In dieser Tabelle werden die Abonnements mit dem Produkt verknüpft. Je nach Abonnement kann ein anderer Preis bestimmt werden.

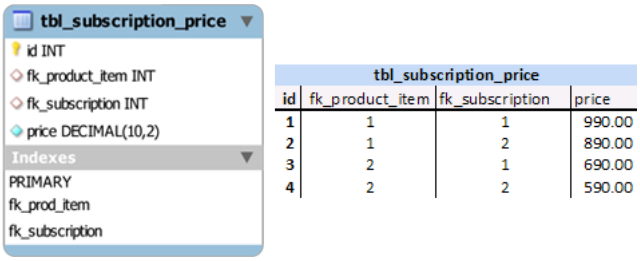


Abbildung 38: Tabelle tbl_subscription_price mit Beispiel

7.2.10 *Tabelle Product Pattern*

In der Tabelle wird das Muster der Pinanordnung gespeichert. Das Muster wird dabei als Binärcode abgespeichert. Da dies führende Nullen zur Folge haben kann (z.B. «010101»), wurde ein VARCHAR verwendet.

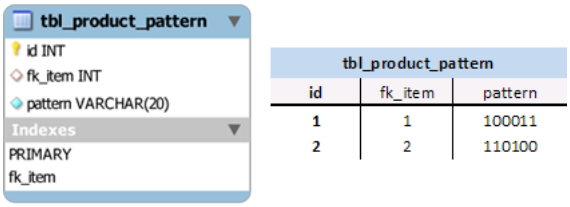


Abbildung 39: Tabelle tbl_product_pattern mit Beispiel

7.3 SYSTEM SEQUENZDIAGRAMME

Bei den folgenden Beschreibungen der Sequenzdiagramme wird der Kunde oder Verkäufer als Benutzer bezeichnet.

7.3.1 *SSD01 - Produkt platzieren*

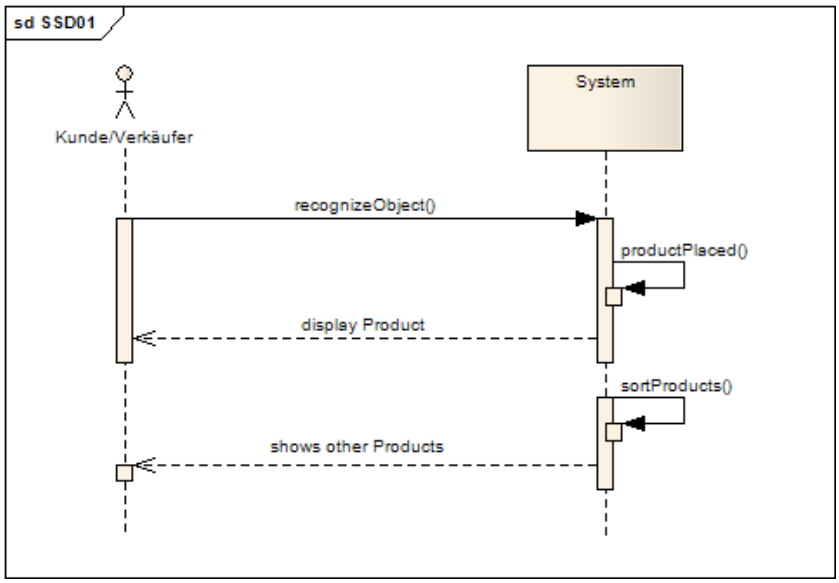


Abbildung 40: SSD01 - Produkt platzieren

Der Benutzer platziert ein Smartphone auf dem Tisch, wodurch eine Methode «recognizeObject» ausgeführt wird. Während dieser Methode wird vom System eine Erkennung des Smartphones durchgeführt. Wurde das Smartphone erkannt, werden anschliessend die weiteren Smartphones nach ihrer Bewertung rangiert. Sobald diese Methoden ausgeführt wurden, wird das Smartphone mit seinen Detailinformationen angezeigt. Die weiteren Smartphones werden dabei auch angezeigt.

7.3.2 *SSD02 - Kategorien festlegen*

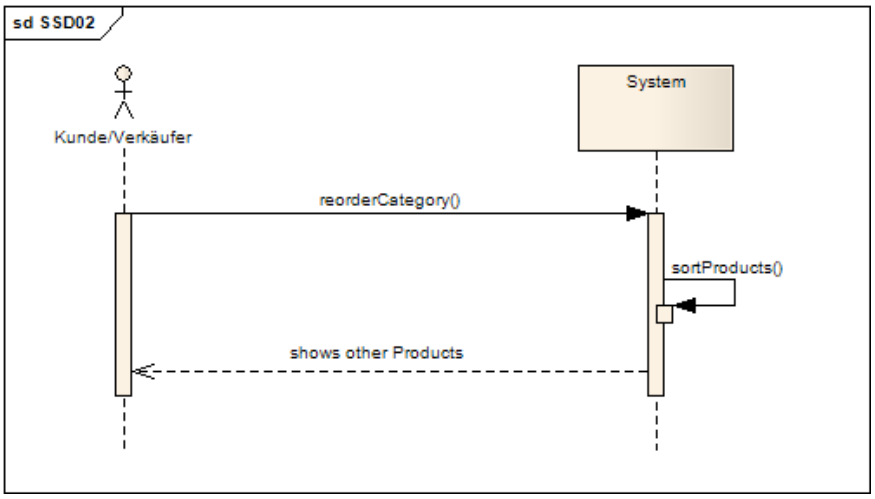


Abbildung 41: SSD02 - Kategorien festlegen

Der Benutzer verändert die Gewichtung der Kategorien. Dies hat zur Folge, dass die Methode «reorderCategory» ausgeführt wird. Das System wird dabei angewiesen, die weiteren Smartphones neu zu sortieren.

7.3.3 *SSD03 - Position ändern*

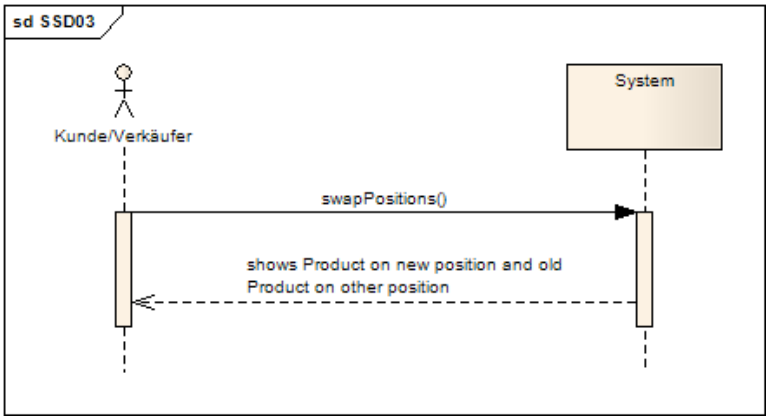


Abbildung 42: SSD03 - Position ändern

Der Benutzer verändert die Position des Smartphones. Das System verschiebt danach das Smartphone an die neue Position. Das bisher dort angezeigte Smartphone wird danach an die alte Position des verschobenen Smartphone gesetzt.

7.3.4 *SSD04 - weiteres Produkt platzieren*

Das Systemsequenzdiagramm SSD04 entspricht dem des SSD01. Die Sortierung der Produkt entfällt jedoch hierbei.

7.3.5 *SSD05 - Produkt entfernen*

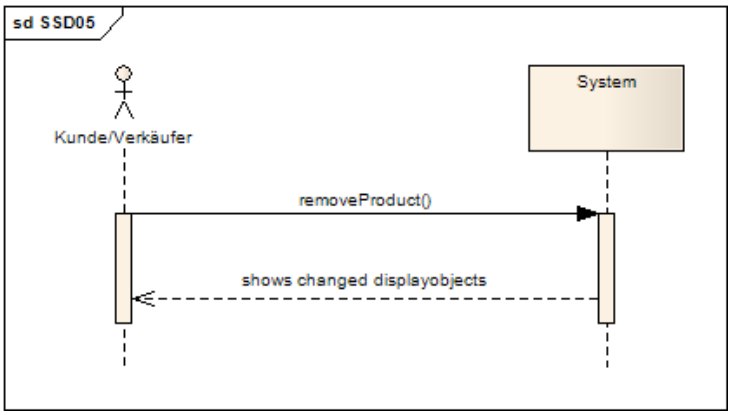


Abbildung 43: SSD05 - Produkt entfernen

Der Benutzer entfernt ein Smartphone aus der Vergleichstabelle. Die Spalte in der Tabelle wird dabei vom System entfernt.

7.3.6 *SSD06 - Kategorie entfernen*

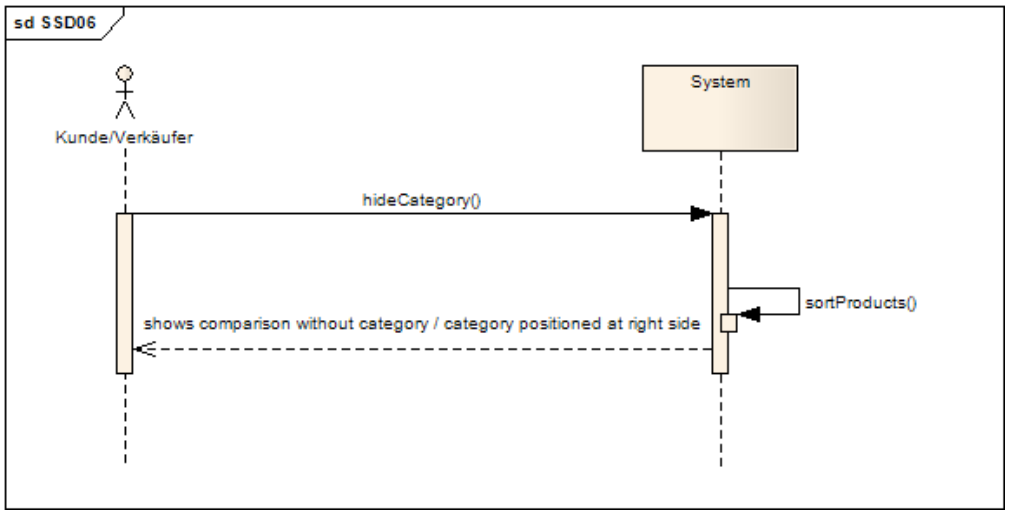


Abbildung 44: SSD06 - Kategorie entfernen

Der Benutzer verschiebt eine Kategorie aus der Vergleichstabelle. Das System stellt diese Kategorie danach am Rand dar und entfernt es von der Vergleichstabelle.

7.3.7 *SSD07 - Kategorie hinzufügen*

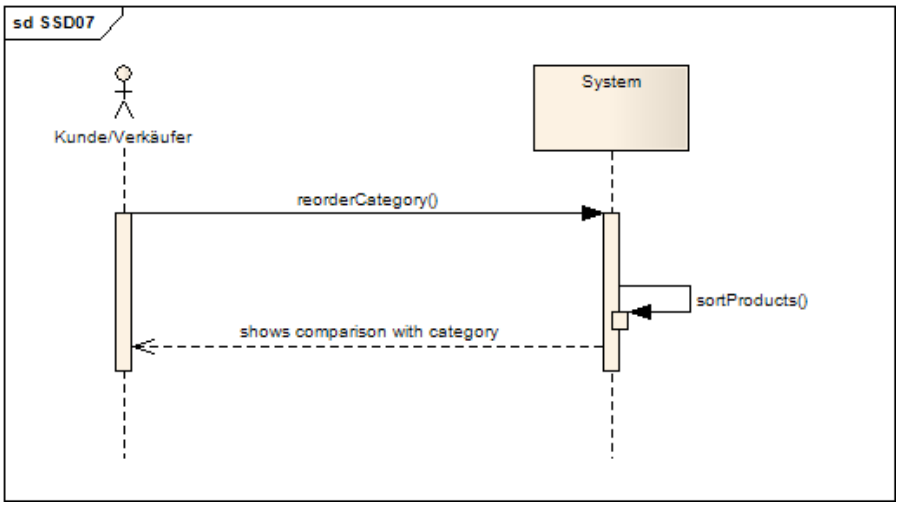


Abbildung 45: SSD07 - Kategorie hinzufügen

Der Benutzer fügt eine Kategorie vom Rand zur Vergleichstabelle hinzu. Das System entfernt die Anzeige der Kategorie vom Rand und verschiebt sie in die Vergleichstabelle.

7.3.8 *SSD08 - Oberfläche drehen*

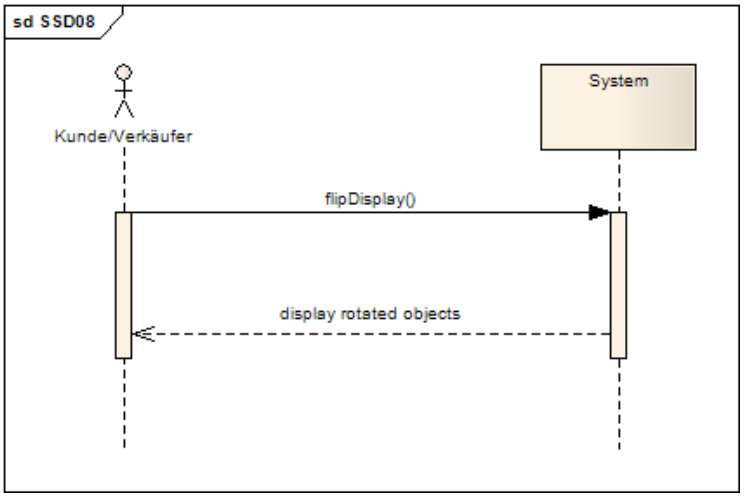


Abbildung 46: SSD08 - Oberfläche drehen

Der Benutzer weist das System an, die Oberfläche zu drehen. Das System dreht dabei alle Elemente auf der Oberfläche um 180 Grad.

7.3.9 *SSD09 - Oberfläche zurücksetzen*

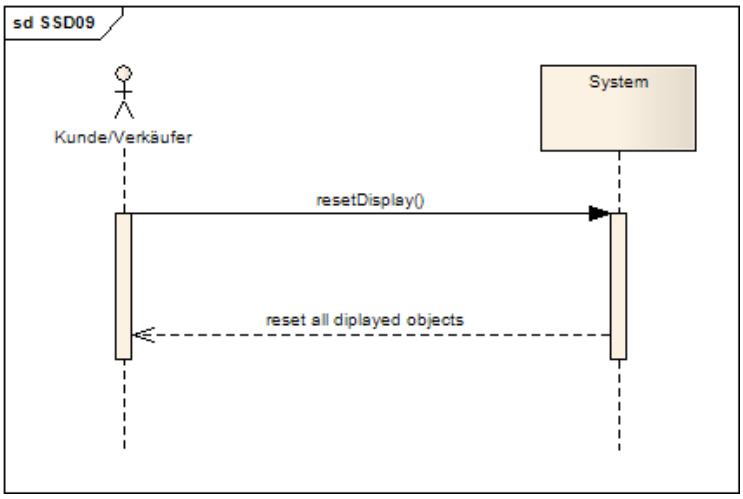


Abbildung 47: SSD09 - Oberfläche zurücksetzen

Der Benutzer weist das System an, die Oberfläche zurückzusetzen. Das System sorgt dafür, dass alle Elemente auf den Anfangszustand zurückgesetzt werden und die richtigen Darstellungselemente ausgeblendet bzw. eingeblendet werden.

8 ENTWURF

8.1 SCHICHTENARCHITEKTUR

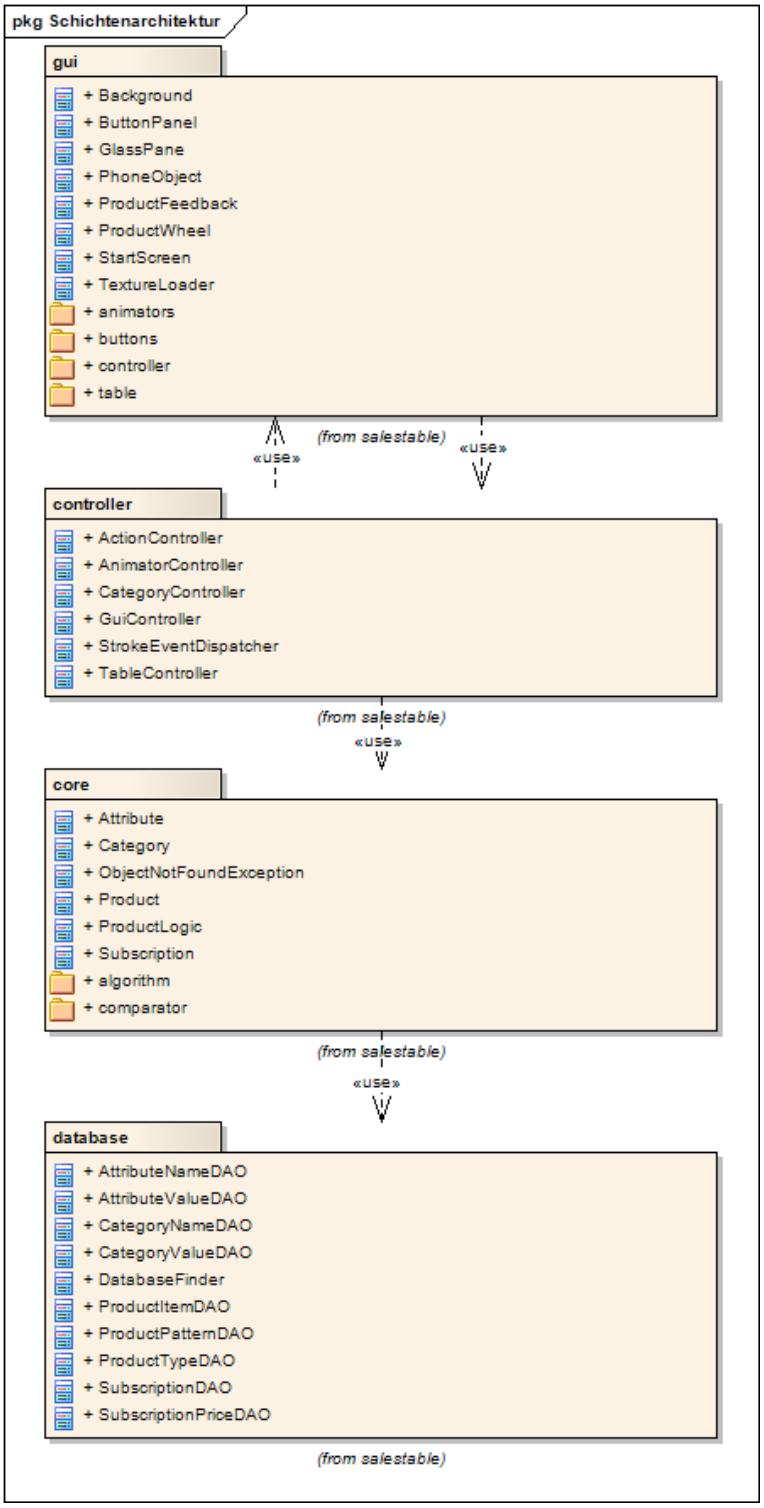


Abbildung 48: Schichtenarchitektur

Zum Einsatz kam das MVC-Konzept

Die Architektur der Applikation entspricht dem MVC-Konzept (Model-View-Controller). Neben den drei Schichten Model, View und Controller wurde zusätzlich die Datenschicht eingeführt. Dies ermöglichte die Aufbereitung der Daten für die Objektorientierte Programmierung.

Die Schichten stützen sich jeweils auf die darunterliegende Schicht. Durch die Animationen der grafischen Elemente wurde es erforderlich, dass die Controller die Darstellungsschicht ansteuern können müssen, was dazu führte, dass die Controller mit der oberen und unteren Schicht kommunizieren. Da die Aktionen, welche zu den Animationen führen, über die Darstellungsschicht ausgelöst werden, müssen diese auch die Controller ansprechen können. Diese zirkuläre Abhängigkeit wurde bestmöglich über das Observer-Pattern gelöst. Bei einigen Aktionen wäre der Aufwand jedoch zu hoch gewesen, weshalb bei diesen die Controller über ein Singleton eines Controllers (GuiController) ansprechbar sind. Dieses Singleton wurde bereits bei den anderen Arbeiten, die auf dem Multi-Touch-Tisch ausgeführt wurden, erfolgreich eingesetzt.

Die weiteren Funktionalitäten der einzelnen Schichten werden im Verlauf dieses Kapitels erklärt.

SCHICHT	FUNKTION
gui	Darstellung der grafischen Elemente und Entgegennahme von Benutzerinteraktionen
controller	Behandlung der Benutzerinteraktionen und Integration der Daten in die grafischen Elemente
core	Abbildung der Daten auf ein logisches Modell
data	Abstraktion der Daten von der Datenbank

Tabelle 24: Funktion der Schichten

8.1.1 *Verwendete Java-Bibliotheken*

BIBLIOTHEK	FUNKTION / WEBSEITE
Hibernate 3	Hibernate ist ein Open-Source-Persistenz-Framework für Java. https://www.hibernate.org
MySQL-Connector	Wird für die Verbindung zur MySQL-Datenbank gebraucht. http://dev.mysql.com/usingmysql/java/
JOGL	Schnittstelle zur OpenGL-API https://jogl.dev.java.net/

Tabelle 25: Verwendete Java-Bibliotheken

8.2 ARCHITEKTURKONZEPTE

8.2.1 *Speicherverwaltung*

Die Speicherverwaltung wird über eine Datenbank realisiert. Durch Hibernate[1] wird der Zugriff auf die Datenbank vereinfacht. Da bei unserer Arbeit nur das Lesen von Daten im Mittelpunkt stand, wurde auf das Schreiben in die Datenbank verzichtet. Für eine zukünftige Implementation dieser Funktion sollte dies dank der Verwendung von Hibernate jedoch sehr einfach erreichbar sein.

8.2.2 *Fehlerbehandlung*

Die Fehlerbehandlung wird in der Controllerschicht erledigt. Nachfolgend werden die möglichen Fehlerquellen beschrieben:

Bei der Datenschicht wird jeweils bei Problemen mit der Datenbank (z.B. wenn die Datenbank nicht verfügbar ist) eine Exception geworfen, welche nicht abgefangen werden konnte. Sollten die Daten nicht verfügbar sein, ist dadurch die Arbeit mit der Applikation jedoch sowieso unmöglich.

In der Modelschicht wird bei Nichtauffinden von Objekten eine ObjectNotFoundException geworfen. Dadurch kann auf diese Fehler jeweils reagiert werden.

8.2.3 *Methodenaufrufe über Reflection*

Bei Animationen ist es nicht möglich gewisse Methoden nach dem Ablauf einer Animation auszuführen, da die Animationen in einem eigenen Thread ablaufen müssen. Deshalb haben wir diese Aufrufe mithilfe von Reflection gelöst. Die Methodenaufrufe wurden dabei wie die Animation selber dem EventDispatcher hinzugefügt. Nachdem der EventDispatcher die Animation fertig ausgeführt hat, wird die Methode dank der mitgegebenen Argumente über eine Klasse MethodInvoker, die die Reflection-Funktionalität enthält, ausgeführt.

