Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet AOT
Prof. Dr.-Ing. habil. Dr. h.c. Sahin Albayrak

DAI-Labor
TU Berlin

Bachelorarbeit

IMPLEMENTIERUNG EINES 3D USER INTERFACES

Andreas Langenhagen

April 2012

Betreuer:
Prof. Dr.-Ing. habil. Dr. h.c. Sahin Albayrak
Mathias Wilhelm

Langenhagen
315720
Studiengang: Informatik B. Sc.
barn07@web.de

ERKLÄRUNG DER URHEBERSCHAFT

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Andreas Langenhagen Berlin, den 29.05.2012

ZUSAMMENFASSUNG

Die Steuerung einer intelligenten Heimumgebung sollte so ergonomisch und so einfach wie möglich funktionieren. Der Nutzer sollte bestenfalls ohne Einführung und mit einem kurzen Blick wichtige Informationen erhalten und Gerätezustände ändern können. Anstatt einer stark abstrahierten, wenig intuitiven Schnittstelle sollte dem Nutzer eine grafische Oberfläche gegeben werden, die er idealerweise schon kennt. Die visuelle Darstellung der Wohnung und ihrer Geräte als dreidimensionales Modell scheint eine Repräsentation zu sein, die diesen Anforderungen gerecht werden kann. Auf diesen Seiten wird der Ansatz einer dreidimensionalen grafischen Nutzerschnittstelle für intelligente Heimumgebungen, das *Smart Home Graphical User Interface*, kurz *Smart Home GUI*, beschrieben. Dabei wird auf die verwendeten Technologien und die Implementierung eingegangen und anschließend eine Studie zur Nutzbarkeit vorgestellt und durchgeführt. In dieser Studie wird die Smart Home GUI mit einem zweidimensionalen Pendant in Bezug auf Handhabung und Effizienz getestet. Die Studie zeigt, dass die dreidimensionale GUI für intelligente Heimumgebungen von ungeübten Nutzern deutlich schneller bedient werden kann, als eine zweidimensionale Nutzeroberfläche.

INHALT

1 MOTIVATION & ZIELSETZUNG

In einer intelligenten Heimumgebung (engl. Smart Home oder Smart Home Environment) geht es darum, möglichst vielen Geräten in einem Haushalt eine gemeinsame Schnittstelle für ihre Überwachung und Nutzung zu geben, somit größeren Wohnkomfort, mehr Sicherheit und höhere Energieeffizienz zu gewährleisten. Um die Heimumgebung so einfach wie möglich managen zu können, müssen die oft multimodalen Nutzerschnittstellen vieler Geräte möglichst einheitlich und intuitiv gestaltet werden; zum Verstehen der Nutzerschnittstelle soll es keiner langen Einarbeitungszeit bedürfen und möglichst viele Aspekte, wie der aktuelle Zustand eines Gerätes, sollten in kürzester Zeit observier- und änderbar sein. Betrachtet man Nutzerschnittstellen herkömmlicher Anwendungsprogramme wie man sie von Heimcomputern kennt, kann man feststellen, dass sich deren Verwendung im Kontext der intelligenten Heimumgebung als zu statisch in Hinblick auf Nutzerfreundlichkeit und Akzeptanz heraus stellen kann. Klassischen Nutzeroberflächen mangelt es wegen ihrer zweidimensionalen Beschaffenheit an der Fähigkeit, dreidimensionale Umgebungen auf dem Monitor einfach zu repräsentieren, ohne dabei hilfreiche Informationen über die Beschaffenheit der Wohnung und Gegenstände in ein wenig intuitives Format, wie z.B. Text, zu transcodieren oder ganz zu vernachlässigen. Auch kann eine Darstellung in einer zweidimensionalen Nutzeroberfläche missverständlich interpretiert werden. Betrachten wir beispielsweise eine zweidimensionale Draufsicht einer Wohnung: Zwar können Positionskoordinaten oder Markierungen auf gerenderten Grundrissen angezeigt werden, jedoch fehlt in der Betrachtung eben die dritte Dimension. Liegt nun ein interaktiver Gegenstand, etwa ein Mobiltelefon, auf einer Ablage genau über einer ebenso interaktiven Geschirrspülmaschine, müsste man, um das Telefon (oder die Spülmaschine) nicht zu übersehen, weitere textuelle oder bildhafte Informationen bereit stellen. Dies kann dem Grad der Einfachheit abträglich sein, sodass man gezwungen ist, einen genaueren Blick auf das Display zu werfen. In einem anderen denkbaren Szenario kann eine etwaige sequenzielle Auflistung mehrerer Gegenstände wiederum verwirrend wirken: Zwei Heizungen zum Beispiel, die auf dem Bildschirm in einer Liste angeordnet sind, könnten durch einen allzu schnellen Blick leicht verwechselt werden. Potentiell sind bei zweidimensionalen Schnittstellen für die Verwaltung vieler Geräte auch viele abstrakte Bezeichner, etwa in Textform, nötig, die der Anwender kennen muss, um sie zuzuordnen. Um derlei Probleme zu umgehen, beschäftigt sich die vorliegende Arbeit mit dem Ansatz, eine dreidimensionale Nutzeroberfläche in ein Smart Home zu integrieren, um so die Ergonomie des Systems zu steigern.