8.2.4 *Einsatz des Observer Patterns*

Das Observer Pattern kam bei dieser Applikation mehrmals zum Einsatz. Nachfolgend die drei wichtigsten Beispiele für die Verwendung des Observer Patterns:

- Der EventDispatcher verwendet das Pattern um benachrichtigt zu werden, wenn ein Animationsthread seine Arbeit erledigt hat.
- Die TableCell benutzt das Pattern, wodurch der Controller über das Hilfsobjekt TableNotfier benachrichtigt wird, wenn der Benutzer eine neue Aktion auf der TableCell ausführt bzw. ausgeführt hat.

1 Hibernate ist ein Open-Source-Persistenz-Framework für Java. Das Framework ermöglicht es, den Zustand eines Objekts in einer relationalen Datenbank zu speichern und aus entsprechenden Datensätzen wiederum Objekte zu erzeugen, ohne die DB-Zugriffe explizit in SQL programmieren zu müssen.
http://de.wikipedia.org/wiki/Hibernate_(Framework)

- Auch die Erkennung der Produkte wird mittels des Observer Patterns gelöst, da die Aufnahme der Eingabepins des Produktes in einem eigenen Thread läuft. Dadurch muss der Thread dem überwachenden Element mitteilen, wann die Aufnahme beendet ist.

8.2.5 *Einsatz des Strategy Patterns*

In dieser Applikation kam das Strategy Pattern für die verschiedenen Animationen zum Einsatz. Da die Animationen den EventDispatcher jedoch bei Beendigung benachrichtigen mussten, haben wir diese gemeinsame Funktionalität der verschiedenen Animationen als abstrakte Klasse und nicht als Interface, wie dies beim klassischen Strategy Pattern definiert ist, erstellt.

Der CellContent setzt das Strategy Pattern ein

Ein weiterer Einsatzort des Strategy Pattern ist der CellContent. Hier wird von den abgeleiteten Klassen die Höhe und Breite jeweils unterschiedlich berechnet. Der CellContent musste aber wiederum als abstrakte Klasse erstellt werden, da die abgeleiteten Objekte zwingend vom Typ SceneGraphObject[2] sein müssen. Das SceneGraphObject selbst ist eine Klasse aus dem zur Verfügung gestellten Framework, welche nicht ohne weiteres verändert werden kann.

8.2.6 *Einsatz des Singleton Patterns*

Der GuiController stellt in dieser Applikation eine Anwendung des Singleton Patterns dar. Die bisher mit dem Framework erstellten Applikationen zeigten uns, dass dies die beste Lösung für diese Art von Applikation ist. Durch den Einsatz des Singleton Patterns konnten auch einige Referenzen auf Objekte verhindert werden, welche ansonsten einen Mehraufwand für die Referenzverwaltung bedeuten würde.

2 Das SceneGraphObject wird vom Charger-Framework zur Verfügung gestellt und bietet grundlegende Funktionen für die Objektmanipulation

8.3 DATENSCHICHT

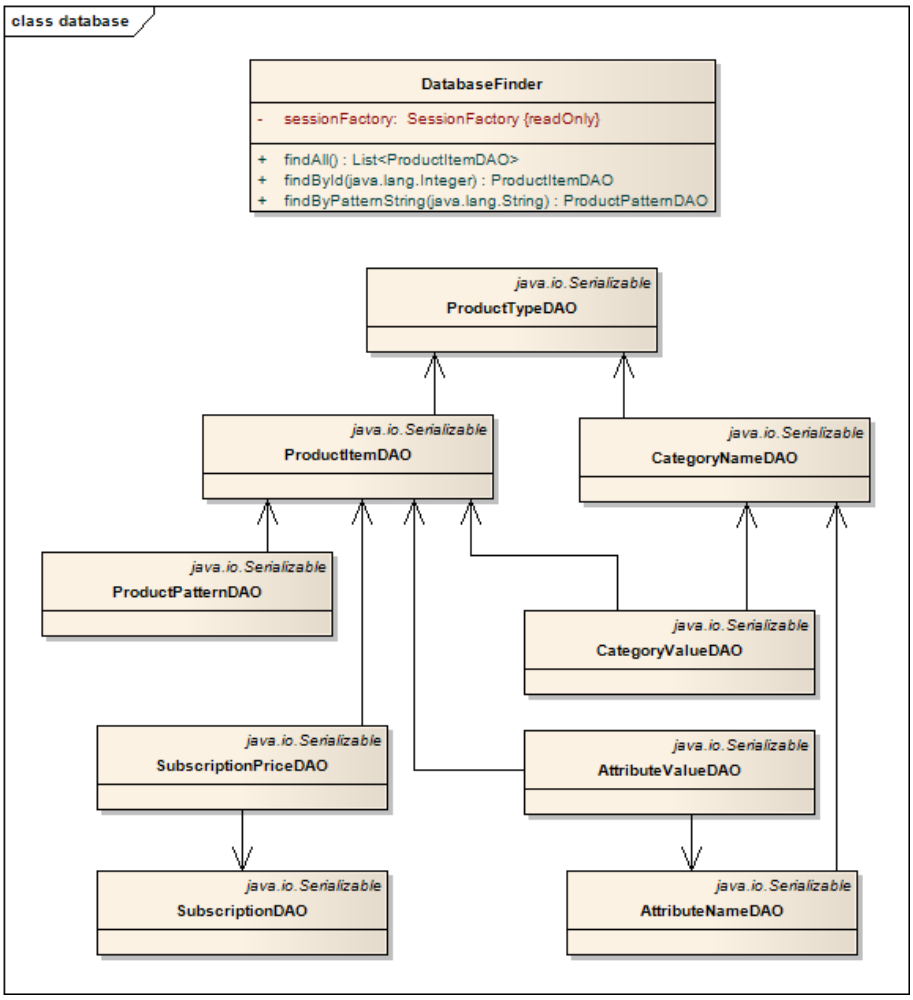


Abbildung 49: Klassendiagramm Datenbankschicht

Die Datenschicht dient zur Abstraktion der Daten von der darunterliegenden Datenbank. Bis auf die Klasse DatabaseFinder wurden sämtliche Klassen in dieser Schicht generiert.
Die Klasse DatabaseFinder dient als Schnittstelle für die oberen Schichten und bietet Methoden an um die gewünschten Daten zu erhalten. Der Zugriff erfolgt hierbei durch die Modelschicht.

8.3.1 *Schnittstellen*

KLASSENNAME	BESCHREIBUNG
DatabaseFinder	Die Kommunikation mit der Datenschicht wird über die Klasse DatabaseFinder abgefertigt, wodurch sie als Facade fungiert. Sie wird von der Klasse ProductLogic aus der Modelschicht erstellt und auch verwendet. Der DatabaseFinder kümmert sich dabei um den Verbindungsaufbau mit der Datenbank und gibt bei den Methodenaufrufen die gewünschten Datenobjekte zurück.

Tabelle 26: Schnittstellen der Datenschicht

8.4 MODELSCHICHT

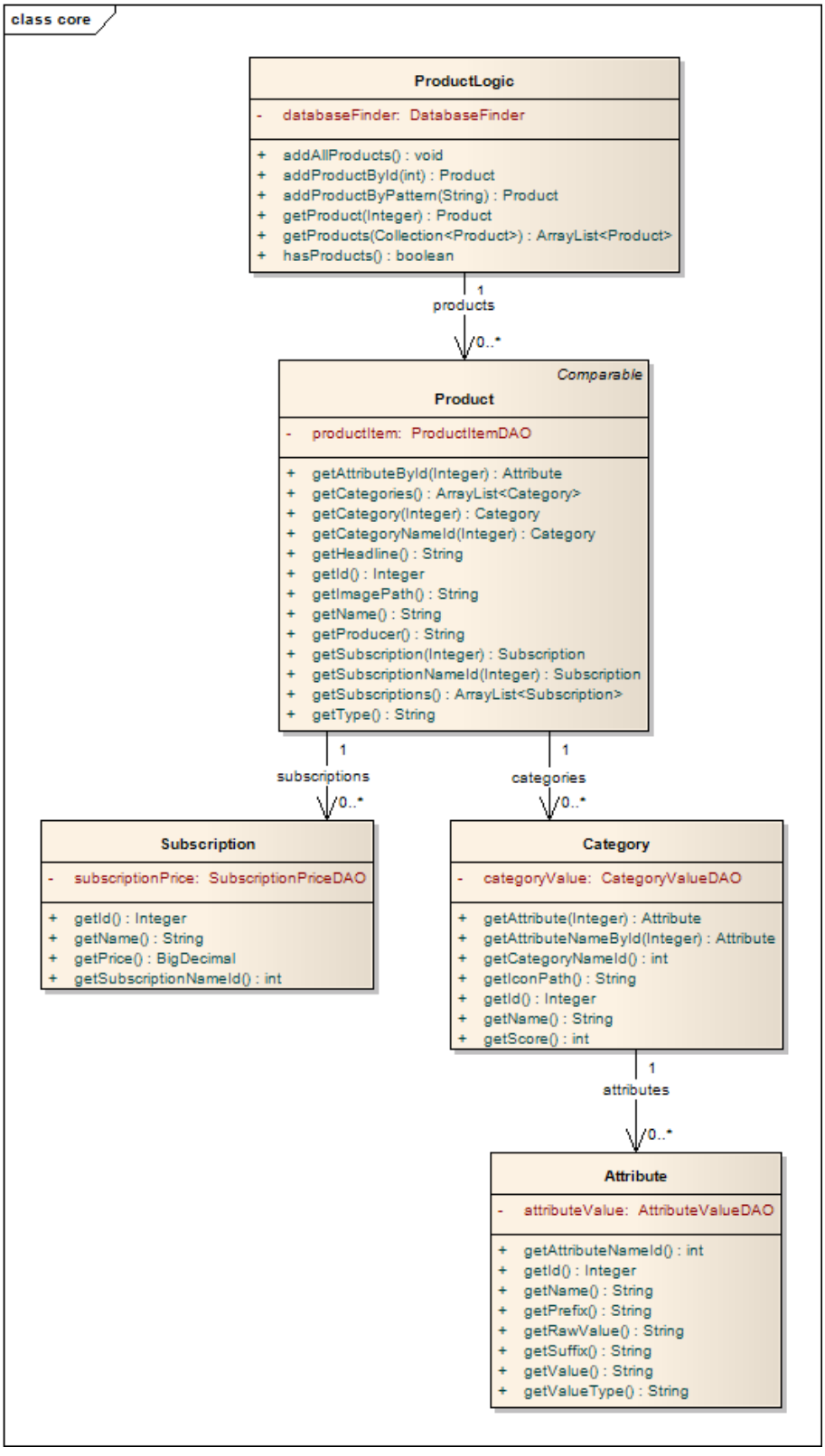


Abbildung 50: Klassendiagramm Modelschicht

Die Modelschicht dient der Abbildung der Realität und bringt die Daten in eine sinnvolle Struktur. Damit stellt die Modelschicht die Problem Domain der Applikation dar. Die weiteren Details dazu wurden bereits im Kapitel 7 erwähnt.

8.4.1 *Schnittstellen*

KLASSENNAME	BESCHREIBUNG
ProductLogic	Über die Klasse ProductLogic können die anderen Schichten auf die einzelnen Produkte zugreifen, welche Vergleichsmöglichkeiten und verschiedene Daten enthalten. Die ProductLogic wird über die Klasse GuiController referenziert und verwendet.

Tabelle 27: Schnittstellen der Modelschicht

8.5 CONTROLLERSCHICHT

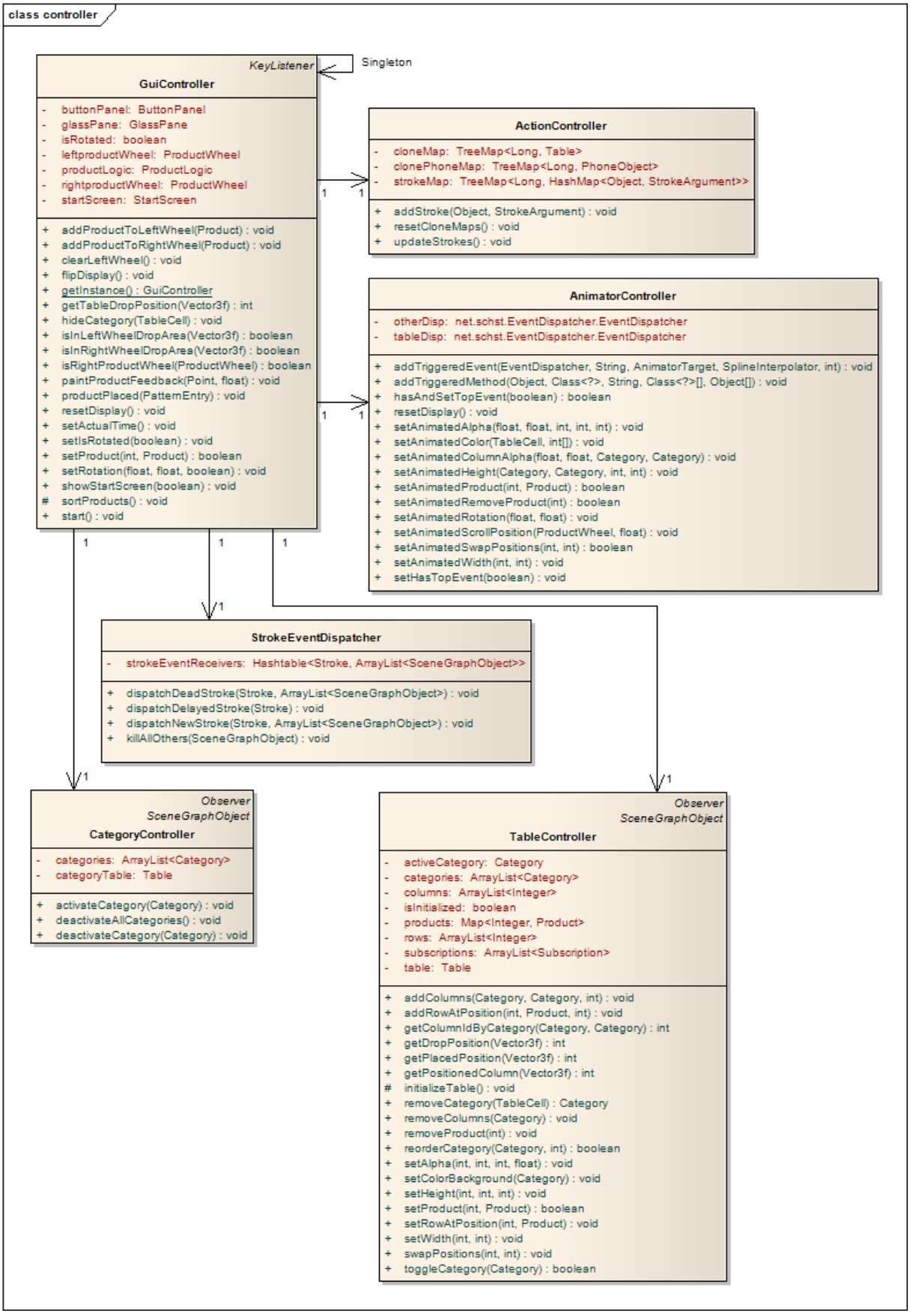


Abbildung 51: Klassendiagramm Controllerschicht

In der Controllerschicht werden die GUI-Elemente gesteuert, wodurch die Daten aus der Modelschicht für diese Elemente aufbereitet werden müssen.
Hierbei sind die folgenden Verwantwortlichkeiten zu nennen:

GuiController

Der GuiController als Singleton dient zur Ansteuerung und Anzeige der einzelnen GUI-Elemente und der verschiedenen Controller. Er delegiert die Aufgaben weiter und dient nur als Schnittstelle für die weiteren Elemente. Eine weitere Aufgabe ist die Positionierung der direkt dem GuiController untergeordneten GUI-Elemente, da er diese allesamt kennt.

StrokeEventDispatcher

Der StrokeEventDispatcher besitzt als einzige Aufgabe die Weiterleitung der einkommenden Berührungen an die richtigen Elemente.

ActionController

Der ActionController führt die auf der Oberfläche vollführten Bewegungen und Aktionen aus. Die gesamte Auswertung von Klicks, Flicks oder Bewegungen werden durch diesen Controller durchgeführt. Der ActionController muss ausserdem auch dafür sorgen, dass auf der Vergleichstabelle nur eine Aktion gleichzeitig ausgeführt wird. Würde dies nicht verhindert, könnte dies zu Inkonsistenzen in der Tabelle führen (z.B. beim gleichzeitigen Ersetzen und Entfernen eines Produktes).

AnimatorController

Sobald eine Animation ausgeführt wird, wurde dies über den AnimatorController gestartet. Dabei werden die Animationen im Zusammenspiel mit dem EventDispatcher nacheinander ausgeführt. Wie auch der ActionController führt der AnimatorController eine Überprüfung durch, damit nicht mehrere Aktionen gleichzeitig auf der Vergleichstabelle ausgeführt werden.

TableController

Bei Manipulationen an der Vergleichstabelle werden jederzeit über den TableController ausgeführt. Durch ihn werden Berechnungen, Indexzugriffe, Breiten- und Höhenveränderungen und Aktionen wie das Hinzufügen oder Entfernen von Spalten vereinfacht.

CategoryController

Der CategoryController kontrolliert das Hinzufügen und Entfernen der Kategorien beim TableController. Die beiden Controller ergänzen sich bei diesen Arbeiten jeweils gegenseitig. Nebenbei sorgt der CategoryController auch für die richtige Anzeige der entfernten Kategorien.

8.6 DARSTELLUNGSSCHICHT

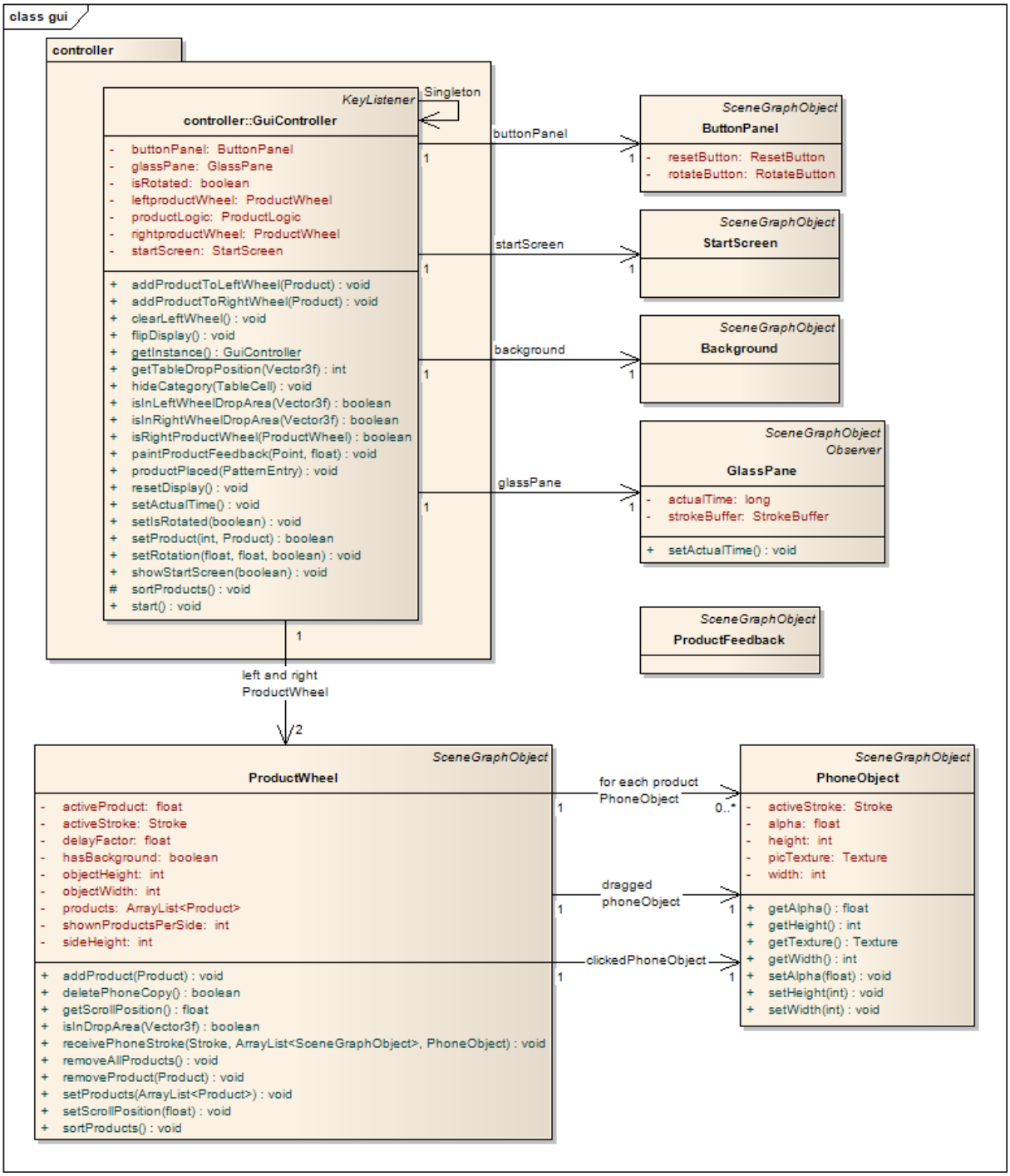


Abbildung 52: Klassendiagramm Darstellungsschicht

Die Darstellungsschicht kümmert sich um die Darstellung der Daten aus der Modelschicht. Dies wird durch die verschiedenen Controller realisiert. Wie im Diagramm ersichtlich ist, werden die einzelnen GUI-Elemente durch den GuiController erstellt und geladen. Die GUI-Elemente selbst kommunizieren nur mit den Controllern. Die Kommunikation beschränkt sich dabei auf die Mitteilung der Benutzereingaben.

8.6.1 *Table*

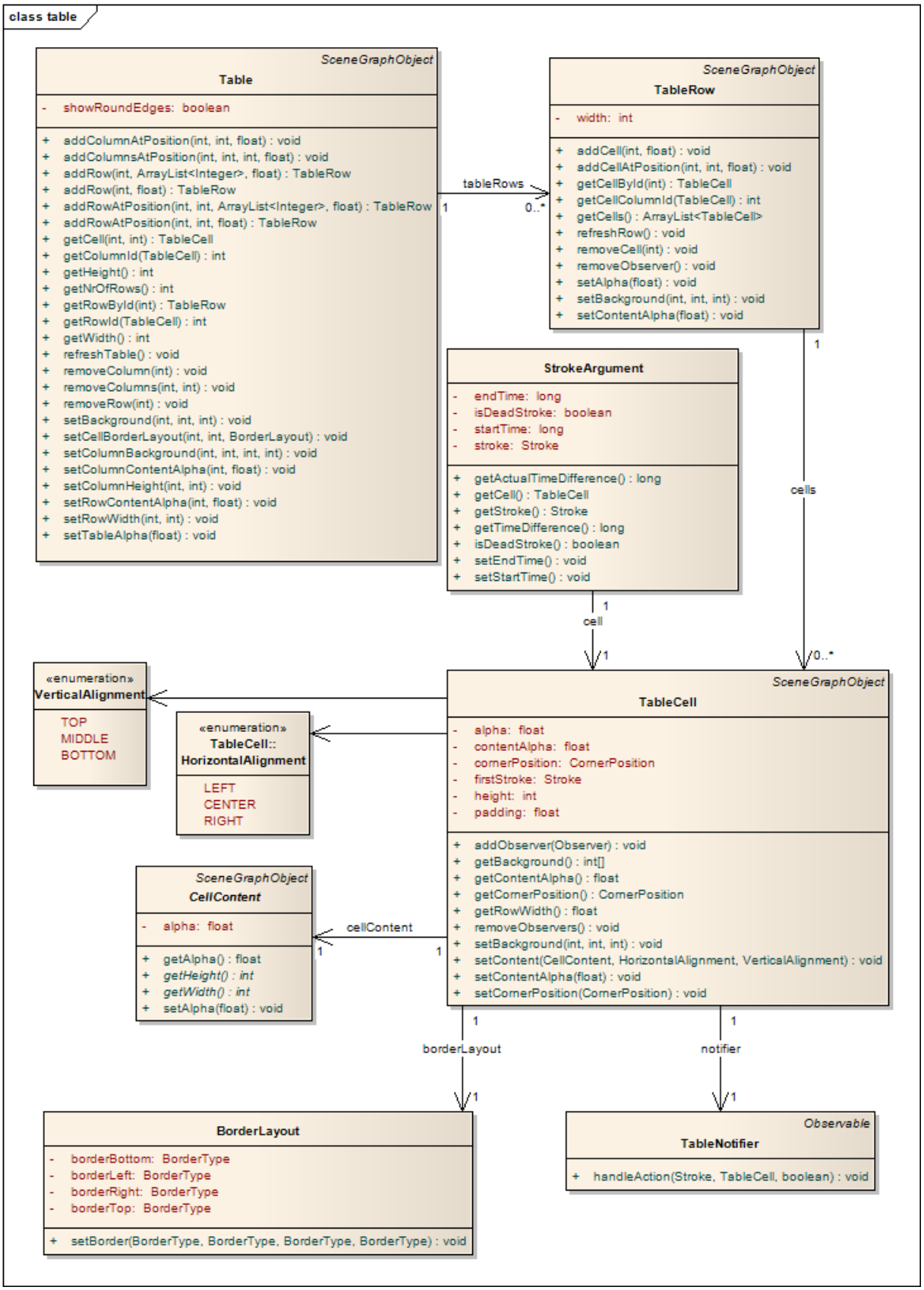


Abbildung 53: Klassendiagramm Darstellungsschicht Table

Die Table wird möglicherweise später im Framework eingesetzt

Beim Entwurf der Table wurde von Beginn weg darauf geachtet, dass diese später im Framework verwendet werden kann. Deswegen besitzen die Table-Komponenten keine spezifisch auf dieses Projekt festgelegten Eigenschaften. Diese Eigenschaften und Methoden wurden in der Klasse TableController in der Controllerschicht erstellt.

In der Table fungiert die TableCell als Rückmeldungsobjekt für Benutzereingaben. Die Klasse TableNotifier dient hierbei als Hilfsklasse, da die Einschränkung der Mehrfachvererbung die Integration der Observable-Klasse durch die TableCell (welche bereits das SceneGraphObject integriert) verhinderte. Dabei wird durch den TableNotifier das StrokeArgument an den Empfänger gesendet.

Die abstrakte Klasse CellContent wurde für die unabhängige Darstellung verschiedener Inhalte erstellt. Im Diagramm in der Abbildung 54 sieht man die eingesetzten Zellinhalte.

KLASSENNAME	BESCHREIBUNG
Table	Durch die Klasse Table wird ein Index-Zugriff auf jede Zelle der Tabelle ermöglicht.

Tabelle 28: Schnittstellen der Table der Darstellungsschicht

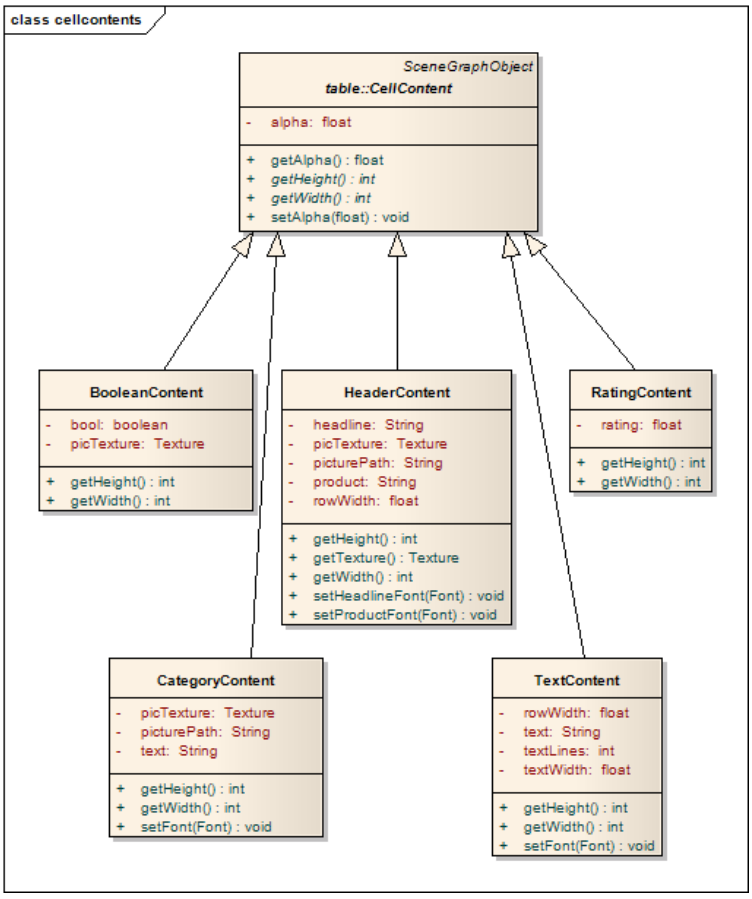


Abbildung 54: Klassendiagramm Darstellungsschicht Table Zelleninhalte

8.6.2 *Animators*

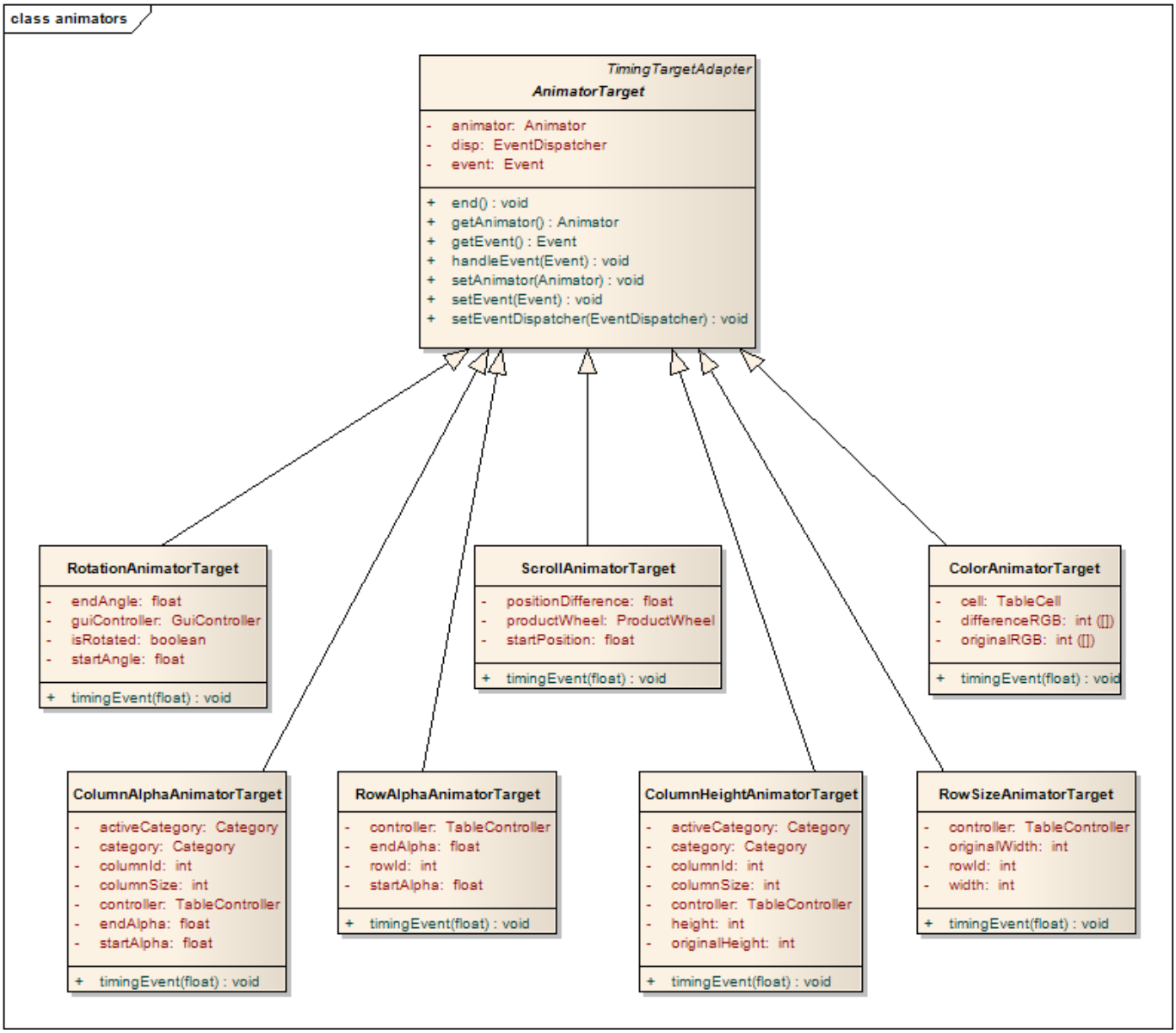


Abbildung 55: Klassendiagramm Darstellungsschicht Animators

Die Animatoren ermöglichen die einfache Erzeugung von Animationen. Die eingesetzten Animatoren wurden in dieser Applikation immer über den EventDispatcher gestartet. Deswegen musste eine Möglichkeit gefunden werden, den EventDispatcher nach Ablauf der Animation zu benachrichtigen. Dies wurde durch das Observer Pattern auf der abstrakten Klasse AnimatorTarget realisiert.

8.6.3 *Buttons*

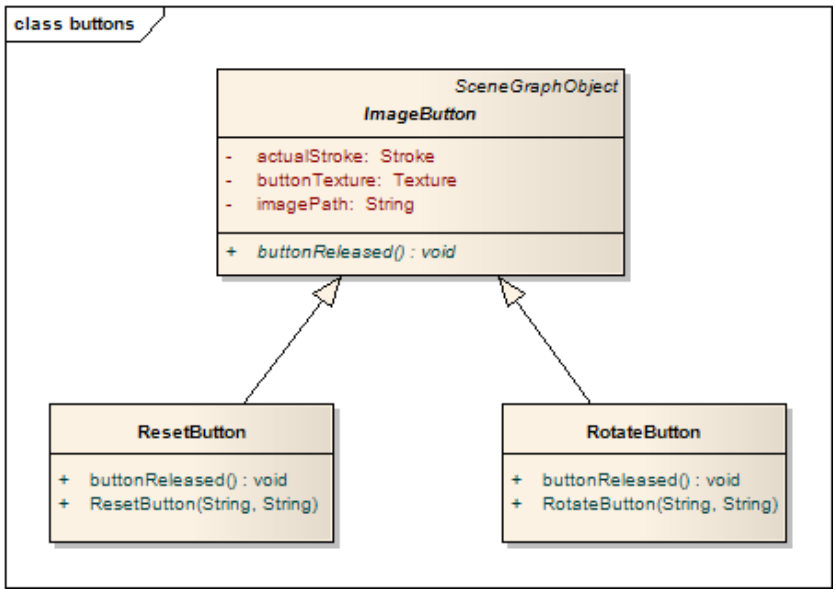


Abbildung 56: Klassendiagramm Darstellungsschicht Buttons

Bei den Buttons wurde das Strategy Pattern realisiert, wodurch bei den abgeleiteten Klassen nur die Aktion beim Klick anders implementiert werden muss.

8.7 DESIGNENTSCHEIDE

8.7.1 *Wegfall der Menüleiste*

Zu Beginn der Arbeit wurde die Benutzeroberfläche mit einer Menüleiste am rechten Rand konzipiert (zu sehen in Abbildung 16 auf Seite 49). In Zusammenarbeit mit dem Betreuer wurden die möglichen Aktionen der Menüleiste durch Gesten ersetzt.

Vorteile:

- intuitive Bedienung
- mehr Platz vorhanden für weitere GUI-Elemente
- dem Medium gerechte Bedienung

Nachteile:

- zusätzlicher Implementationsaufwand

Da die Vorteile die Nachteile überwiegen, wurde dieser Designentscheid umgesetzt.

8.7.2 *Kategoriewert vorgeben*

Ursprünglich war eine Berechnung des Kategoriewertes geplant. Dies sollte abhängig von den Attributen kalkuliert werden. Nach Absprache mit dem Betreuer wurde diese Idee verworfen. Die Kategoriewerte wurden danach statisch in der Datenbank festgelegt und nicht dynamisch anhand der Attribute berechnet.

Vorteile:

- massiv geringerer Implementationsaufwand

Nachteile:

- Kalkulation der Kategoriewerte erfolgt statisch
- hoher Verwaltungsaufwand (stetige Anpassung der Kategoriewerte bei neuen Telefonen)

Aufgrund des neu gelegten Fokus auf die Benutzeroberfläche, wurde dieser Designentscheid aus zeitlichen Gründen umgesetzt.

8.7.3 *Einführung des CellContent*

Eine Tabellenzelle soll beliebigen Inhalt darstellen können. Anfänglich unterschied die Klasse TableCell selber, welcher Inhalt gezeichnet wird. Dies überfrachtete die Klasse bei vielen verschiedenen Inhalten jedoch sehr schnell. Deshalb wurde der CellContent konzipiert. Der CellContent wird dabei in der Zelle gespeichert und eine vom CellContent abgeleitete Klasse kann individuell angepasst werden.

Vorteile:

- besserer Code
- sehr gut erweiterbar
- TableCell wird applikationsunabhängig

Nachteile:

- zusätzliches Objekt

Die Tabelle wurde von Anfang an für eine spätere Integration in das Framework konzipiert. Durch die Umsetzung mit dem CellContent können die Zellinhalte applikationsspezifisch festgelegt werden. Die Tabelle ist dadurch unabhängig von ihren Inhalten. Aufgrund dessen wurde dieser Designentscheid realisiert.

8.7.4 *Anzeige aller Produkte als Drehrad*

Bei der Anzeige eines Produkts sollten die Vergleichsprodukte innerhalb der Tabelle angezeigt werden. Da jedoch der Benutzer nicht immer diese in der Tabelle angezeigt haben will, musste eine alternative Darstellungsform gesucht werden. Bei der Suche nach einer Alternative

stellte sich die Frage, ob nicht alle Produkte sortiert angezeigt werden können. Damit der Benutzer dabei nicht mit Produkten überschwemmt wird, können die Produkte unter Verwendung eines Produktrads stufenlos ein- und ausgeblendet werden.

Vorteile:

- grafisch sehr attraktiv
- geringerer Implementationsaufwand
- keine Modifikation an der Tabelle notwendig
- eingängige Steuerung

Nachteile:

- weniger Darstellungsraum

Durch die intuitive Bedienung und das attraktive Erscheinungsbild haben wir uns entschieden, das Produktrad umzusetzen.

8.7.5 *Aktionsbehandlung im ActionController*

Für die Behandlung der verschiedenen Aktionsmöglichkeiten wurde zu Beginn eine Auswertung in der Klasse TableCell angedacht. Schnell kam die Idee eines zentralen Controllers auf, der alle diese Aktionen behandelt.

Vorteile:

- zentrale Aktionsbehandlung
- der ActionController ist nicht nur auf die Tabelle fixiert
- dadurch können auch die Aktionen anderer GUI-Elemente ausgewertet werden

Nachteile:

- mehr Methoden werden beim Aufruf durchlaufen

Die Vorteile einer zentralen Aktionsbehandlung überwiegen den Nachteil. Deswegen wurde der ActionController implementiert.

8.7.6 *Beschränkung der Anzahl paralleler Aktionen*

Während dem zwischenzeitlichen Testen wurde bemerkt, dass bei zu vielen gleichzeitigen Aktionen auf der Tabelle der EventDispatcher nicht mehr zuverlässig funktionierte. Im Gespräch mit Christian Iten kam die Idee auf, die Aktionen auf eine Aktionsreihe zu begrenzen. Das heisst, dass eine Aktion wie «Produkt entfernen» mit den Animationen durchlaufen werden kann, dabei kann jedoch keine weitere Aktion parallel ablaufen.

Vorteile:

- weniger fehleranfällig
- einfacher testbar
- keine Aneinanderreihung von sehr vielen Aktionen mehr möglich (unendliche Aktionsqueue)

Nachteile:

- Begrenzung der Eingabemöglichkeiten
- parallel ablaufende Animationen auf einem GUI-Element werden verunmöglicht

Der Vorschlag wurde für die Swisscom-Präsentation umgesetzt. Da sich die Änderung bewährte und viele Probleme behob, wurde sie beibehalten.

8.8 SEQUENZDIAGRAMME

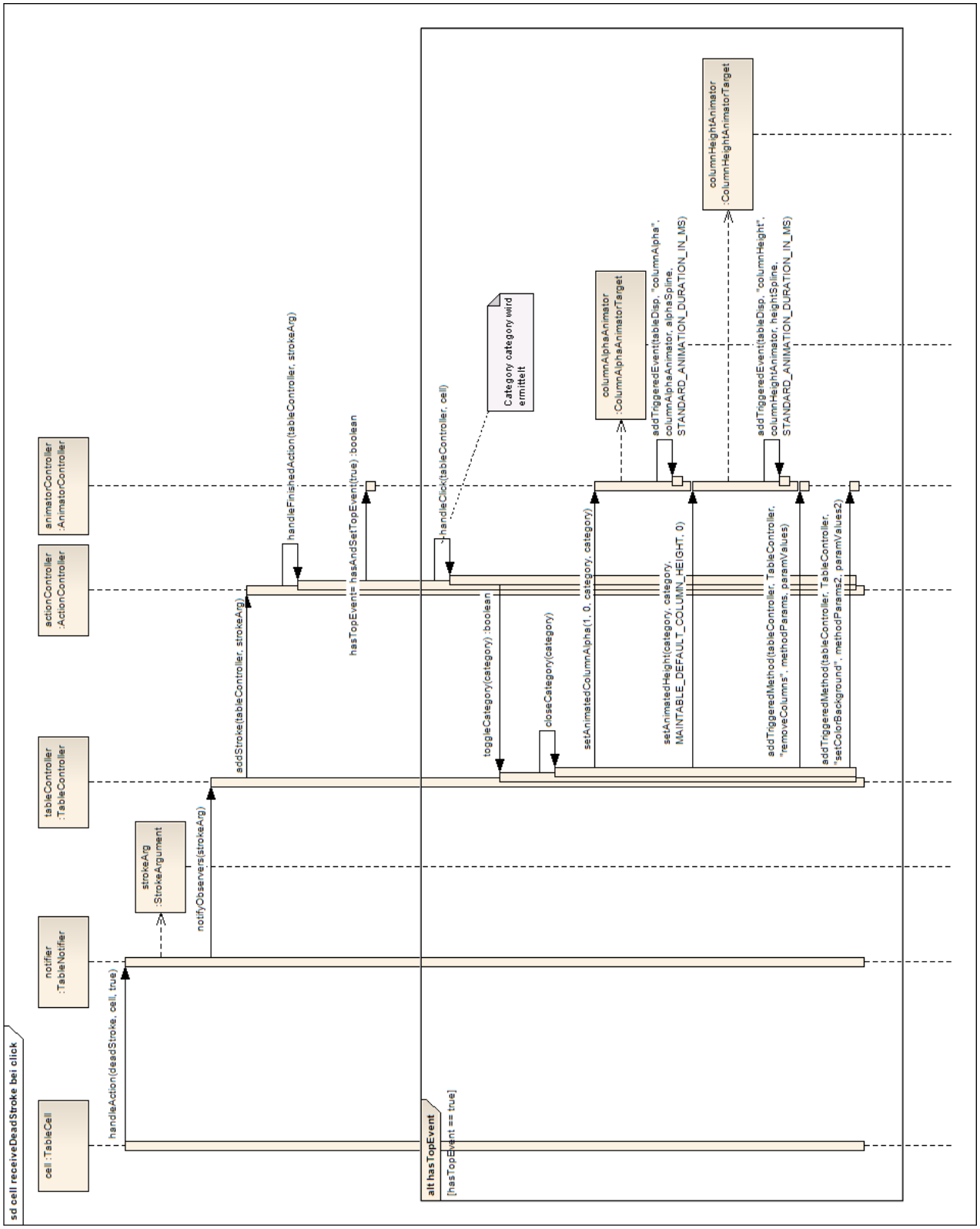


Abbildung 57: Sequenzdiagramm bei einem Klick auf eine Zelle

Die Aktion vom Sequenzdiagramm aus der Abbildung 57 wird durch ein Druck auf eine Zelle und dem anschliessenden Loslassen auf der gleichen Position ausgelöst.

Ablauf bei einem Druck auf eine Tabellenzelle

Die Zelle cell gibt dem TableNotifier notifier danach den Auftrag ein neues StrokeArgument strokeArg mit den gesetzten Parametern (den betreffenden Stroke, die aktuelle Zelle und das Flag, ob der Stroke durch Loslassen ausgelöst wurde) als Benachrichtung an den TableController tableController zu senden. Der TableController leitet dies dem ActionController weiter, welcher zuerst überprüft, ob es eine zulässige, abgeschlossene Aktion ist.
Über den AnimatorController wird überprüft, ob auf dem EventDispatcher bereits eine Aktion durchgeführt wird. Sollte keine Aktion durchgeführt werden, so wird das Flag gesetzt, damit keine andere Aktion möglich ist.