Ziel der Arbeit ist die Entwicklung und Evaluierung einer dreidimensionalen, multitouchfähigen und gegebenenfalls dynamischen Nutzerschnittstelle zur Darstellung und Manipulation der Aspekte einer realen Heimumgebung, um die Nutzerfreundlichkeit möglicher Heimanwendungen zu verbessern. Die Schnittstelle, genannt die Smart Home GUI, soll GUI-Elemente und eine dreidimensionale Repräsentation eines Heimumgebung mitsamt seiner Objekte umfassen. Als mögliche Eingabegeräte werden multitouchfähige Geräte beachtet, sodass direkte graphische Eingabemethoden anstelle von Maus und Tastatur als Eingabemedien verwendet und getestet werden können. Zudem sollen

mithilfe von herkömmlichen GUI- Komponenten, sogenannten Widgets, auch einfache Touchscreens unterstützt werden. Diese Widgets können auch Nutzern, denen Fingergesten als Eingabeschemata fremd sind, eine vertraute Schnittstelle bieten, welche man von herkömmlichen Betriebssystemen kennt. Auch Ansätze von Endbenutzer-Entwicklung (engl. End-User Development) (Langenhagen 2010) sollen in der Anwendung wieder zu finden sein. Etwa soll der Nutzer die grafischen Darstellungen von Geräten und der Wohnung verändern können und gegebenenfalls auch selbst 3D-Modelle von Haushaltsgeräten erstellen. Auch sollen ganze visuelle Designs der Nutzeroberfläche einfach austauschbar sein, um idealerweise verschiedenen Displayformaten gerecht zu werden. Hierfür soll eine starke Trennung zwischen Programmcode und Design umgesetzt werden. Um eine solche Trennung möglichst konsistent und strikt zu bewahren, soll reichlicher Gebrauch von Entwurfsmustern (engl. Design Patterns) (Gamma, et al. 1994) wie der abstrakten Fabrik und dem Singleton-Entwurfsmuster gemacht werden.