Danach führt der ActionController die auf dem Klick definierte Aktion aus. Hier wird bei dieser Aktion die aktive Kategorie auf dem TableController gewechselt. Für diesen Wechsel müssen mehrere Animationen nacheinander ausgeführt werden. Deswegen werden die Animationen auf den EventDispatcher gelegt, welcher diese nacheinander ausführt.

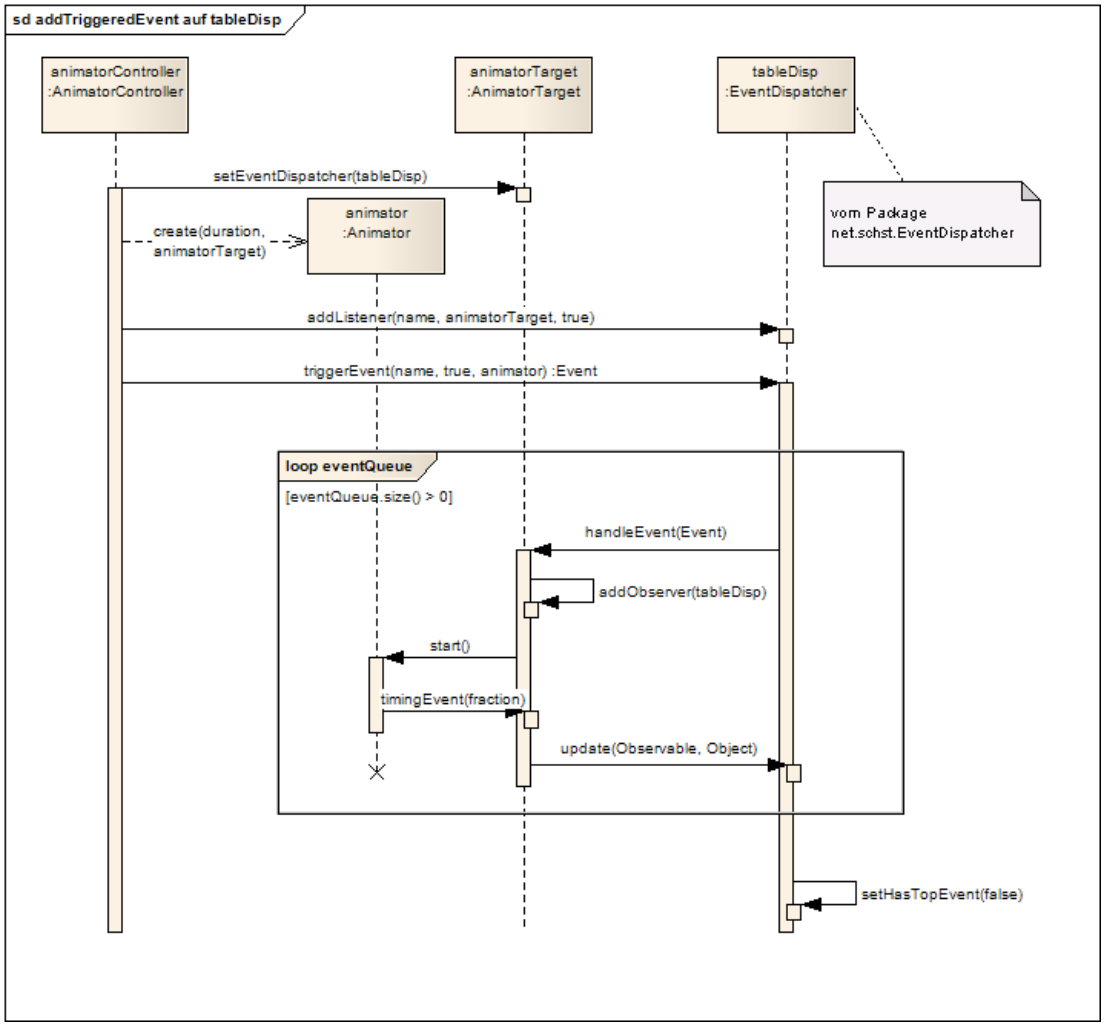


Abbildung 58: Sequenzdiagramm des Eventdispatchers

Die Aktion vom Sequenzdiagramm aus der Abbildung 58 wird beim Aufruf z.B. von setAnimatedHeight ausgeführt. Bei diesem Sequenzdiagramm wird also der anschliessende Ablauf im EventDispatcher erklärt.

Ablauf einer Animation im EventDispatcher

Zu Beginn wird auf dem AnimatorTarget (welches die Animationsaktion durchführt) der EventDispatcher gesetzt. Dadurch können unterschiedliche EventDispatcher verwendet werden, was sich bei unabhängigen GUI-Elementen als sehr nützlich erweist.
Anschliessend wird ein Animator[3] mit dem AnimatorTarget erzeugt. Auf dem EventDispatcher wird danach das AnimatorTarget als Listener definiert und das Event in die Queue des EventDispatcher eingefügt.

Der EventDispatcher führt die Events in der Queue aus, wobei auf dem AnimatorTarget über die Methode handleEvent der gesetzte EventDispatcher als Observer eingetragen wird. Danach startet das AnimatorTarget den zugewiesenen Animator. Die Animation wird durchgeführt

3 eine im Framework integrierte Klasse der JDesktop-Library

und anschliessend benachrichtigt das AnimatorTarget den EventDispatcher. Dieser kann danach das nächste Event ausführen.

Sobald der EventDispatcher keine Events mehr in der Queue hat, wird das Flag zur Verhinderung von mehreren gleichzeitigen, durch die Controller ausgelöste Aktionen zurückgesetzt.

8.9 ALGORITHMUS FÜR DIE MUSTERERKENNUNG

Um die auf dem Tisch platzierten Smartphones zu unterscheiden, wird jedes Smartphone mit einem eindeutig identifizierbaren Muster versehen. In diesem Abschnitt wird sowohl der Aufbau des Musters wie auch die Erkennung dieses Musters näher dokumentiert. Im Rahmen dieser Arbeit wurde das Muster mithilfe einer unterschiedlichen Anordnung von Legosteinen realisiert.

8.9.1 *Aufbau des Musters*

Das Muster wird durch gleichmässig angeordnete Pins dargestellt. Dabei wird ein Feld von 3 x 4 Pins verwendet. Die Abstände der Pins müssen immer gleich gross sein. Die Pins 1-3 sowie 10-12 (siehe Abbildung 59) sind reserviert, um die Ausrichtung des Smartphones bestimmen zu können. Der rot schattierte Bereich (Pins 4-9) kann für die Abbildung eines individuellen Bitmusters verwendet werden. Dabei ist ein Pin entweder gesetzt (dies entspricht intern einer «1») oder nicht gesetzt («0»). Beim Beispiel in der Abbildung 59 repräsentieren die orangen Punkte die gesetzten Pins. Dies ergibt das Bitmuster «110011».

Festgelegte Pin-Positionen

Die restlichen Pins müssen immer wie im Beispiel gesetzt sein. Sprich die Pins an den Positionen 1, 3 und 11 sind nicht gesetzt und diejenigen an den Positionen 2, 10 und 12 sind gesetzt.

Dadurch können insgesamt sechs Pins für die Abbildung des Bitmusters verwendet werden. Damit können 2^6 (64) Kombinationen gebildet werden. Das Bitmuster «000000» kann jedoch nicht verwendet werden. Somit bleiben 63 Kombinationen übrig.

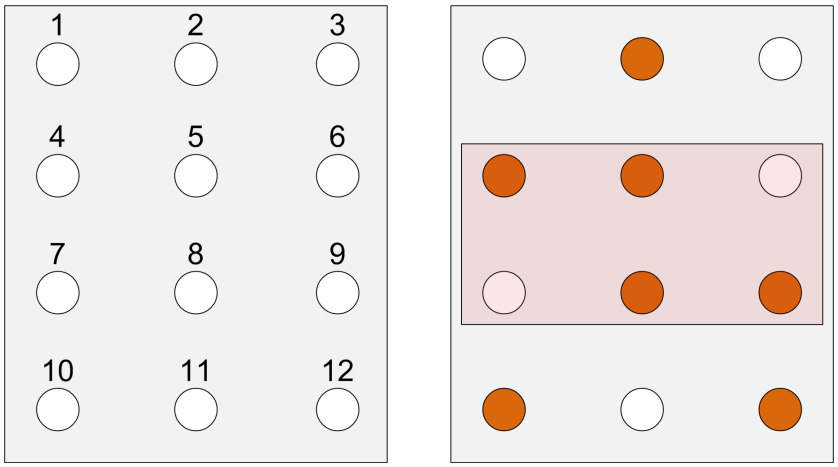


Abbildung 59: Anordnung der Pins und ein Beispiel

8.9.2 *Der Algorithmus*

Idee des Algorithmus

Die Position des Musters kann dann eindeutig bestimmt werden, wenn sowohl ein eindeutig zuweisbarer Punkt vorhanden ist, als auch der Drehwinkel bekannt ist. Als Voraussetzung muss hierbei der Abstand zwischen zwei Pins sowie die Grösse des Bitmusters bekannt sein.

Der einzige Punkt der eindeutig identifizierbar ist, ist der Punkt 2 (siehe Abbildung 60). Mithilfe der Voraussetzungen kann die Strecke zwischen den Punkten 2 und 10 sowie 2 und 12 mit dem Satz des Pythagoras

$$(c = \sqrt{a^2 + b^2})$$

berechnet werden. Da es sich bei der vorgegebenen Anordnung um ein gleichschenkliges Dreieck handelt, reicht die Berechnung nur einer Distanz aus.

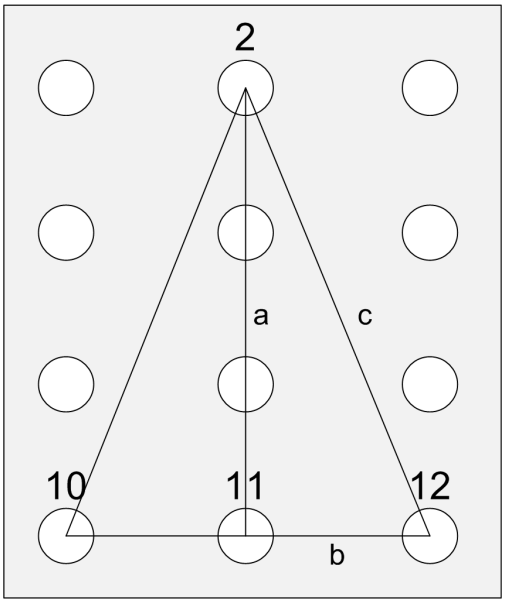


Abbildung 60: Die Strecke zwischen den Eckpunkten kann berechnet werden.

Identifikation der Eckpunkte

Aus der Menge der Eingangspunkte wird nun zwischen jedem Punkt die Strecke berechnet. Ist diese Strecke gleich gross wie die zuvor berechnete Strecke zwischen den Verbindungspunkten muss es sich um einen Eckpunkt handeln. Noch ist jedoch unklar um welchen Eckpunkt es sich genau handelt. Wird vom Punkt 2 ausgegangen werden jedoch dank des gleichschenkligen Dreiecks gleich zwei Punkte gefunden, während von den Punkten 10 und 12 logischerweise nur ein Punkt (der Punkt 2) gefunden wird. Dadurch kann der Punkt 2 eindeutig bestimmt werden.

Mit den berechneten Eckpunkten könnten prinzipiell bereits die Punkte des Bitmusters berechnet werden. Jedoch ist dies nur dann möglich, wenn das Smartphone nicht gedreht wird, sprich mit einem Winkel von 0 Grad auf dem Tisch liegt. Dies ist aber in der Realität nicht möglich, weswegen der Verschiebungswinkel bestimmt werden muss.

Bestimmung des Drehwinkels

Für die Bestimmung des Drehwinkels (siehe Abbildung 61) sind lediglich die Punkte 2 und 11 erforderlich. Dabei dient der Punkt 2 als Drehpunkt. Die rot eingefärbte Fläche repräsentiert das gedrehte Muster, während die graue Fläche eine Drehung von 0 Grad darstellt. Da der Punkt 2 bereits bekannt ist, wird für die Bestimmung des Punktes 11 lediglich der Mittelpunkt der Strecke zwischen den Punkten 10 und 12 berechnet. Die Strecken a und b sind bereits bekannt. Die Strecke c kann ebenfalls berechnet werden, da die beiden Punkte 11 bekannt sind. Sind alle Strecken bekannt, kann über den Cosinussatz

$$\gamma = \arccos(\frac{c^2-a^2-b^2}{2ab})$$

der Winkel γ berechnet werden.

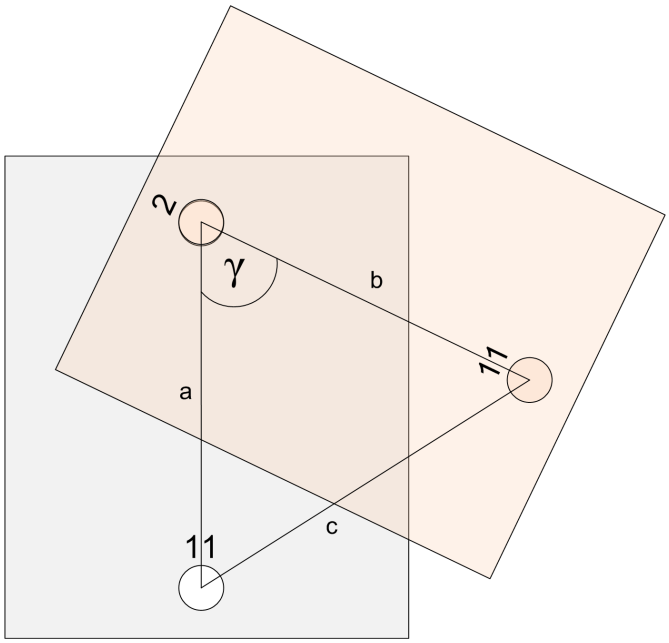


Abbildung 61: Der Winkel kann mit dem Cosinussatz bestimmt werden.

Berechnung der Pinpositionen

Konnte der Drehwinkel erfolgreich bestimmt werden, kann die Pinbelegung ausgehend vom Punkt 2 (siehe Abbildung 59) bestimmt werden. Entspricht der Winkel 0 Grad, kann eine Verschiebung auf der X-Achse wie auch auf der Y-Achse relativ einfach bestimmt werden. Die Koordinaten des Punktes 4 könnten dann mit der einfachen Rechnung

$$x_4 = x_2 - abstand$$

sowie

$$y_4 = y_2 + abstand$$

berechnet werden. Hierbei entspricht der Abstand der Strecke zwischen zwei Punkten.

Bei einer Drehung muss natürlich die Drehung der Abstände ebenfalls beachtet werden (siehe Abbildung 62). Bei einer vertikalen Verschiebung können die neuen Koordinaten wie folgt berechnet werden:

$$x_2 = x * \sin(\gamma)$$
$$y_2 = x * \cos(\gamma)$$

Für eine horizontale Verschiebung müssen die x- und y-Werte vertauscht werden.

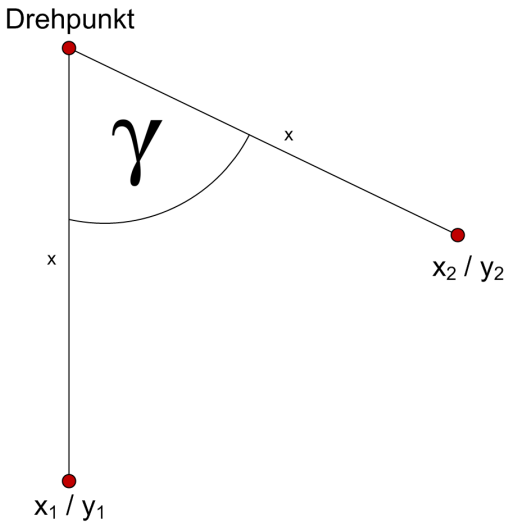


Abbildung 62: Die X und Y-Koordinaten müssen angepasst werden.

Abgleich der Eingabepunkten mit den berechneten Punkten

Anhand der berechneten vertikalen und horizontalen Verschiebungen, können nun auch die Punkte bei einem gedrehten Objekt bestimmt werden. Nun müssen nur noch die berechneten Punkte mit denen vom Multi-Touch-Tisch erkannten Punkten abgeglichen werden. Dabei werden sämtliche berechneten Punkte mit den Eingabepunkten verglichen. Stimmen die Koordinaten der Punkte überein, ist ein Pin an dieser Position gesetzt.

8.9.3 *Probleme des Algorithmus*

Problem ohne Toleranz

Der Algorithmus erkennt die Objekte nur, wenn die Koordinaten exakt übereinstimmen. Da die Eingaben des Multi-Touch-Tisches jedoch zu ungenau sind, sodass immer der exakt gleiche Abstand zwischen den Punkten vorhanden ist, werden die Pins nicht erkannt.

Problemlösung

Um dieses Problem zu lösen, wurde ein Toleranzbereich um einen berechneten Punkt festgelegt. Befindet sich der Eingabepunkt innerhalb dieses Toleranzbereichs, wird der Punkt dennoch erkannt. Die Abbildung 63 zeigt den Toleranzbereich rot schattiert. Der Punkt links unten würde nicht erkannt werden, da er ausserhalb des Toleranzbereichs liegt.

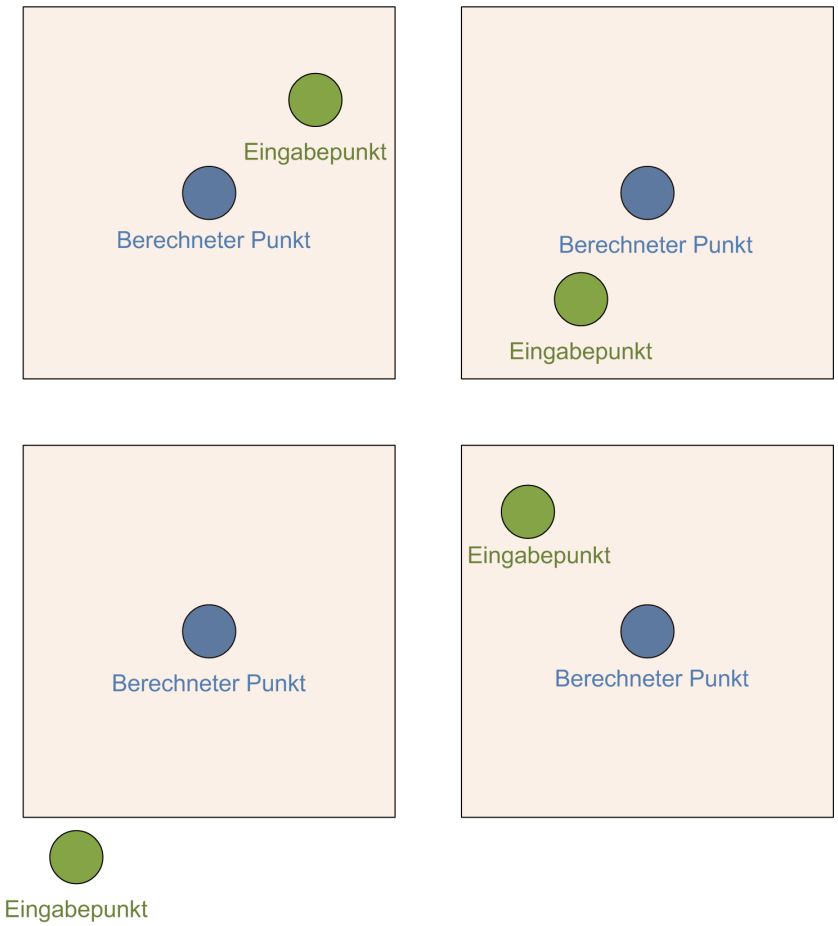


Abbildung 63: Visualisierung des Toleranzbereichs

Problem bei Drehpunkterkennung

Ein weiteres Problem entdeckten wir bei der Erkennung der Eckpunkte. Wenn das Objekt schräg positioniert wurde und ein Muster wie in Abbildung 64 vorlag, konnte es vorkommen, dass ein weiterer Eckpunkt erkannt und dieser auch noch als Drehpunkt definiert wurde. Dies sieht man in der Grafik durch die zwei eingefärbten Dreiecke, wobei das grüne Dreieck die korrekten Eckpunkte beinhaltet.

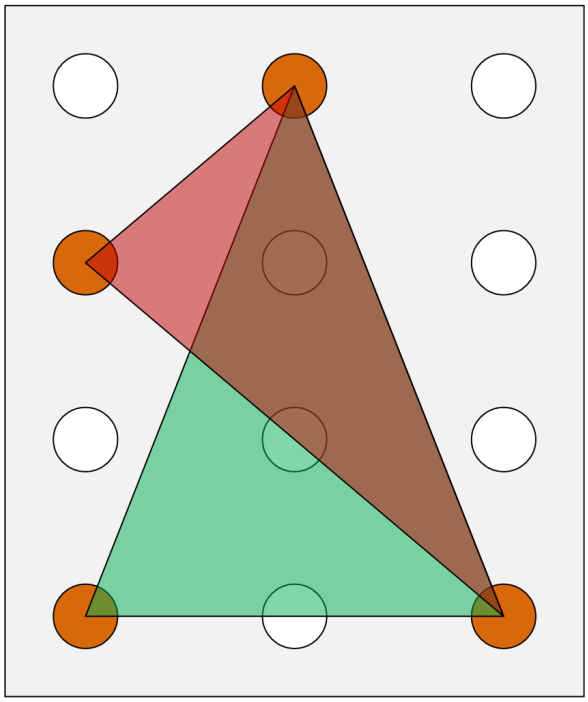


Abbildung 64: Visualisierung des Problems bei der Drehpunkterkennung

Problemlösung

Für die Lösung dieses Problems wird die Abweichung der korrekten Distanz ohne Toleranzbereich mit den zwei Punkten zwischengespeichert. Damit diese Abweichungen der Grösse nach angeordnet werden können, wird dazu der Absolutwert verwendet.

$$abweichung = |berechneteDistanz - korrekteDistanz|$$

Die berechneten Abweichungen werden danach der Grösse nach aufwärts sortiert. Bei der Iteration durch diese sortierten Abweichungen wird der erste Punkt, der zweimal in einer Abweichung vorkommt, als Drehpunkt bestimmt. Da dieser die kleinsten Abweichungen zur korrekten Distanz hat, ist es diesmal mit sehr hoher Wahrscheinlichkeit der korrekte Drehpunkt. Bei einer schlechten Kalibration des Multi-Touch-Tisches oder schlechter Verarbeitung der Pins können jedoch zu grosse Abweichungen entstehen, wodurch eine Erkennung des Drehpunkts verunmöglicht wird.

Zum Schluss werden vom Drehpunkt nochmals die Distanzen zu den weiteren möglichen Eckpunkten berechnet und zusätzlich mit der korrekten Distanz inklusive des Toleranzbereichs verglichen. Dadurch werden nur noch zwei weitere Eckpunkte gefunden.