Im Kapitel 2 werden zunächst verwandte Arbeiten mit intelligenten Heimumgebungen vorgestellt, dann eine Auswahl an Arbeiten zum Thema Multitouch-Gesten besprochen. Dies soll einen Einblick in den Stand der Technik geben und als Anregung für Ideen für die Steuerung einer Multitouch-Anwendung dienen. Es gibt eine Reihe an Software, die das Schreiben von Multitouch-Anwendungen vereinfacht. In Kapitel 3 werden dem Projekt potenziell zuträgliche Multitouch-Frameworks, 3D-Modellierer und ein Programm zur Verwaltung einer intelligenten Heimumgebung vorgestellt. Auf Grundlage dieser Informationen wird sodann eine Entscheidung für die Nutzung externer Software gefällt. In Kapitel 4 wird der allgemeine Lösungsansatz für die Implementierung der Smart Home GUI vorgestellt und es werden die allgemeinen Designrichtlinien festgehalten. Kapitel 5 befasst sich im Detail mit der konkreten Implementierung des Projekts und stellt die Module der Software näher vor. In Kapitel 6 wird die fertige Nutzerschnittstelle kurz vorgestellt. Zur Auswertung der Arbeit wird in Kapitel 7 eine Evaluierung bezüglich der Nutzbarkeit der 3D-Schnittstelle durchgeführt. In dieser Evaluierung wird die 3D-Schnittstelle mit einer entsprechenden 2D-Nutzeroberfläche gegenübergestellt wird. Hierfür wird eine Evaluationsstudie durchgeführt, die die Geschwindigkeit, mit der Geräte mithilfe einer 2D-Oberfläche und einer 3D-Oberfläche gesteuert werden, gemessen. Als Eingabegeräte werden jeweils Tablet-Computer eingesetzt. In Kapitel 8 wird eine kurze Zusammenfassung und Kritik zu den Resultaten der Arbeit und der Studie präsentiert. Schlussendlich gibt Kapitel 9 einen Ausblick auf mögliche, künftige Arbeiten in Bezug auf Multitouch-Interaktion mit dreidimensionalen Objekten und die Smart Home GUI im Speziellen.

2 VERWANDTE ARBEITEN

In diesem Kapitel werden mit der Arbeit vergleichbare Lösungen vorgestellt. Dabei werden ab Abschnitt 2.1 zuerst drei Applikationen für intelligente Heimumgebungen, beziehungsweise ihre Nutzerschnittstellen vorgestellt. Die gezeigten Schnittstellen sollen einen Eindruck vom Stand der Technik verschaffen und als Inspiration für die weitere Arbeit dienen. Ab Abschnitt 2.2 werden Arbeiten mit speziellem Bezug auf die Analyse und Anwendung von Multitouch-Interaktion in Augenschein genommen. Da dreidimensionale Multitouch-Nutzerschnittstellen im Alltagsleben selten vorkommen, gibt es bis zum heutigen Zeitpunkt keine allgemein gültigen Standards für die Multitouch-Steuerung mit 3D-Schnittstellen. Das Aufzeigen der hier angesprochenen Arbeiten soll einen Aufschluss über moderne Steuerungstechniken geben und das Bewusstsein für verschiedene Möglichkeiten der Multitouch-Interaktion wecken.

2.1 Existierende Smart Home Interfaces

2.1.1 CRISTAL

Seifried et al. (Seifried, et al. 2009) entwickelten eine Nutzerschnittstelle für intelligente Heimumgebungen, vornehmlich zur Wiedergabe von Medien und zur gemeinsamen Nutzung von im Haushalt vorkommenden Geräten. Mithilfe eines Multitouch-Tisches, der eine Draufsicht des Raumes zeigte, konnten hier Nutzer mit dem Smart Home interagieren. Die Draufsicht wurde mithilfe einer Kamera generiert, welche den gesamten Raum von der Zimmerdecke aus filmte und das Multitouch-Display so mit Livedaten versorgte. Statt einer gewöhnlichen GUI dienten die videogerenderten Darstellungen der Haushaltsgegenstände selbst als Interaktionsflächen. So konnten Lampen durch Berühren und darauf folgende Wisch-Gesten gedimmt werden oder Videos aus dem Videoregal zum Schauen ausgewählt werden. Der Einsatz einer Kamera hatte den Vorteil, dass künstlich Eingabefeedbacks zum Teil hinfällig wurden; wenn man die Lampe dimmte, veränderte sich das angezeigte Videobild automatisch. Ein weiterer Vorteil war, dass die künstliche Generierung einer grafischen Repräsentation des Raumes innerhalb der Software ebenso hinfällig wurde, da das Bild des Raumes selbst seine Repräsentation war. Jedoch, wenn die Visualisierung des Zimmers durch ein Kamerabild entsteht, muss man immer die Sorge tragen, dass die Räume ausreichend beleuchtet sind, um ein gutes Bild zu gewährleisten. Auch die statische Sicht auf den Raum – es sei denn, man stattet ihn, anders als im CRISTAL-Schaubeispiel, mit mehreren Kameras aus – kann Potentiale in Sachen Ergonomie möglicherweise nicht ausreizen. Eine virtuelle 3D-Darstellung des Raumes ist möglicherweise besser geeignet, einen intelligenten Haushalt darzustellen.