9 REALISIERUNG

9.1 DATENSCHICHT

Die Datenschicht wurde über Hibernate realisiert. Dies ermöglicht die Unabhängigkeit vom Datenbankhersteller. Da die Datenschicht über ein Eclipse-Plugin erzeugt werden konnte, wird dies hier nicht weiter beschrieben. Es sollen nur Ausschnitte aus den Mapping-Dateien gezeigt werden. Ausserdem soll die Zugriffmöglichkeit über die selbst erstellte Klasse DatabaseFinder näher erläutert werden.

9.1.1 *DatabaseFinder*

Der DatabaseFinder besitzt eine SessionFactory, mit der er die Verbindung zur Datenbank herstellt. Zudem wurden drei verschiedene Suchmethoden erstellt, welche die von uns benötigten Datenbankzugriffe abdeckten. Innerhalb der Suchmethoden musste jeweils eine Transaktion gestartet werden, da die Suche ansonsten in einer Exception resultierte.

```
1  public ProductItemDAO findById(java.lang.Integer id) {
2    try {
3      Transaction tx = sessionFactory.getCurrentSession().beginTransaction();
4      ProductItemDAO instance = (ProductItemDAO) sessionFactory.
           getCurrentSession().get(ProductItemDAO.class, id);
5      tx.commit();
6      return instance;
7    } catch (RuntimeException re) {
8      throw re;
9    }
10 }
```

Suchmethode findById des DatabaseFinder

9.1.2 *Mappings für Hibernate*

Bei den Mappings wird für jede Klasse eine XML-Datei definiert, welche die Tabelle, Verbindungen und Attribute festlegt. Zusätzlich dazu wird im Source-Root-Verzeichnis eine XML-Datei mit dem Namen «hibernate.cfg.xml» benötigt, welche die Verbindungskonfigurationen enthält und die weiteren XML-Dateien einbindet.

```
...
<hibernate-configuration>
  <session-factory>
    <!-- Database connection settings -->
    <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/multitouch</property>
    <property name="connection.username">...</property>
    <property name="connection.password">...</property>
    ...
    <!-- SQL dialect -->
    <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
    ...
    <!-- Drop and re-create the database schema on startup -->
    <mapping resource="ch/hsr/salestable/database/mappings/SubscriptionDAO.hbm.xml" />
    <mapping resource="ch/hsr/salestable/database/mappings/SubscriptionPriceDAO.hbm.xml" />
    ...
    <mapping resource="ch/hsr/salestable/database/mappings/ProductTypeDAO.hbm.xml" />
  </session-factory>
</hibernate-configuration>
...
```

Ausschnitt aus hibernate.cfg.xml

```
...
<hibernate-mapping>
    <class name="ch.hsr.salestable.database.CategoryNameDAO" table="tbl_category_name" catalog="multitouch">
        <id name="id" type="java.lang.Integer">
            <column name="id" />
            <generator class="identity" />
        </id>
        <many-to-one name="productType" class="ch.hsr.salestable.database.ProductTypeDAO" fetch="select" lazy="false">
            <column name="fk_product_type" />
        </many-to-one>
        <property name="name" type="string">
            <column name="name" length="50" not-null="true" />
        </property>
        ...
        <set name="categoryValues" inverse="true" lazy="false">
            <key>
                <column name="fk_category_name" />
            </key>
            <one-to-many class="ch.hsr.salestable.database.CategoryValueDAO" />
        </set>
        ...
    </class>
</hibernate-mapping>
...
```

Ausschnitt aus CategoryNameDAO.hbm.xml als Beispiel

9.2 MODELSCHICHT

Die Modelschicht wurde im Package core realisiert. Die Zusammenhänge der Klassen in diesem Package wurden im Kapitel 7 bereits ausführlich erklärt. Die Komplexität dieser Klassen hält sich in Grenzen, weshalb nicht weiter darauf eingegangen werden soll.

Zu diesen Klassen wurden zusätzlich Vergleichsklassen erstellt (im Subpackage «comparator»), welche das Sortieren von Listen und Maps vereinfachte. Auch der Erkennungsalgorithmus findet sich im Subpackage «algorithm» wieder.

9.2.1 *Comparator*

Die Comparator ermöglichten uns eine einfache Sortierung der Objekte in einer Liste oder Map. Dadurch konnte an einigen Orten Code gespart und die Java-internen Möglichkeiten genutzt werden.

```
1  public class CategoryScoreComparator implements Comparator<Category> {
2    public int compare(Category c1, Category c2) {
3      if (c1.getScore() < c2.getScore()) {
4        return 1;
5      } else if (c1.getScore() == c2.getScore()) {
6        return 0;
7      } else {
8        return -1;
9      }
10   }
11 }
```

Comparator anhand des Score einer Category

9.2.2 *Erkennungsalgorithmus*

Der Erkennungsalgorithmus kann in zwei Aufgaben unterteilt werden. Der erste Teil kümmert sich um die Entgegennahme der einzelnen Eingaben, während beim zweiten Teil die Erkennung realisiert wird.

Entgegennahme der Eingaben

Die Entgegennahme der Eingaben wird durch die Klasse StrokeBuffer erledigt. Dabei läuft bei der ersten Erkennung einer Eingabe (welche der Grösse eines Pins entspricht) ein Timer ab.

```
1  public StrokeBuffer() {
2    timer = new Timer(Constants.BUFFER_TIME_IN_MILLIS, new ActionListener() {
3      @Override
4      public void actionPerformed(ActionEvent e) {
5        timerStopped();
6        timer.stop();
7      }
8    });
9  }
```

Konstruktor der Klasse StrokeBuffer

Auswertung der Eingaben

Sobald der Timer abläuft , werden die erhaltenen Eingaben durch die Methode «timerStopped» bearbeitet. Dabei werden die Eingaben nach einer Umwandlung in Punkte an die Erkennungsklasse PatternRecognizer übergeben. Bei einer erfolgreichen Erkennung wird das Erkennungsmuster, ansonsten die verfolgten Eingaben an den Observer übermittelt.

```
1  public void timerStopped() {
2    if (trackedObjects.size() > Constants.MIN_PATTERN_STROKES) {
3      List<Point> pointList = createPointList();
4      PatternEntry patternEntry = PatternRecognizer.recognizeObject(pointList)
           ;
5      if (patternEntry != null) {
6        setChanged();
7        notifyObservers(patternEntry);
8      } else {
9        setChanged();
10       notifyObservers(trackedObjects);
11     }
12   } else {
13     setChanged();
14     notifyObservers(trackedObjects);
15   }
16 }
```

Bearbeitung der Eingaben bei Ablauf des Timers

9.3 EVALUATION DER PINANORDNUNG

Bevor die genaue Pinanordnung bestimmt werden konnte, mussten Tests mit verschiedenen Materialien (siehe Abbildung 65) durchgeführt werden.

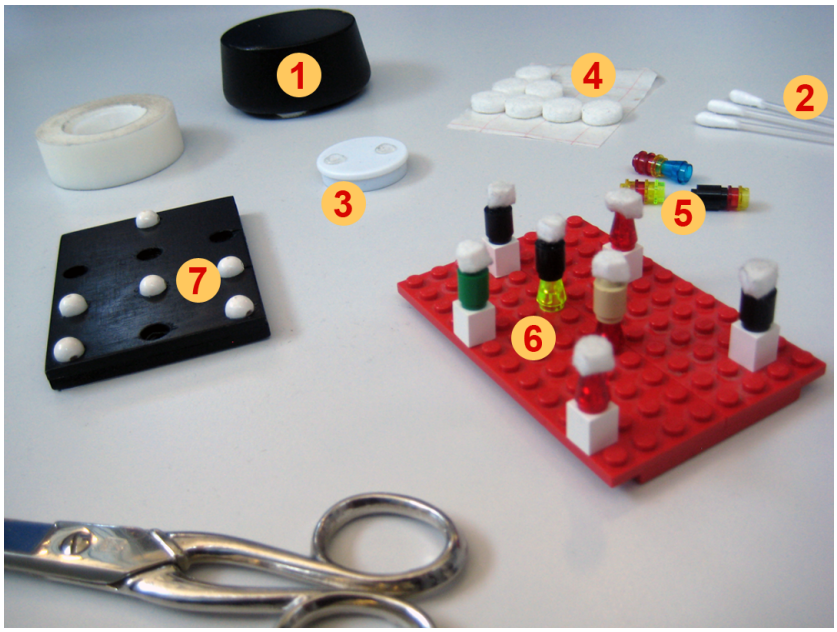


Abbildung 65: Materialienübersicht

Phycon

Der Phycon (1) wurde bereits in einer vorangegangenen Diplomarbeit als Steuerungselement eingesetzt. Für die Erkennung der Pins ist dabei eine Kombination aus weissen Füssen und einer schwarzen Oberfläche ideal. Im Falle des Phycons sind die Füsse jedoch deutlich zu gross um ein Muster bestehend aus 12 Pins auf einer kompakten Fläche zu positionieren.

Wattestäbchen

Mit der Hilfe von Wattestäbchen (2) konnte die minimale Grösse eines noch erkennbaren Objektes ermittelt werden. Der Kopf des Wattestäbchens wurde dabei nur teilweise erkennt, es musste demzufolge eine grössere Auflagefläche gefunden werden.

transparente Möbeluntersetzer

Durch den Einsatz von transparenten Möbeluntersetzern (3) konnte die Erkennung transparenter Objekte getestet werden. Dies funktionierte nicht sonderlich gut, weswegen transparente Objekte nicht weiter in Frage kamen.

weisse Möbeluntersetzer

Mit den nicht transparenten Möbeluntersetzern (4) konnten verschieden grosse Auflageflächen zurechtgeschnitten werden. Anhand dieser unterschiedlich grossen Objekte konnte die benötigte Objektgrösse relativ exakt bestimmt werden.

Legoteile

Der Abstand zwischen zwei Pins konnte durch den Einsatz von Legoteilen (5 und 6) ermittelt werden. Eine Kombination der nicht transparenten Möbeluntersetzer und den Legoelementen stellte sich dabei als ideal heraus. Mit der Hilfe der Tracker-Software konnten die erkannten Objekte schnell und einfach angezeigt werden (siehe Abbildung 66).

Nach einigen Tests konnte so der ideale Abstand ermittelt werden, er beträgt 2.4cm.

Schlussfolgerung

Mit den aus den verschiedenen Tests gewonnenen Erkenntnissen stellte uns Christian Iten einige Holzplatten mit exakt gleich grossen Pins zur Verfügung (7). Diese Holzplatten entsprechen den schlussendlich in dieser Arbeit eingesetzten Objekten (siehe Abbildung 67).

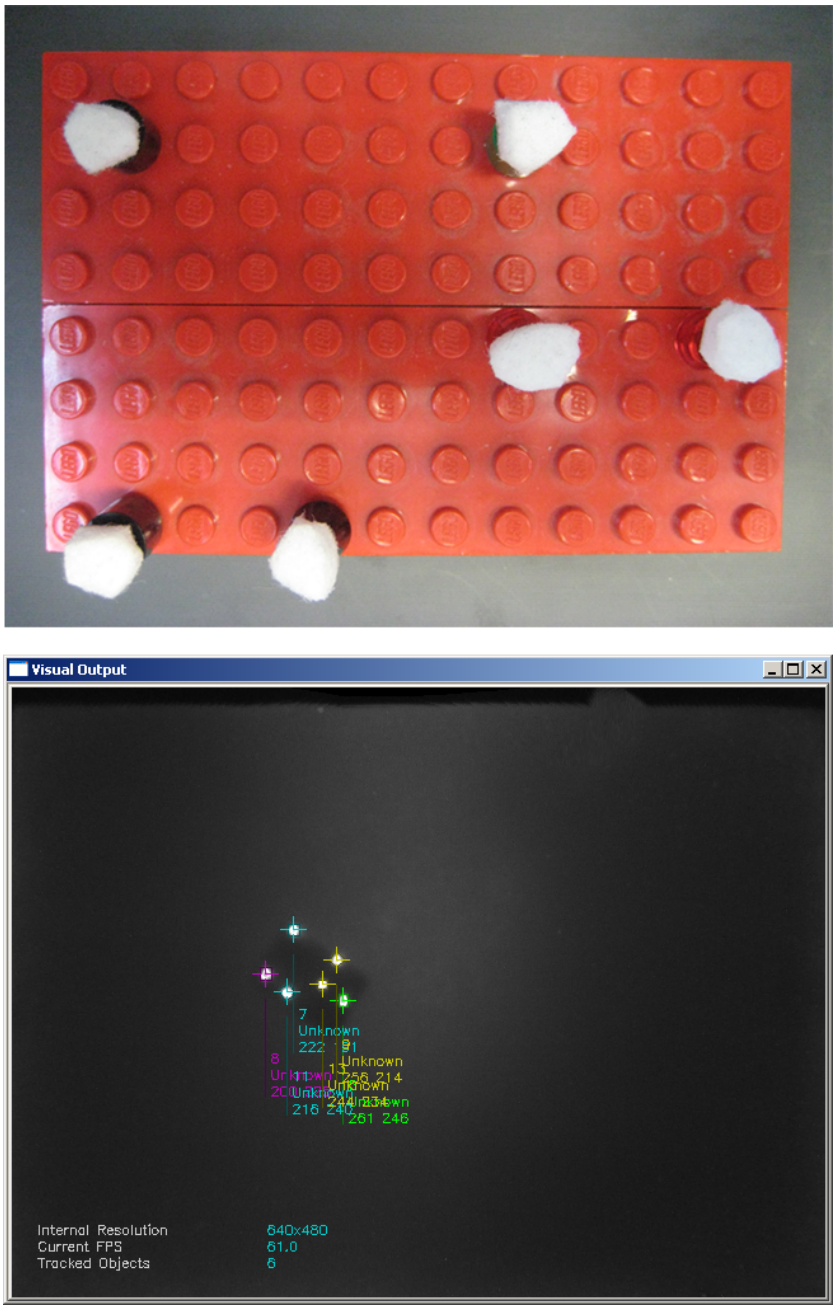


Abbildung 66: Muster mit Legoelementen und die dazugehörige Darstellung in der Tracker-Software

Abbildung 67: Die schlussendlich eingesetzten Holzplatten

9.4 CONTROLLERSCHICHT

Die Controller wurden in ihre bestimmten Aufgabenbereiche getrennt. Gewisse Controller verwenden das SceneGraphObject für die vereinfachte Positionierung der Grafikelemente. Dies hätte ansonsten an mehreren Orten nachgeführt werden müssen, was in einem erheblichen Mehraufwand resultiert hätte.

9.4.1 *GuiController*

Der GuiController delegiert die Aktionen jeweils an die bestimmten Objekte weiter. Da der GuiController ein Singleton ist und die Referenzen auf die weiteren Objekte besitzt, eignet er sich bestens für die Entgegennahme von Aufgaben und fungiert als Facade. Der GuiController kümmert sich ausserdem um die Positionierung der grafischen Elemente. Da der GuiController dadurch die Positionen der Elemente kennt, kümmert er sich auch um die Drehung der Oberfläche um 180 Grad.

Während der Drehung des Grafikelementes wird zugleich die Position verändert, wodurch sich eine Spiegelung anhand der Diagonale ergibt (siehe Abbildung 68).

Abbildung 68: Drehung der Bildoberfläche

9.4.2 *StrokeEventDispatcher*

Der StrokeEventDispatcher leitet die empfangenen Eingaben an das jeweils oberste grafische Element an dieser Position weiter. Dabei muss überprüft werden, ob die Eingabe über einen Finger oder ein Pin eines Produktes geschah. Die Unterscheidung basiert hierbei auf der Fläche der Eingabe. Dies verhindert, dass bei der Weiterleitung jede Eingabe über den Timer zur Produkterkennung verzögert wird.

9.4.3 *ActionController*

Funktionsweise der Drag-Aktion

Beim ActionController werden die Eingaben ausgewertet, worauf eine Aktion ausgelöst wird. Dabei gibt es Flick-, Click- oder Drag-Aktionen. Die Drag-Aktionen werden von Flick- oder Click-Aktionen durch ein Zeitfenster unterschieden. Sobald dieses Zeitfenster überschritten wurde, wird die Aktion als Drag-Aktion ausgewertet.

Bei der Drag-Aktion wird zu Beginn von dem angegebenen grafischen Element eine Kopie erstellt, welche jeweils auf die aktuelle Position der Eingabe (z.B. der Finger) verschoben wird. Nach dem Ende der Drag-Aktion wird die Kopie wieder entfernt.

Je nachdem auf welchem grafischen Element die Aktion ausgeführt wurde, wird dann bei der Beendigung der Eingabe die zuständige Animation ausgeführt, welche durch den AnimatorController übernommen wird.

Abbildung 69: Kopiertes Element bei der Drag-Aktion

9.4.4 *AnimatorController*

Der AnimatorController sorgt für die Erzeugung der Animationselemente und die Eintragung dieser in den EventDispatcher. Der Aufruf der verschiedenen Animationen erfolgt von ausserhalb, wodurch diese Animationen auch verschiedentlich kombiniert werden. Da bei gewissen Aktionen keine zusätzlichen Animationen zugelassen sind, kann dies von den aufrufenden Objekten über die Methode «hasAndSetTopEvent» gleichzeitig abgefragt und für andere Objekte verhindert werden. Das Abfragen und gleichzeitige Setzen sorgt für eine atomare Aktion, wodurch die Verhinderung weiterer Aktionen auch mit mehreren Threads funktioniert.

```
1 public synchronized boolean hasAndSetTopEvent(boolean hasTopEvent) {
2   if (tableDisp.hasTopEvent()) {
3     return false;
4   } else {
5     tableDisp.setHasTopEvent(hasTopEvent);
6     return true;
7   }
8 }
```

Methode zur Verhinderung von weiteren Aktionen

9.4.5 *TableController*

Im TableController werden sämtliche Änderungen auf der Vergleichstabelle ausgeführt. Die Anpassung der Tabelle führt dabei sehr schnell zu viel Code, da die Tabelle selbst für die Integration ins Framework entwickelt wurde. Deswegen enthält die Tabelle keine dynamischen Funktionen. Um den Information Expert zu wahren, wurde die Funktionalität der Klasse TableController nicht in die anderen Controller ausgelagert.

Anhand des folgenden Beispiels wird ersichtlich, dass kleine Änderungen bereits viele Codezeilen produzieren.

```
1 ...
2 TableCell dragCell = dragTable.getCell(rowId, 0);
3 dragCell.setBackground(Constants.MAINTABLE_DRAG_CATEGORY_CELL_RGB[0],
      Constants.MAINTABLE_DRAG_CATEGORY_CELL_RGB[1], Constants.
      MAINTABLE_DRAG_CATEGORY_CELL_RGB[2]);
4 Category category = getCategoryByColumnId(columnId, true);
5 ...
6 Category categoryValue = products.get(rowId).getCategoryNameId(category.
      getCategoryNameId());
7 RatingContent cellContent = new RatingContent(dragCell.getName() + "_rating_
      " + category.getName(), (float) categoryValue .getScore() / Constants.
      MAX_SCORE_VALUE);
8 setCellAttributes(dragCell, cellContent, dragCell.getContentAlpha(),
      HorizontalAlignment.CENTER, VerticalAlignment.MIDDLE, true);
9 ...
```

Eine Anpassung des Zellinhaltes benötigt viel Code

Zusätzlich zu den Änderungsmöglichkeiten bietet der TableController auch Möglichkeiten um zum Beispiel von einer Position auf eine Zelle zu schliessen oder mit der Zeilennummer die Kategorie zu bestimmen.

In der Vergleichstabelle benutzen wir eine zusätzliche Zeile zur Füllung des Leerraums, damit die Tabelle immer gleich hoch ist.

9.4.6 *CategoryController*

Durch den CategoryController werden die in der Vergleichstabelle entfernten Kategorien verwaltet. Die Kategorien werden dabei durch eine kleine Tabelle angezeigt. Über den Transparenzwert einer Kategorie wird dabei entschieden, ob sie in die Vergleichstabelle gezogen werden kann oder nicht.

9.5 DARSTELLUNGSSCHICHT

Die Darstellungselemente wurden jeweils durch die Ableitung der Klasse SceneGraphObject aus dem Framework realisiert. Die wichtigsten Elemente werden nachfolgend beschrieben.

9.5.1 *GlassPane*

Dadurch, dass der ActionController kein SceneGraphObject ist, kann er seine Eingaben nicht ständig erneuern. Damit er trotzdem die Drag-Elemente den Eingabepositionen anpassen kann, wird die zuständige Methode vom GlassPane aufgerufen. Die Klasse GlassPane kümmert sich ausserdem um das Laden der Texturen über den TextureLoader, welcher in einem späteren Abschnitt beschrieben wird.

Jede Eingabe passiert diese Klasse. Dadurch wurde ihr die Aufgabe zugeteilt, die Benutzeroberfläche nach einer gewissen Zeitspanne ohne Eingabe zurückzusetzen. Sie setzt diese jedoch nicht direkt zurück, sondern ruft den GuiController hierfür auf.

```
1 public void update() {
2   GuiController.getInstance().getActionController().updateStrokes();
3   if (!TextureLoader.isInitialized()) {
4     TextureLoader.loadDirectory("img");
5   } else if (System.currentTimeMillis() - actualTime > Constants.
        RESET_TIME_IN_MILLIS) {
6     actualTime = System.currentTimeMillis();
7     GuiController.getInstance().resetDisplay();
8   }
9 }
```

Update-Methode des GlassPane

9.5.2 *ProductWheel - das Produktrad*

Berechnung der Skalierung

Die Klasse ProductWheel sorgt für die Anzeige der Produkte in einer einem Rad ähnelnden Darstellung. Dieser Effekt wird durch einen linearen Abstieg der Anzeigegrösse und des Transparenzwertes der Produkte erreicht. Zusätzlich wird bei den Produkten eine absteigende Z-Position (für die Übereinanderlegung der Produkte) gesetzt. Diese verschiedenen Werte werden anhand einer Skala berechnet, wie in der Abbildung 86 ersichtlich ist. Durch die Vergrösserung der Skala von -2 bis 2 wird eine grössere Abhebung der Werte zwischen den Produkten untereinander erreicht.

Die Position des Rades wird mit einer Fliesskomazahl bestimmt

Die verwendete Skala ist stufenlos, da dazu eine Fliesskommazahl verwendet wird. Vom Mittelpunkt ausgehend werden jeweils die Werte der oberen und unteren Produkte berechnet. Dabei wird zuerst der Skalierungsfaktor des mittleren Produktes berechnet. Dieser wird anschliessend für die weiteren Produkte als Ausgangspunkt verwendet und auf das nächste Produkt umgerechnet. Da die Produkte immer den selben Abstand haben, können jeweils die Werte zweier Produkte berechnet werden.

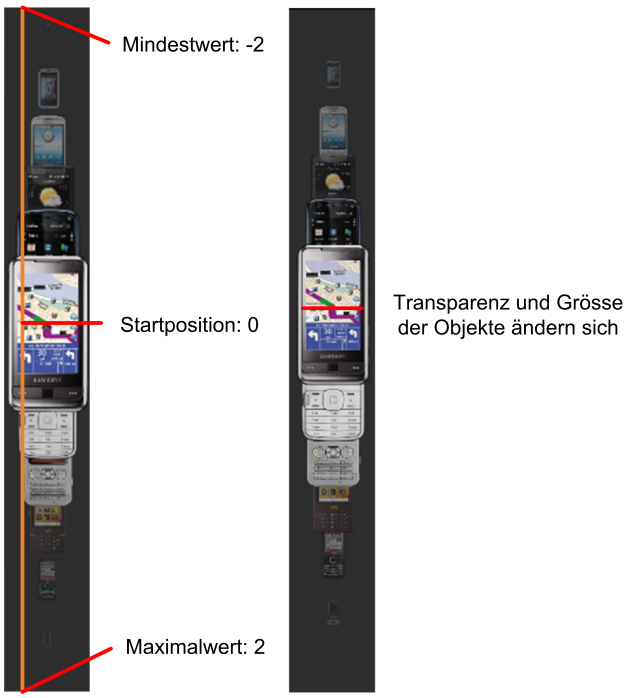


Abbildung 70: Skala des Produktrades

9.5.3 *Animationen*

Die abstrakte Klasse AnimatorTarget enthält bereits den Start der Animation, sowie auch die Benachrichtigung nach Ablauf der Animation.

Die Animationen werden über den EventDispatcher aufgerufen, welcher dies mit dem Gebrauch von Events erledigt. Dabei wird auf dem betreffenden Objekt die Methode «handleEvent» ausgelöst. Beim AnimatorTarget wird hierbei der EventDispatcher als Observer eingetragen und danach die Animation gestartet.

```
1 public void handleEvent(Event e) {
2   event = e;
3   if (disp != null) {
4     addObserver(disp);
5   }
6   animator = (Animator) e.getContext();
7   animator.start();
8 }
```

Start der Animation

Nach dem Ende der Animation wird die Methode «end» aufgerufen, wodurch der Observer benachrichtigt wird.

Die Fraction beim «timingEvent» muss gerundet werden

Die abgeleiteten Klassen müssen für eine bestimmte Animation nur noch die Methode «timingEvent» überschreiben und darin die gewünschte Änderung auf einem Objekt definieren. Das «timingEvent» bekommt dabei immer einen Wert zwischen 0 und 1. Dabei muss beachtet werden, dass bei der letzten Durchführung der Wert 1 nicht erreicht wird (erreichter Wert beträgt ca. 0.99999). Dieses Problem muss über eine Rundung des Werts gelöst werden, wie auch im nachfolgenden Beispiel ersichtlich ist.

```
1  public void timingEvent(float fraction) {
2    if (cell != null) {
3      //Wert muss gerundet werden, da sonst 1 nicht erreicht wird
4      if ((Math.round(fraction * Constants.MATH_ROUND_MULTIPLIER) / Constants.
           MATH_ROUND_MULTIPLIER) < 1) {
5        int r = originalRGB[0] + Math.round(differenceRGB[0] * fraction);
6        int g = originalRGB[1] + Math.round(differenceRGB[1] * fraction);
7        int b = originalRGB[2] + Math.round(differenceRGB[2] * fraction);
8        cell.setBackground(r, g, b);
9      } else {
10       cell.setBackground(originalRGB[0], originalRGB[1], originalRGB[2]);
11     }
12   }
13 }
```

Beispiel anhand einer Farbwechsel-Animation

9.5.4 *Buttons*

Für die Buttons wurde eine abstrakte Klasse ImageButton erstellt, welche die gesamte Funktionalität eines Buttons enthält. Der Button wird dabei über ein zu definierendes Bild dargestellt. Die abgeleiteten Klassen müssen dabei nur die abstrakte Methode «buttonReleased» implementieren, in der die auszuführende Aktion definiert wird.

9.6 TABELLE

Für die Darstellung der Kategorien und Attribute eines Smartphones in einer übersichtlichen Form eignet sich eine Tabelle ideal. Seitens des Frameworks stand keine Tabelle zur Verfügung. Diese musste von Grund auf konzipiert und programmiert werden.

9.6.1 *Aufbau der Tabelle*

Der Indexzugriff beginnt bei 0

Das Ziel bei der Konzeption der Tabelle war, dass auf jede Tabellenzelle mit Hilfe eines Index zugegriffen werden kann. Hierfür muss sowohl die Spalte wie auch die Zeile bekannt sein. Dabei werden die Spalten wie auch die Zeilen aufsteigend, beginnend beim Wert 0 durchnummeriert. Ein Zugriff auf die Zelle in der dritten Spalte und der zweiten Zelle wäre demnach möglich mit dem Indexpaar (2,1). Um beim Zellenzugriff sämtliche Möglichkeiten zu bieten, müssen alle Elemente der Tabelle von der Klasse SceneGraphObject abgeleitet sein.

Die Tabelle wird über die Spalten aufgebaut

Bei der Erstellung der Tabelle mussten wir uns entscheiden, ob wir die Tabelle über die Spalten oder die Zeilen aufbauen. Wir entschieden uns für den Aufbau über die Spalten, das heisst es werden zuerst die Spalten gezeichnet und erst nachher werden für jede Spalte die Zeilen gezeichnet. Dadurch kann auf Tabellenspalten direkt über einen Index zugegriffen werden. Bei einer Zeile ist dies nicht direkt und nur über Umwege (eine Selektion aller Zellen mit dem gleichen Zeilenindex) möglich.

9.6.2 *Zellpositionierung*

Für die Positionierung des Zellinhaltes wurden sowohl für die vertikale wie auch die horizontale Richtung je drei Positionen festgelegt. Dies führt zu einem Total von neun Positionen des Zellinhaltes (siehe Abbildung 71). Intern kann diese Positionierung über zwei Enumwerte angepasst werden.

```
1 public enum HorizontalAlignment {
2     LEFT, CENTER, RIGHT
3 }
4
5 public enum VerticalAlignment {
6     TOP, MIDDLE, BOTTOM
7 }
```

Enumwerte für die Zellausrichtung

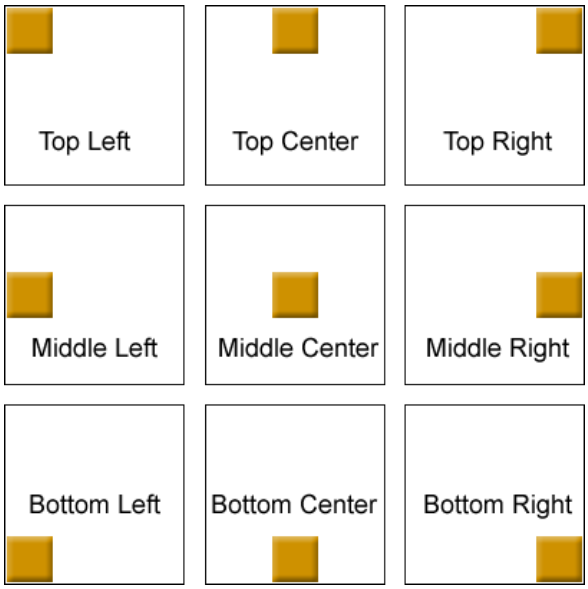


Abbildung 71: Die möglichen Zellanordnungen

9.6.3 *Initialisierung einer simplen Tabelle*

Anhand eines kleinen und einfachen Beispiels zeigen wir die Funktion der Tabelle auf. Die einzelnen Codeelemente bauen dabei aufeinander auf.

Die Initialisierung einer Tabelle erfolgt idealerweise mit vorbestimmten Spalten- und Zeilengrössen. Selbstverständlich ist eine Manipulation der Spalten und Zeilen nachher weiterhin möglich. In dem untenstehenden Beispiel werden zuerst die Spaltenbreiten und Zeilenhöhen bestimmt. Danach wird die Tabelle kreiert und eine Hintergrundfarbe gesetzt. Der Boolean beim Konstruktor bestimmt, ob die Tabelle runde Ecken hat oder nicht.

```
1  ArrayList<Integer> rowWidths = new ArrayList<Integer>();
2  rowWidths.add(80);
3  rowWidths.add(120);
4  rowWidths.add(60);
5
6  ArrayList<Integer> columnHeights = new ArrayList<Integer>();
7  columnHeights.add(60);
8  columnHeights.add(120);
9  columnHeights.add(80);
10
11 Table table = new Table("table", rowWidths, columnHeights, false);
12 table.setBackground(30, 52, 85);
```

Initialisierung einer Tabelle mit definierten Spalten- und Zeilenwerten.

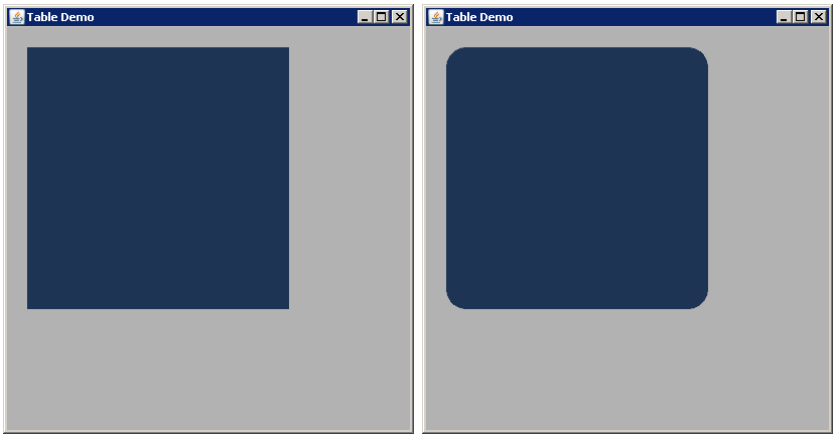


Abbildung 72: Die Beispielstabelle mit und ohne runde Ecken

9.6.4 *Hinzufügen einer Spalte*

Wir möchten eine neue Spalte bei einem bestimmten Index einfügen. Zusätzlich soll der Hintergrund dieser Spalte eine andere Farbe besitzen. Über den Transparenzwert rowAlpha kann die Transparenz des Zellinhaltes (nicht der Zelle) festgelegt werden.

```
1 int rowIndex = 1;
2 int rowWidth = 100; //in Pixel
3 float rowAlpha = 1f;
4 TableRow addedRow = table.addRowAtPosition(rowIndex, rowWidth, columnHeights
      , rowAlpha);
5 //RGB-Farbraum 0-255
6 addedRow.setBackground(200, 120, 80);
```

Hinzufügen einer Spalte an einem bestimmten Index

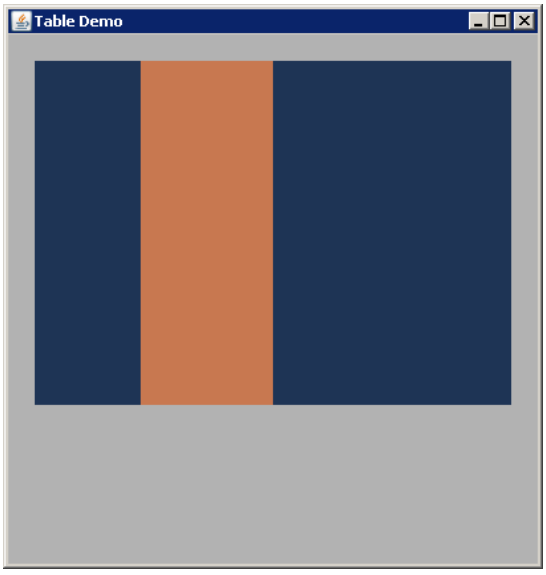


Abbildung 73: Tabelle mit hinzugefügter Spalte

9.6.5 *Hinzufügen einer Zeile*

Der Einsatz einer neuen Zelle gestaltet sich schwieriger als derjenige einer neuen Spalte. Dies ist mit dem Aufbau der Tabelle über die Spalten zu erklären. Sprich die Tabellenzeilen können nicht direkt angewählt werden. Hierfür wurden entsprechende Methoden integriert, die grundsätzliche Aufgaben wie die Änderung eines Hintergrunds einer Zeile erleichtern.

```
1 int columnIndex = 2;
2 int columnHeight = 50; //in Pixel
3 float columnAlpha = 1f;
4 table.addColumnAtPosition(columnIndex, columnHeight, columnAlpha);
5 table.setColumnBackground(columnIndex, 170, 180, 200);
```

Hinzufügen einer Zeile an einem bestimmten Index

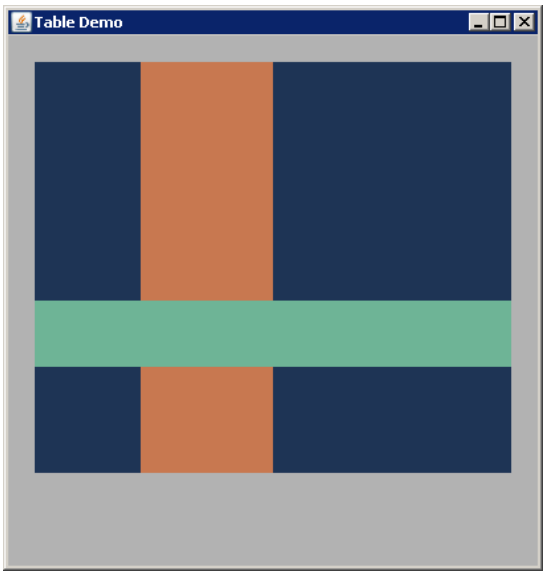


Abbildung 74: Tabelle mit hinzugefügter Zeile

9.6.6 *Setzen eines Rahmens*

Ein Rahmen kann auf eine beliebige Zelle gesetzt werden. Dafür wird eine Zelle der Tabelle benötigt. In diesem Falle wählen wir die Zelle am Schnittpunkt der zuvor eingefügten Spalte und Zeile. Anhand des Enums BorderType kann der Rahmentyp festgelegt werden.

```
1 TableCell middleCell = table.getCell(rowIndex, columnIndex);
2 middleCell.setBackground(0, 0, 0);
3
4 //Reihenfolge = top, right, bottom, left
5 middleCell.setBorderLayout(BorderType.DASHED, BorderType.DOTTED, BorderType.
      SOLID, BorderType.NONE);
```

Setzen eines Rahmens um eine Zelle

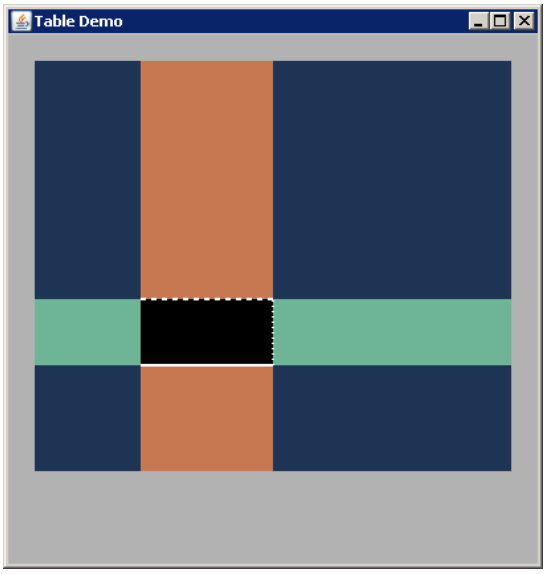


Abbildung 75: Tabelle mit gesetzten Rahmen in mittlerer Zelle

9.6.7 *Der Zellinhalt*

In eine Tabellenzelle kann ein beliebiger Zellinhalt eingesetzt werden. Theoretisch ist somit innerhalb einer Zelle alles möglich, was sich mit der Hilfe von OpenGL und einem SceneGraphObject umsetzen lässt. Ein Zellinhalt muss dabei zwingend von der Klasse CellContent abgeleitet sein und die Methoden getHeight() und getWidth() implementieren. Aufgrund dieser Methoden kann die Zellausrichtung exakt bestimmt werden. Ansonsten ist dies nicht möglich. In diesem Beispiel verwenden wir eine Zelle die ein Häkchen als Textur darstellt.

Der Zellinhalt wird anschliessend der ersten Zelle der Tabelle hinzugefügt. Zusätzlich wird die Zellausrichtung entsprechend gesetzt.

```
1  public class IconContent extends CellContent {
2    private Texture picTexture = null;
3
4    public IconContent(String name) {
5      super(name);
6    }
7
8    public void drawShape(GL gl) {
9      super.drawShape(gl);
10     if (picTexture != null) {
11       PrimitiveUtil.drawTexture(gl, picTexture, 0, 0, picTexture
12           .getWidth(), picTexture.getHeight());
13     }
14   }
15
16   public int getHeight() {
17     if (picTexture != null) {
18       return picTexture.getHeight();
19     } else {
20       return 0;
21     }
22   }
23
24   public int getWidth() {
25     if (picTexture != null) {
26       return picTexture.getWidth();
27     } else {
28       return 0;
29     }
30   }
31
32   public void update() {
33     if (picTexture == null) {
34       picTexture = TextureLoader.getTexture("checkmark.png");
35     }
36   }
37 }
```

Die Klasse IconContent stellt ein Häkchen als Textur dar

```
1 TableCell firstCell = table.getCell(0, 0);
2 IconContent iconContent = new IconContent("checkmark");
3 firstCell.setContent(iconContent,HorizontalAlignment.LEFT, VerticalAlignment
      .TOP);
```

Hinzufügen des IconContents bei der ersten Zelle

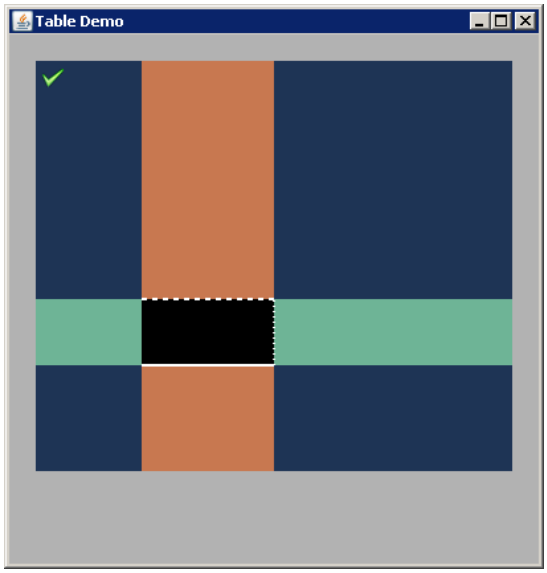


Abbildung 76: Tabelle mit Zellinhalt in der ersten Zelle

9.6.8 *Aktionsbehandlung*

Bei einer Aktion auf einer Tabellenzelle wird mit dem Einsatz eines Observers der ActionController benachrichtigt. Dabei wird der Stroke, die Tabellenzelle und der Zustand des Strokes (neuer Stroke oder alter, «toter» Stroke) mitgeteilt. Ist bereits ein Stroke auf einer Tabellenzelle aktiv, kann kein weiterer Stroke auf derselben Zelle erfasst werden.

```
1 public void handleAction(Stroke stroke, TableCell cell, boolean isDeadStroke
      ) {
2   setChanged();
3   notifyObservers(new StrokeArgument(stroke, cell, isDeadStroke));
4 }
```

Benachrichtigung der Observer bei einer Zellaktion

9.7 EVENTDISPATCHER

Für den Ablauf von Animationen musste eine Möglichkeit gefunden werden, diese nacheinander auszuführen. Dies wurde durch einen EventDispatcher gelöst, welcher von der Webseite http://java.schst.net/EventDispatcher stammt.

Dank des Observer-Pattern können Animationen nacheinander gestartet werden

Dadurch konnte die Problematik von nacheinander ablaufenden Animationen jedoch noch nicht gelöst werden, da diese jeweils in einem separaten Thread ablaufen und der EventDispatcher dadurch alle Animationen startet, aber nicht vor dem Start einer Animation auf die Beendigung der vorangegangenen Animation wartet.

Deshalb wurde der EventDispatcher für diese Arbeit erweitert. Dabei wurde der EventDispatcher als Observer auf die gestarteten Animationen gesetzt, wodurch er nach Beendigung der Animation benachrichtigt werden konnte.

```
1  public void update(Observable o, Object arg) {
2    Event e = (Event) arg;
3    // Entfernt ausgeführtes Element
4    event.remove(e);
5    if (event.size() > 0) {
6      try {
7        backFromUpdate = true;
8        // führt nächsten Befehl in der Queue aus
9        propagate(event.remove(0), true);
10     } catch (Exception exc) {
11       ...
12     }
13   } else {
14     setHasTopEvent(false);
15   }
16 }
```

Durch Benachrichtigung gesteuerte Abarbeitung

Zusätzlich musste eine Möglichkeit eingebaut werden, dass bei elementaren Veränderungen keine weiteren Anpassungsanimationen ermöglicht werden. Dies wird über das Flag «hasTopEvent» erreicht. Da dieses Flag allerdings vor dem Hinzufügen der Animationen gesetzt werden muss, wird dies nicht vom EventDispatcher selbst gesetzt. Der

EventDispatcher setzt das Flag jedoch zurück, wenn keine weitere Animation ausgeführt wird oder sich in der Warteschlange befindet.

9.8 WICHTIGE HILFSKLASSEN

9.8.1 *MethodInvoker*

Der MethodInvoker kann beliebige Methoden mit dem Einsatz von Reflection aufrufen. Dies ist nötig, da Methoden zeitversetzt aufgerufen werden müssen. Das Beispiel in der Abbildung 77 zeigt dies auf.

Abbildung 77: Beispiel verzögerter Methodenaufruf

Mittels Reflection wird die Methode ausgeführt

Dabei muss die Aktion «neue Daten laden» erst nach der Animation «Alte Daten ausblenden» ausgeführt werden. Hier kommt der MethodInvoker zum Einsatz. Er wird vom EventDispatcher aufgerufen und kann die Methode dann ausführen. Nachdem die Methode ausgeführt wurde, wird der Observer des EventDispatchers benachrichtigt, damit die nächste Aktion innerhalb des EventDispatchers ausgeführt werden kann.

```
1  public void handleEvent(Event e) throws Exception {
2    //Falls die Methode vom EventDispatcher ausgeführt wird, Observer
         hinzufügen.
3    if (disp != null) {
4      addObserver(disp);
5    }
6    invokeMethod();
7    setChanged();
8    notifyObservers(e);
9    deleteObservers();
10 }
11
12 private void invokeMethod() {
13   try {
14     if (methodParams != null) {
15       classType.getDeclaredMethod(methodName, methodParams).invoke(
16           object, paramValues);
17     } else {
18       classType.getDeclaredMethod(methodName).invoke(object);
19     }
20   } catch (Exception e) {
21     ChargerLogger.log(Level.WARNING, "unable to invoke method", e);
22   }
23 }
```

Aufruf einer Methode über den MethodInvoker

9.8.2 *TextureLoader*

Der TextureLoader ist eine statische Klasse, die sämtliche für die Applikation benötigte Texturen lädt. Dabei können mit einem Methodenaufruf sämtliche Bilder in einem Verzeichnis und dessen dazugehörigen Unterverzeichnissen geladen werden. Hierfür ist ein rekursiver Aufruf von Nöten. Mittels einer Regular Expression wird die Endung einer Datei überprüft. Da alle Bilder und Icons im PNG-Format gespeichert wurden, musste der Dateiname lediglich auf diese Endung geprüft werden.

```
...
if (!file.isDirectory()) {
  // Auf Dateityp überprüfen, in diesem Falle nur .png-Dateien annehmen.
  if (filename.matches(".*\\.png")) {
    addTexture(directoryPath + File.separatorChar + filename);
  }
} else {
  // rekursiver Aufruf
  loadDirectory(directoryPath + File.separatorChar + filename);
}
...
```

Laden von Bilddateien eines Verzeichnisses

Die Texturen werden innerhalb einer Map mit dem Dateipfad als Index gespeichert und können demzufolge über den Dateipfad wieder aufgerufen werden.

```
private static Texture addTexture(String path) {
  try {
    Texture picTexture = TextureIO.newTexture(new File(path), false);
    textureMap.put(path, picTexture);
    return picTexture;
  } catch (GLException e) {
    ChargerLogger.log(Level.WARNING, "could not load texture: " + path,
        e);
  } catch (IOException e) {
    ChargerLogger.log(Level.WARNING, "could not load texture: " + path,
        e);
  }
  return null;
}
```

Laden einer separaten Bilddatei

```
public static Texture getTexture(String path) {
  // Umwandeln der Slashes in Backslashes
  String replacedPath = path.replace('/', File.separatorChar);
  if (textureMap.containsKey(replacedPath)) {
    return textureMap.get(replacedPath);
  } else {
    return addTexture(replacedPath);
  }
}
```

Lesen einer Textur

9.8.3 *Constants*

Im Verlaufe der Arbeit haben sich sehr viele Konstanten angesammelt. Aufgrund der Tatsache, dass es sich um einen Prototyp handelt, wurden die Konstanten nicht vollständig externalisiert. Sie wurden aber in einer separaten statischen Klasse gespeichert.

```
public static final int[] COLUMN_RGB = { 50, 50, 50 };
```

Beispiel einer Konstante

10 TESTING UND QUALITÄTSSICHERUNG

10.1 QUALITÄTSSICHERUNG

Die folgenden Tools und Massnahmen wurden während des Projekts verwendet:

10.1.1 *Regelmässige Code-Reviews*

Um die Qualität des Codes hoch zu halten, wurden regelmässige Code-Reviews durchgeführt.

10.1.2 *Paarweises Programmieren*

Bei komplexen Aufgaben wurde oftmals paarweise programmiert. Dies beeinträchtigt zwar die Entwicklungsgeschwindigkeit, ist jedoch bezüglich der Codequalität ein klarer Vorteil.

10.1.3 *Checkstyle*

Checkstyle überprüft den Code nach formalen Fehlern. Unter anderem werden fehlende Javadoc-Kommentare notiert. Der Code kann dadurch vereinheitlicht werden.

http://checkstyle.sourceforge.net/

10.1.4 *Metrics*

Metrics hilft bei der Optimierung des Codes. Es zeigt unter anderem zu grosse Klassen und Methoden und zyklische Abhängigkeiten an.

http://metrics.sourceforge.net/

10.1.5 *Code-Formatierung*

Bei der Code-Formatierung wurde auf die von Eclipse integrierte Formatierung zurückgegriffen.

10.1.6 *Ant*

Ant hilft bei der automatisierten Erstellung von Software-Paketen.

http://ant.apache.org/

10.2 VERWENDETE PROGRAMME

Bei der Projektausführung waren verschiedene Geräte und Tools von Nöten.

- Hardware
  - Multi-Touch-Tisch
  - Der zum Multi-Touch-Tisch dazugehörige Computer mit der Treibersoftware
  - SVN-Server vom IFS (Institut für Software)
  - Arbeitsstationen und Drucker im Raum 1.258
  - Private Notebooks
- Software
  - Eclipse IDE
  - MySQL
  - MySQL Workbench Visual Database Designer
  - TortoiseSVN
  - TeXnicCenter
  - Java JDK 1.6
  - Enterprise Architect
  - Metrics-Plugin für Eclipse
  - Checkstyle-Plugin für Eclipse
  - Ant

10.3 JUNIT-TESTS

10.3.1 *Allgemeines*

Die JUnit-Tests wurden für komplizierte Methoden bzw. Klassen erstellt und sollen den korrekten Ablauf garantieren. Da ein Grossteil der Applikation die Benutzeroberfläche ausmacht, konnten nur wenige Operationen mittels JUnit-Tests getestet werden.

Alle JUnit-Tests befinden sich im Package «test».

10.3.2 *DatabaseReadTest*

Der DatabaseReadTest überprüft das Lesen aus der Datenbank und die Speicherung der Daten im Model. Zusätzlich wird bei einem unbekannten Element überprüft, ob die ObjectNotFoundException geworfen wird.

Da einige Strings überprüft werden, funktioniert dieser Test selbstverständlich nur mit der mitgelieferten Datenbank. Werden die Daten der Datenbank geändert, muss der DatabaseReadTest entsprechend angepasst werden.

10.3.3 *PatternRecognizerTest*

Der PatternRecognizerTest testet den PatternRecognizer auf dessen mathematische Korrektheit. Dabei werden sämtliche 64 mögliche Pattern überprüft. Um die Rotation des Gerätes zu simulieren, wird jedes dieser Pattern mit Winkeln zwischen 0-360 Grad getestet. Der Abstand zwischen den einzelnen Winkeln beträgt hierbei 0.1 Grad. Das heisst, dass insgesamt

$$64 * (360/0.1) = 230400$$

Kombinationen getestet werden.

10.4 SYSTEMTESTS

10.4.1 *Allgemeines*

Aufgrund der vielen GUI-Elemente mussten sehr viele Systemtests durchgeführt werden. Auch die Eingabe über den Multi-Touch-Tisch konnte nicht mittels Unit-Tests überprüft werden. Deswegen wurden viele Systemtests konzipiert und periodisch durchgeführt.

10.4.2 *Voraussetzungen*

Die Datenbank und der Tracker[1] müssen gestartet sein.

10.4.3 *Allgemeine Tests*

T01 - SMARTPHONE PLATZIEREN Es wird ein Smartphone auf dem Tisch platziert. Die Applikation zeigt daraufhin das entsprechende Smartphone an.

T02 - SMARTPHONE WEGNEHMEN Das Smartphone wird vom Tisch entfernt. Die Applikation zeigt die Daten des Smartphones weiterhin an.

T03 - TISCHANZEIGE DREHEN Die Tischanzeige wird durch einen Druck auf den Drehbutton um 180 Grad gedreht.

T04 - TISCHANZEIGE ZURÜCKSETZEN Die Tischanzeige wird durch einen Druck auf den Resetbutton zurückgesetzt.

TESTFALL	OK	FEHLER	BEREINIGT
T01 - Produkt platzieren	X		
T02 - Smartphone wegnehmen	X		
T03 - Produkt platzieren	X		
T04 - Produkt platzieren	X		

Tabelle 29: Testfälle Allgemeine Tests

1 Der Tracker ist der Treiber für den Multi-Touch-Tisch

10.4.4 *Kategorietests*

T05 - KATEGORIE AUFKLAPPEN Eine Kategorie wird angewählt. Die derzeit geöffnete Kategorie wird geschlossen und die gewünschte Kategorie wird mitsamt ihren Attributen angezeigt.

T06 - KATEGORIE SCHLIESSEN Die derzeit geöffnete Kategorie wird angewählt. Daraufhin wird diese geschlossen und die entsprechenden Attribute werden nicht mehr angezeigt.

T07 - KATEGORIE VERSCHIEBEN Es wird eine beliebige Kategorie angewählt. Diese wird verschoben und anschliessend vom System an der gewünschten Stelle angezeigt.

T08 - KATEGORIE ENTFERNEN Es wird eine Kategorie angewählt. Diese wird über den Tabellenrand hinweg geschoben. Die Applikation entfernt die Kategorie in der Tabelle und stellt sie am linken Seitenrand dar.

T09 - KATEGORIE EINFÜGEN Es wird eine inaktive Kategorie am linken Rand ausgewählt. Diese wird in die Tabelle geschoben. Die Applikation fügt die Kategorie der Tabelle an der entsprechenden Stelle hinzu.

T10 - DRAG-EFFEKT Wird eine Kategorie verschoben, kreiert die Applikation eine Kopie dieser Kategorie als visuelles Feedback.

TESTFALL	OK	FEHLER	BEREINIGT
T05 - Kategorie aufklappen	X	X	X
T06 - Kategorie schliessen	X		
T07 - Kategorie verschieben	X	X	X
T08 - Kategorie entfernen	X		
T09 - Kategorie einfügen	X		
T10 - Drag-Effekt	X		

Tabelle 30: Testfälle Kategorietests

FEHLERFALL T05 - KATEGORIE AUFKLAPPEN Teilweise klappte nicht die gewünschte, sondern eine andere Kategorie auf.

FEHLERFALL T07 - KATEGORIE VERSCHIEBEN Das System stürzte beim Verschieben der Kategorie über- und unterhalb der Tabelle ab.

Die Indexwerte der Tabelle wurden nicht immer korrekt nachgetragen. Durch eine Neuberechnung der Tabellenindizes konnten beide Probleme behoben werden.

10.4.5 *Produktradtests*

T11 - KLICK AUF EIN SMARTPHONE Es wird auf ein Smartphone im Produktrad geklickt. Das System dreht daraufhin das Rad bis zum gewünschten Smartphone.

T12 - DREHEN DES RADS Das Rad wird gedreht mit einer Bewegung nach oben, resp. unten. Das System dreht das Rad anhand der Geschwindigkeit der Bewegung und bremst am Schluss der Bewegung nicht abrupt ab, sondern dreht noch ein wenig weiter.

T13 - PLATZIEREN EINES SMARTPHONE Es wird ein Smartphone vom Produktrad in die Tabelle gezogen. Das Produkt wird daraufhin an der gewünschten Stelle in der Tabelle eingefügt und aus dem Produktrad entfernt.

T14 - SMARTPHONE ZUM RAD HINZUFÜGEN Es wird ein Smartphone von der Tabelle zum Produktrad gezogen. Die Applikation entfernt das Smartphone von der Tabelle und fügt es dem Produktrad hinzu. Zusätzlich wird das Rad automatisch an die Stelle des neu hinzugefügten Smartphones gedreht.

T15 - SMARTPHONE VON RAD ZU RAD BEWEGEN Hier müssen zwei Fälle unterschieden werden. Wird das Smartphone vom an der linken Seite positionierten Rad zum rechts positionierten Rad gezogen, ist dies möglich. Die umgekehrte Aktion (das heisst von rechts nach links) ist jedoch nicht gestattet. Während beim zweiten Fall nichts passiert, wird beim ersten Fall das Smartphone im linken Produktrad entfernt und beim rechten Produktrad eingefügt. Zusätzlich wird das Rad automatisch an die Stelle des neu hinzugefügten Smartphones gedreht.

TESTFALL	OK	FEHLER	BEREINIGT
T11 - Klick auf ein Produkt	X		
T12 - Drehen des Rads	X	X	X
T13 - Platzieren eines Smartphone	X		
T14 - Smartphone zum Rad hinzufügen	X		
T15 - Smartphone von Rad zu Rad bewegen	X		

Tabelle 31: Testfälle Produktradtests

FEHLERFALL T12 - DREHEN DES RADS Die Drehbewegung konnte über das oberste respektive unterste Element hinaus geführt werden. Dadurch waren die Elemente danach nicht mehr erreichbar und konnten somit nicht mehr angewählt werden.

Das Problem konnte durch eine minimale und eine maximale Begrenzung des Drehbereichs behoben werden.

10.4.6 *Tabellentests*

T16 - SMARTPHONE ENTFERNEN MIT FLICK Es wird ein Flick auf einem HeaderContent ausgeführt. Daraufhin wird das Produkt entfernt und zum linken Produktrad hinzugefügt.

T17 - SMARTPHONES AUSTAUSCHEN Es werden zwei Spalten der Tabelle ausgetauscht.

T18 - DRAG-EFFEKT Wird ein Smartphone verschoben, kreiert die Applikation eine Kopie dieses Smartphones als visuelles Feedback.

TESTFALL	OK	FEHLER	BEREINIGT
T16 - Smartphone entfernen mit Flick	X		
T17 - Smartphones austauschen	X		
T18 - Drag-Effekt	X		

Tabelle 32: Testfälle Tabellentests

10.5 METRIKEN

Die Metriken wurden mit dem Eclipse-Plugin Metrics erstellt. Sämtliche Javadoc-Kommentare wurden dabei nicht berücksichtigt.

10.5.1 *Codestatistiken*

STATISTIK	WERT
Anzahl Pakete	15 Pakete
Anzahl Klassen	75 Klassen
Anzahl Zeilen Code	5534 Zeilen
Anzahl Codezeilen pro Coder pro Woche	163 Zeilen

Tabelle 33: Codestatistiken

10.5.2 *Tabelle*

Da die Tabelle möglicherweise den Weg in das Charger-Framework findet, wurde bei der Qualitätssicherung des Codes besonderen Wert darauf gelegt. Die mit Metrics generierten Statistiken bestätigen dies. So wurde versucht, die Methoden möglichst kurz und prägnant zu halten (siehe Abbildung 78) sowie die Anzahl Variablen im lokalen Bereich möglichst gering zu halten (siehe Abbildung 79).

Zusätzlich wurde die Anzahl verschachtelter Ebenen in einer Methode gering gehalten (siehe Abbildung 80).

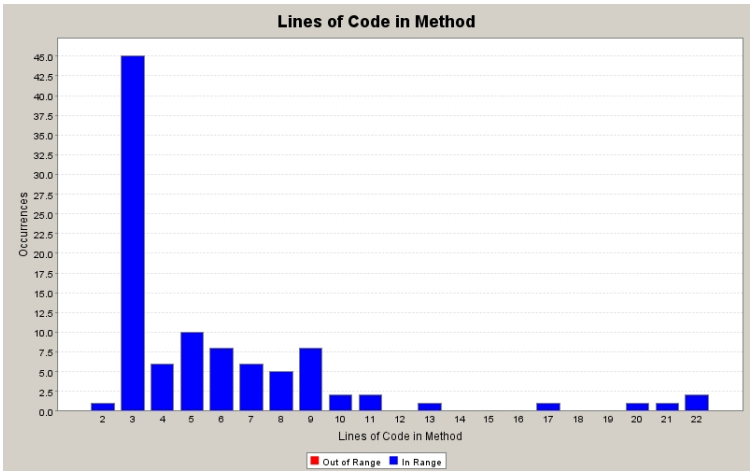


Abbildung 78: Anzahl Zeilen Code pro Methode

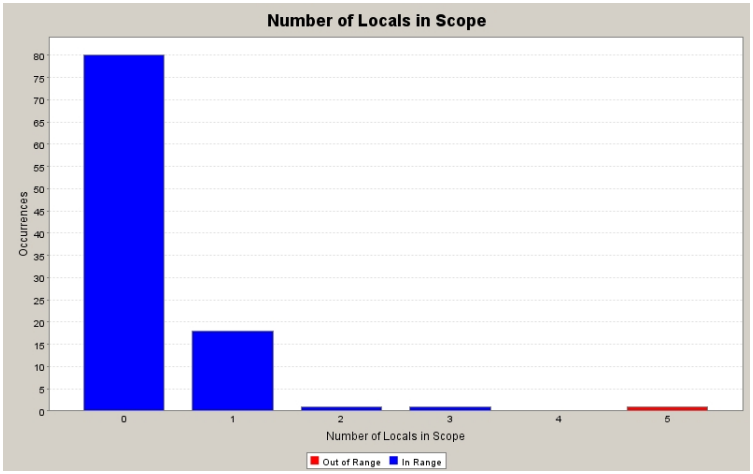


Abbildung 79: Anzahl Variablen im lokalen Bereich

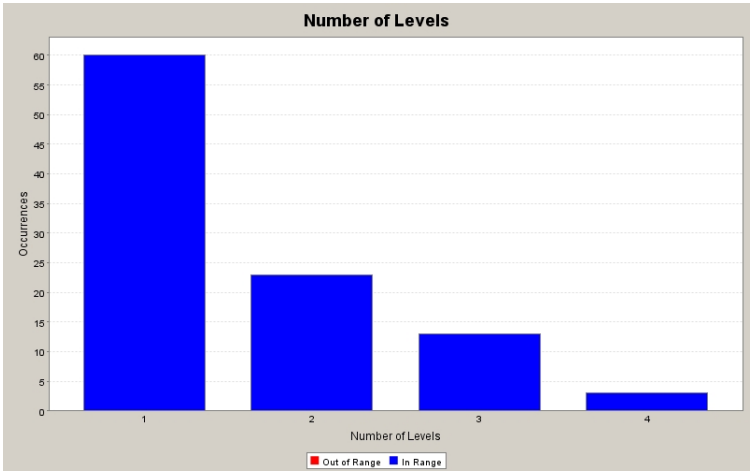


Abbildung 80: Anzahl verschachtelter Ebenen pro Methode

11 EVALUATION

11.1 RESULTATE

Das Resultat der Arbeit ist eine funktionierende Applikation für den zur Verfügung gestellten Multi-Touch-Tisch. Unter anderem wurden die folgenden Features implementiert:

11.1.1 *Erkennungsalgorithmus*

Die Applikation erkennt auf dem Multi-Touch-Tisch platzierte Smartphones, die mit einem festgelegten und eindeutig identifizierbaren Muster versehen sind. Dabei werden sämtliche Positionen des Multi-Touch-Tisches sowie sämtliche Drehwinkel bei der Platzierung des Smartphones unterstützt. Mit Hilfe von Unit-Tests konnte die mathematische Korrektheit des Erkennungsalgorithmus überprüft werden.

11.1.2 *Datenbankanbindung*

Die Daten der Smartphones werden aus einer Datenbank gelesen. Dabei wurde Hibernate eingesetzt. Dies ermöglicht eine schnelle Anpassung an ein unterschiedliches Datenbankmanagementsystem.

11.1.3 *Tabellarische Darstellung*

Für die Darstellung der Daten wurde eine Tabelle implementiert. Dies war notwendig, da unter OpenGL keine Tabellen zur Verfügung standen. Die Tabelle wurde bewusst einfach gehalten um eine spätere Integration in das Charger-Framework zu ermöglichen.

11.1.4 *Bedienung der Applikation mit dem Multi-Touch-Tisch*

Die Applikation ist vollständig über den Multi-Touch-Tisch bedienbar. Die Interaktion geschieht hierbei mit den Fingern oder mit Smartphones, die mit einem festgelegten und eindeutig identifizierbaren Muster versehen sind.

11.1.5 *Benutzeroberfläche komplett in OpenGL*

Die Benutzeroberfläche wurde vollständig mit OpenGL umgesetzt. Durch den Gebrauch von OpenGL läuft die Applikation sehr performant und die Animationen können problemlos ausgeführt werden.

11.1.6 *Grafische Animationen*

Um eine hohe Attraktivität der Applikation gewährleisten zu können, wurden diverse Animationen eingesetzt. Dazu gehören unter anderem verschiedene Blend- und Verschiebungseffekte. Bei der Verschiebung eines grafischen Elements wird ein entsprechendes grafisches Feedback eingesetzt.

Am Beispiel des Produktrads zeigt sich dies sehr gut.

11.1.7 *Drehen der Anzeige*

Die Anzeige der Applikation lässt sich komplett um 180 Grad drehen. Dies ist insbesondere dann nützlich, wenn bei einem Verkaufsgespräch der Verkäufer auf der einen Seite des Tisches und der Käufer auf der anderen Seite des Tisches steht.

11.1.8 *Zurücksetzen der Anzeige*

Die Anzeige der Applikation kann über Knopfdruck in den Anfangszustand zurückgesetzt werden. Zusätzlich wird die Anzeige automatisch nach einer definierbaren Zeit ohne Benutzereingaben zurückgesetzt.

11.1.9 *Vorschlag ähnlicher Produkte*

Der Benutzer der Applikation kann die Kategorien unterschiedlich anordnen. Diese Kategorien werden dabei je nach Anordnung unterschiedlich gewichtet. Anhand dieser Gewichtung sucht die Applikation nach ähnlichen Produkten und stellt diese in einem Produktrad dar.

11.1.10 *Anpassung der Kategoriepräferenzen*

Der Benutzer kann für ihn uninteressante Kategorien aus der Tabelle entfernen und nur die für ihn wichtigen Kategorien anzeigen lassen. Dies beeinflusst den Vorschlag ähnlicher Produkte.

11.1.11 *Erweiterung des Eventdispatchers*

Die Applikation verwendet einen bereits implementierten Eventdispatcher[1]. Dieser ist zuständig für die Auslösung von Events. Dies funktionierte jedoch bei Threads nicht wie gewünscht, da der Dispatcher nicht die Beendigung eines Threads abwartete.

Eine Erweiterung mit dem Einsatz des Observer-Patterns behob dieses Problem. Nun wartet der Dispatcher auf die Beendigung des Animationsthreads.

1 Der Eventdispatcher ist unter http://java.schst.net/EventDispatcher zu finden

11.2 VERBESSERUNGS- UND ERWEITERUNGSMÖGLICHKEITEN

11.2.1 *Externalisierung*

Im Rahmen dieses Projekts wurden sämtliche Konstanten in einer Datei zusammengefasst. Diese befindet sich jedoch ebenfalls im kompilierbaren Bereich. Für eine komplette externe Konfiguration müssten diese Konstanten in ein anderes Format (beispiel XML) übertragen werden.

11.2.2 *Datenverwaltung*

Die Datenverwaltung muss derzeit mit einem direkten Eingriff in die Datenbank vorgenommen werden. Dadurch können logischerweise Fehleinträge entstehen. Dies könnte mit einer separaten Datenverwaltungsapplikation verhindert werden.

11.2.3 *Erkennung von Gesten*

Aus Zeitgründen können zwei Funktionalitäten der Applikation (Drehen und Zurücksetzen) nur über Buttons ausgelöst werden. Ursprünglich waren hierfür Gesten vorgesehen.

11.2.4 *Änderung des Erkennungsmusters*

Prinzipiell wäre es möglich, die Produkte über andere Erkennungsmuster zu erkennen. Beispielsweise wäre der Einsatz eines Barcodes denkbar. Hierfür müsste jedoch die Hardware und der Treiber modifiziert werden.

11.3 PROBLEME UND EINSCHRÄNKUNGEN

11.3.1 *Lichteinfall auf Tischoberfläche*

Die Oberfläche des Multi-Touch-Tisches ist äussert lichtsensitiv. Dies hat zur Folge, dass bei Lichteinfall die Applikation nicht mehr wunschgemäss bedient werden kann. Da die Problemlösung mögliche Änderungen an der Hardware und am Treiber zur Folge hätte, konnte dieses Problem während dieser Arbeit nicht behoben respektive verringert werden.

11.3.2 *Mindestabstand von zwei Objekten*

Liegen zwei Objekte sehr nahe beieinander auf dem Tisch, werden diese als ein zusammenhängendes Objekt erkannt. Diese Einschränkung hatte zur Folge, dass die Abstände beim Pinmuster verhältnismässig gross gewählt werden mussten. Das Problem liegt hierbei wiederum beim Treiber und der Hardware.

11.3.3 *FPS-Einbrüche*

Teilweise traten beim Betrieb des Multi-Touch-Tisches massive FPS-Einbrüche[2] bei der im Tisch integrierten Kamera auf.

Eine Problemlösung besteht darin, das Firewire-Kabel zwischen Computer und Multi-Touch-Tisch aus- und wieder einzustecken. Während der Arbeit nützte auch dies nichts, da die alte Kamera defekt war. Diese musste durch Christian Iten von to-fuse[3] mit einer neuen Kamera ersetzt werden.

11.3.4 *Fingereingaben mit Erkennungsmuster*

Es ist möglich, dass neben den Pins des Erkennungsmuster auch die Finger des Benutzers erkannt werden. Sind diese nahe genug beim Muster selber, kann das Muster nicht erkannt werden. Eine mögliche Problemlösung ist die Verwendung eines grösseren Abstandes zwischen der Platine und dem Muster.

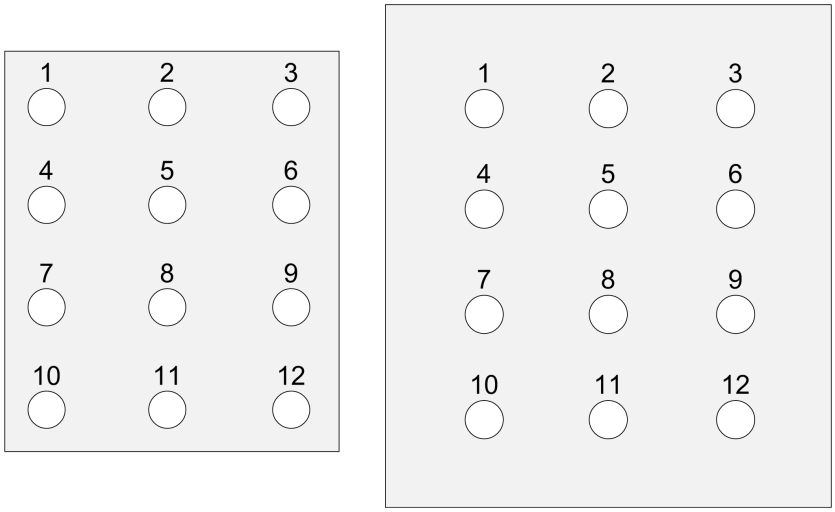


Abbildung 81: Eine vergrösserte Platte (rechts) verhindert Fehlinterpretationen

11.3.5 *Scrollen der Tabelle*

Die Applikation ist nicht in der Lage, innerhalb einer Tabelle zu scrollen. Bei einer Anzeige von vielen Daten müsste dies entsprechend erweitert werden.

2 FPS (Frames Per Second) ist das Mass der Bildfrequenz und bezeichnet die Anzahl der Bilder pro Sekunde
3 http://to-fuse.ch/

11.4 AUSBLICK

11.4.1 *Einsatzmöglichkeiten*

Als potentieller Einsatzort sticht natürlich sofort eine Verkaufsstelle eines Mobilfunkbetreibers heraus. Prinzipiell lässt sich die Applikation jedoch universell verwenden. So könnte die Applikation beispielsweise beim Verkauf von Schmuck oder bei Uhren ebenso eingesetzt werden. Zu berücksichtigen gilt dabei jedoch, dass eine stabile Standfläche vorhanden sein muss, an der das Muster angebracht werden kann.

11.4.2 *Erweiterungsmöglichkeiten der Hardware*

Die to-fuse stations werden individuell konfiguriert. Somit werden jederzeit die neuesten Entwicklungen im Bereich Beamer- und Kameratechnologien in das Gerät integriert. Auch in der Form und Materialität des Gehäuses können spezielle Kundenwünsche berücksichtigt werden. Zudem lassen sich mehrere to-fuse stations in einem Gehäuse mit nahtlosem Screen zu einer interaktiven Fläche von zwei bis drei Metern Länge kombinieren.

Teil III

ANHANG

A

GLOSSAR

Nachfolgend werden einige in der Dokumentation erwähnten Begriffe erläutert.

BEGRIFF	ERKLÄRUNG
Charger-Framework	Dieses Framework erkennt die Benutzereingaben und liefert grundlegende Darstellungsfunktionen
Drag & Drop	Drag & Drop, deutsch «Ziehen und Fallenlassen», ist eine Methode zur Bedienung grafischer Benutzeroberflächen von Rechnern durch das Bewegen grafischer Elemente mittels eines Zeigegerätes. http://de.wikipedia.org/wiki/Drag_and_Drop
Firewire	FireWire ist ein von Apple entwickeltes serielles Bussystem. http://de.wikipedia.org/wiki/FireWire
Fitts Law	Fitts Law besagt, dass die benötigte Zeit um mit einem Ziel zu interagieren abhängig von dessen Grösse und der Distanz ist. Je kleiner und weiter entfernt das Ziel ist, desto grösser wird der Zeitaufwand einer Interaktion.
Flick	Ein Flick ist eine kurze Ziehbewegung.
FPS	Die Abkürzung fps (für das englische Frames per Second) bezeichnet bei Film- und Videoaufnahmen sowie bei graphischen Computeranwendungen die Anzahl der Bilder pro Sekunde. http://de.wikipedia.org/wiki/Bildfrequenz
FTIR	Die Totalreflexion (engl. frustrated total internal reflection, FTIR) ist ein optisches Phänomen, bei dem elektromagnetische Strahlung an der Grenzfläche zweier Medien nicht gebrochen, sondern vollständig reflektiert wird, obwohl die Grenzfläche nicht beschichtet ist. http://de.wikipedia.org/wiki/Totalreflexion
GUI	Eine grafische Benutzeroberfläche (engl. Graphical User Interface, GUI) ist eine Software-Komponente, die dem Benutzer eines Computers die Interaktion mit der Maschine über grafische Symbole erlaubt. http://de.wikipedia.org/wiki/Grafische_Benutzeroberfläche
Hibernate	Hibernate ist ein Open-Source-Persistenz-Framework für Java. http://de.wikipedia.org/wiki/Hibernate_(Framework)
Hover	Hover bezeichnet das Schweben über einem Bereich. Wenn ein Mauszeiger über einem Bedienelement liegt, kann dieses Element dadurch einen Effekt anzeigen.
JOGL	Jogl (Java OpenGL) ist eine externe OpenGL-Programmbibliothek für die Programmiersprache Java. http://de.wikipedia.org/wiki/Jogl

BEGRIFF	ERKLÄRUNG
JUnit	JUnit ist ein Framework zum Testen von Java-Programmen, das besonders für automatisierte Unit-Tests einzelner Units (meist Klassen oder Methoden) geeignet ist. http://de.wikipedia.org/wiki/JUnit
LaTeX	LaTeX ist ein Softwarepaket, das die Benutzung des Textsatzprogramms TeX mit Hilfe von Makros vereinfacht. http://de.wikipedia.org/wiki/LaTeX
Multi-Touch	Multi-Touch ist ein System zur Mensch-Maschine-Interaktion. Es vereint Ein- und Ausgabegerät und kommt dadurch ohne herkömmliche Eingabegeräte wie Maus, Tastatur oder Stylus aus. http://de.wikipedia.org/wiki/Multi-Touch
MVC	Model-View-Controller (MVC, «Modell/Präsentation/-Steuerung») bezeichnet ein Architekturmuster zur Strukturierung von Software-Entwicklung in die drei Einheiten Datenmodell (engl. Model), Präsentation (engl. View) und Programmsteuerung (engl. Controller). http://de.wikipedia.org/wiki/MVC
MySQL	Der MySQL Server ist ein Relationales Datenbankverwaltungssystem. http://de.wikipedia.org/wiki/Mysql
OpenGL	OpenGL (Open Graphics Library) ist eine Spezifikation für eine plattform- und programmiersprachenunabhängige Programmierschnittstelle zur Entwicklung von 2D- und 3D-Computergrafik. http://de.wikipedia.org/wiki/Opengl
PNG	Portable Network Graphics ist ein Grafikformat für Rastergrafiken mit verlustfreier Bildkompression. http://de.wikipedia.org/wiki/Png
Reflection	In der Programmierung bedeutet Reflexion (engl. reflection) bzw. Introspektion, dass ein Programm seine eigene Struktur kennt und diese, wenn nötig, modifizieren kann. http://de.wikipedia.org/wiki/Reflection
RUP	Der Rational Unified Process (RUP) ist ein objektorientiertes Vorgehensmodell zur Softwareentwicklung und ein kommerzielles Produkt der Firma Rational Software, die seit 2002 Teil des IBM Konzerns ist. http://de.wikipedia.org/wiki/Rational_Unified_Process
SceneGraphObject	Das SceneGraphObject wird vom Charger-Framework zur Verfügung gestellt und bietet grundlegende Funktionen für die Objektmanipulation.
Smartphone	Ein Smartphone vereint den Leistungsumfang eines Mobiltelefons mit dem eines Personal Digital Assistants (PDA). http://de.wikipedia.org/wiki/Smartphone
Stroke	Eine Stroke ist ein erkannter Punkt auf dem Multi-Touch-Tisch
SVN	Subversion (SVN) ist eine Open-Source-Software zur Versionsverwaltung von Dateien und Verzeichnissen. http://de.wikipedia.org/wiki/Svn
TeX	TeX ist ein von Donald E. Knuth ab 1977 entwickeltes und 1986 fertig gestelltes Textsatzsystem mit eingebauter Makrosprache, die ebenfalls als «TeX» bezeichnet wird. http://de.wikipedia.org/wiki/TeX

BEGRIFF	ERKLÄRUNG
User Centered Design	Die nutzerorientierte Gestaltung zielt darauf ab, interaktive Produkte so zu gestalten, dass sie über eine hohe Gebrauchstauglichkeit (Usability) verfügen. Dies wird im Wesentlichen dadurch erreicht, dass der (zukünftige) Nutzer eines Produktes mit seinen Aufgaben, Zielen und Eigenschaften in den Mittelpunkt des Entwicklungsprozesses gestellt wird. http://de.wikipedia.org/wiki/User_Centered_Design
VARCHAR	Ein Varchar oder Variable Character Field ist eine Menge von Zeichen unbestimmter Länge. http://en.wikipedia.org/wiki/Varchar
XML	Die Extensible Markup Language, abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten. http://de.wikipedia.org/wiki/Xml

B

AUFGABENSTELLUNG

Auf den nachfolgenden Seiten ist die originale Aufgabenstellung zu finden.

[11. Juni 2009]

[11. Juni 2009]

C

BENUTZERDOKUMENTATION

C.1 VORAUSSETZUNGEN

Für den Betrieb der Applikation müssen die folgenden Voraussetzungen erfüllt sein:

- Java 1.6 muss vorhanden sein
- Ein 3D-Grafikbeschleuniger muss vorhanden sein (für die Unterstützung von OpenGL)
- Die Trackersoftware muss vorhanden und gestartet sein
- Die MySQL-Datenbank muss gestartet sein
- Der Multi-Touch-Tisch muss mit dem Computer verbunden sein

C.2 DIE BENUTZEROBERFLÄCHE

In diesem Abschnitt werden die grundlegenden Interaktionsmöglichkeiten der Applikation aufgezeigt.

Abbildung 82: Benutzeroberfläche und Terminologie der GUI-Elemente

C.2.1 *Die Startanzeige*

Beim Start der Applikation wird lediglich ein Hintergrundbild angezeigt. Durch das Auflegen eines mit einem Muster versehenen Smartphones, kann die Produktspalte eingeblendet werden.

Abbildung 83: Startanzeige

C.2.2 *Die Tabelle*

In der Tabelle werden die Produkteigenschaften der Smartphones angezeigt. Dabei sind die folgenden Interaktionen möglich:

ENTFERNEN EINES PRODUKTS Das Produkt kann aus der Tabelle entfernt werden, indem beginnend beim Kopf der Produktspalte (siehe Abbildung 84) das Produkt entweder zum Produktrad auf der rechten Seite oder zu den bereits angesehenen Produkten gezogen wird. Damit nicht jedes Produkt gezogen werden muss, ist zudem das Entfernen mit der Verwendung eines Flicks[1] nach links oder nach oben möglich. Bei der Verwendung des Flicks wird das entfernte Produkt dann automatisch zu den bereits angesehenen Produkte hinzugefügt.

Abbildung 84: Der Kopf einer Produktspalte

1 Ein Flick ist eine kurze Ziehbewegung.

AUSTAUSCHEN ZWEIER PRODUKTSPALTEN Um die Position zweier Produktspalten zu tauschen, kann wiederum beginnend beim Kopf der Spalte das Produkt auf die zu tauschende Spalte geschoben werden.

ÖFFNEN EINER KATEGORIE Mit einem Druck auf eine Bewertung oder eine Kategorie wird die entsprechende Kategorie geöffnet und die bis anhin geöffnete Kategorie geschlossen. Bei nochmaligem Druck auf dieselbe Kategorie wird diese geschlossen.

ENTFERNEN EINER KATEGORIE AUS DER TABELLE Durch den Zug einer Kategorie über den Rand der Tabelle wird diese Kategorie aus der Tabelle entfernt. Die Kategorien können nicht beginnend bei einer Bewertungszelle verschoben werden.

AUSTAUSCHEN DER KATEGORIEREIHENFOLGE Ebenfalls beginnend bei der Anzeige der Kategorie können innerhalb der Tabelle die verschiedenen Kategorien ausgetauscht werden (siehe Abbildung 85).

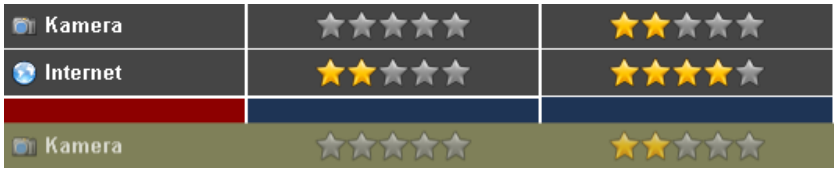


Abbildung 85: Visuelles Feedback beim Verschieben einer Kategorie

C.2.3 *Das Produktrad*

Das Produktrad zeigt Alternativen zu den platzierten Produkten an. Sowohl das Produktrad auf der rechten Seite wie auch das Produktrad bei den bereits angesehenen Produkten weisen die gleichen Bedienungsmöglichkeiten auf. Der einzige Unterschied dabei ist, dass vom linken Rad die Produkte zum rechten Rad gezogen werden können. In der Gegenrichtung ist dies jedoch nicht möglich. Die folgenden Interaktionen sind möglich:

DREHEN DES RADS Das Rad kann durch die Bewegung des Fingers nach oben oder nach unten gedreht werden.

DRUCK AUF EIN PRODUKT Bei einem Druck auf ein Produkt im Produktrad wird das Rad an die entsprechende Stelle gedreht.

PRODUKT ZUR TABELLE HINZUFÜGEN Die Produkte im Produktrad können in die Tabelle gezogen werden. Die Tabelle zeigt die entsprechenden Produkte dann an. Im Produktrad wird dieses dabei entfernt.

Wird ein Produkt dem Produktrad hinzugefügt, springt das Produktrad automatisch an diese Stelle.

Abbildung 86: Produktrad

C.2.4 *Inaktive Kategorien*

Auf der linken Seite befindet sich eine Anzeige der inaktiven Kategorien (siehe Abbildung 87). Dabei werden sämtliche Kategorien angezeigt. Die inaktiven werden dabei besonders hervorgehoben. Die aktiven Kategorien werden ausgegraut dargestellt. Inaktive Kategorien können mittels Drag & Drop in die Tabelle gezogen werden.

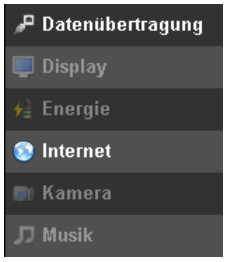


Abbildung 87: Inaktive Kategorien

C.2.5 *Kontrollbuttons*

Mit einem Druck auf den linken Button wird die Darstellung der Benutzeroberfläche auf den Anfangsstand zurückgesetzt. Mit dem rechten Button kann die Anzeige um 180 Grad gedreht werden.

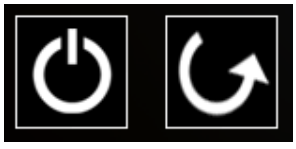


Abbildung 88: Kontrollbuttons

C.3 PROBLEME UND PROBLEMLÖSUNGEN

C.3.1 *Die Applikation reagiert nicht auf die Benutzereingaben*

Das Problem ist vermutlich, dass der Tracker (Softwaretreiber) nicht läuft. Dieser muss vor dem Start der Applikation gestartet werden.

C.3.2 *Die Benutzereingaben werden nur teilweise oder verzögert verarbeitet*

Überprüfen Sie beim Tracker die FPS-Anzeige (Anzahl Bilder pro Sekunde). Diese sollte konstant 60fps anzeigen. Ist dies nicht der Fall hilft teilweise das Aus- und Einstecken des Firewire-Kabels und ein Neustart des Trackers.

C.3.3 *Einige Elemente verschwinden nicht mehr von der Benutzeroberfläche*

Tritt dieses Problem auf, hilft ein Druck auf den Resetbutton. Nützt dies nichts, muss die Applikation neu gestartet werden.

C.3.4 *Der Druckpunkt wird von der Applikation ungenau interpretiert*

Dies hängt mit einer schlechten Kalibrierung des Trackers zusammen. Hier muss eine neue Kalibration durchgeführt werden.

D

VEREINBARUNG UND ERKLÄRUNG

D.1 VEREINBARUNG

D.1.1 *Gegenstand der Vereinbarung*

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit Interaktiver Multi-Touch-Sales-Tisch von Andreas Fischbacher und Marcel Lenz unter der Betreuung von Prof. Dr. Markus Stolze geregelt.

D.1.2 *Urheberrecht*

Die Urheberrechte stehen der Studentin / dem Student zu.

D.1.3 *Verwendung*

Die Ergebnisse der Arbeit dürfen sowohl von der Studentin / dem Student, von der HSR wie von to-fuse nach Abschluss der Arbeit verwendet und weiter entwickelt werden.

Rapperswil, den ....
Die Studentin/der Student

Rapperswil, den ....
Die Studentin/der Student

Rapperswil, den ....
Der Betreuer / die Betreuerin der Bachelorarbeit

Rapperswil, den ....
Der Studiengangleiter / die Studiengangleiterin

D.2 ERKLÄRUNG

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt sind oder mit dem Betreuer schriftlich vereinbart wurden,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Ort, Datum:

Andreas Fischbacher Marcel Lenz

E

MULTI-TOUCH-LISTE

In der nachfolgenden Liste werden einige Multi-Touch-Geräte aufgelistet.

- to-fuse Multi-Touch Plattform
  http://www.to-fuse.ch/#multitouch
- Microsoft Surface
  http://www.microsoft.com/surface/
- Apple iPhone
  http://www.apple.com/iphone/
- Multi-Touch von Jeff Han
  http://cs.nyu.edu/ jhan/ftirtouch/index.html
- Art+Com Floating Numbers
  http://www.artcom.de/index.php?option=com_acprojects&page=6&id=14&Itemid=144&details=0&lang=de
- iBar
  http://www.i-bar.ch/
- Reactable
  http://www.reactable.com/
- Mitsubischi Electronic Reserch Lab
  http://www.merl.com/projects/touchgestures/
- Fingerworks
  http://www.fingerworks.com/
- Touch Table
  http://www.touchtable.com/
- Lemur Jazzmutant
  http://www.jazzmutant.com/
- Dialog Table
  http://dialogtable.com/
- James Patten Audiopad
  http://www.jamespatten.com/audiopad/index.php
- SmartSkin von Jun Rekimoto
  http://www.sonycsl.co.jp/person/rekimoto/smartskin/

F

QUELLENANGABEN

Für die Grafiken in der Applikation wurden die folgenden Quellen verwendet:

- Hintergrund der Applikation
  http://lys036.deviantart.com/art/Circles-by-lys036-114216682
- Sämtliche verwendete Icons stammen von der Seite iconfinder.net
  http://www.iconfinder.net/
- Sämtliche Produktbilder stammen von der Seite inside-handy.de
  http://www.inside-handy.de/

Die in der Applikation verwendeten Produktdaten entstammen der Seite http://www.inside-handy.de/. Die Produktdaten wurden für einen besseren Demoeffekt teilweise modifiziert und entsprechen nicht zwingend den Originaldaten.

Das verwendete LaTeX-Template ist das classicthesis von André Miede.
http://www.kom.tu-darmstadt.de/miede/2miede/index.php?page=classicthesis

[1] Patrick Baudisch and Gerry Chu. Back-of-device interaction allows creating very small touch devices. 2009. http://patrickbaudisch.com/publications/2009-Baudisch-CHI09-BackOfDeviceInteractionAllowsCreatingVerySmallTouchDevices.pdf. (Zitiert auf der Seite 32.)

[2] Augusto Celentano and Andrea Minuto. Gestures, shapes and multitouch interaction. *IEEE Computer Society - 2008 19th International Conference on Database and Expert Systems Application*, pages 137–141, 2008. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4624705&isnumber=4624651. (Zitiert auf der Seite 31.)

[3] James E. Dion. *Retail Selling Ain't Brain Surgery, It's Twice As Hard*. Dionco Inc., 1. edition, 1996. (Zitiert auf den Seiten 32 and 33.)

[4] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. *Symposium on User Interface Software and Technology archive, Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, 2005. http://portal.acm.org/citation.cfm?id=1095054. (Zitiert auf der Seite 31.)

[5] Jan-Keno Janssen. Fass mich an! microsofts multitouch-tisch surface startet in deutschland. *c't 8/09*, 2009. http://www.heise.de/ct/Microsofts-Multitouch-Tisch-Surface–/artikel/135188. (Zitiert auf der Seite 31.)

[6] Craig Larman. *UML 2 und Patterns angewendet - Objektorientierte Softwareentwicklung*. Mitp-Verlag, 1. edition, 2005. (Zitiert auf den Seiten 7, 14, and 59.)

[7] Microsoft. Windows vista user experience touch guidelines. Website. http://msdn.microsoft.com/en-us/library/cc872774.aspx. (Zitiert auf den Seiten 45 and 46.)