2.1.2 Virtuelle 3D “Smart Home” Nutzerschnittstelle

Borodulkin et al. (Borodulkin, Ruser und Tränkler 2002) stellten eine 3D-Anwendung vor, welche die Steuerung eines Smart Home ermöglicht. Diese Anwendung war serverbasiert und nutzte die Java3D-Bibliotheken. Beim Entwurf der Software wurde sowohl großer Wert auf Nutzerfreundlichkeit als auch auf Anwendung der Prinzipien moderner

Objektorientierung gelegt. So wurden manipulierbare Gegenstände der Heimumgebung in Etagen, und Etagen in Räume unterteilt. Auch wurde die Oberfläche für die Nutzung mit einem Touchscreen ausgelegt. Da im Jahre 2002 die Verbreitung multitouchfähiger Geräte jedoch noch recht gering war, war das Interface der Anwendung nur zur Interpretation einfacher Fingergesten in der Lage. Heutzutage lassen sich derlei Anwendungen mit Multitouch-Eingabemethoden realisieren, was für eine höhere Akzeptanz seitens des Anwenders sorgen kann. In der vorliegenden Arbeit wird, anders als bei Borodulkin et al., der Interaktion durch Multitouch eine hohe Bedeutung beigemessen.

2.1.3 Gira KNX/EIB System

Das kommerzielle Gira KNX/EIB System der Gira GmbH (Gira GmbH & Co. KG 2012) ermöglicht die Automatisierung und Steuerung einer intelligenten Heimumgebung. Als potenzielle Eingabegeräte kommen die firmeneigenen Touchscreen-Computer "Gira Control Clients" sowie Apple iPhones, Apple iPads in Frage. Eine zentralisierte Steuerung von häuslichen Geräten wird via Home Server realisiert. Die Nutzerschnittstelle ist beim KNX/EIB System zweidimensional, bietet aber, sofern verfügbar, Kamerabilder zur Ansicht der Wohnung. Die Gira GmbH wirbt zwar mit einem hohen Nutzerkomfort, doch treffen die in Kapitel 1 angesprochenen Schwachpunkte zweidimensionaler Darstellungen, wie beispielsweise die hoch abstrakte textuelle Darstellung von Geräten, ebenso hier zu.

2.1.4 Zusammenfassung

Tabelle 2.1 listet die drei vorgestellten Nutzerschnittstellen für intelligente Heimumgebungen noch ein Mal in Kürze auf und vergleicht ihre Darstellungseigenschaften in Tabellenform. Grüne Hintergrundfarbe zeigt Überschneidungen mit den Ideen der in der vorliegenden Arbeit präsentierten Lösung, roter Hintergrund zeigt Unterschiede.

	CRISTAL	**3D Virtual “Smart Home” UI**	**Gira KNX/EIB**
Kamera dynamisch/statisch	Statisch	Dynamisch	Statisch
Multitouch	Ja	Nein	Nein
Repräsentationsart	3D	3D	2D

Tabelle 2.1: Die vorgestellten Nutzerschnittstellen in tabellarischer Form entsprechend einiger Eigenschaften.

Zusammenfassend kann man sagen, dass keines der vorgestellten Programme erstens eine dreidimensionale Repräsentation für eine intelligente Heimumgebung wählt und zweitens zeitgleich dynamische Kameraführung auf einem – drittens – Multitouch-Gerät bietet. Diese drei Aspekte werden in der vorliegenden Arbeit im Gegenzug zu den vorgestellten verwandten Arbeiten vereint.

2.2 Arbeiten zum Thema 3D und Multitouch

2.2.1 DOF Separation in 3D Manipulation-Tasks mit Multitouch Displays

Im Bereich der Multitouch-Interaktion gibt es mittlerweile eine Reihe an Arbeiten, jedoch beschränken sich diese zumeist auf den zweidimensionalen Raum; die Rotation, Translation und Skalierung (kurz RST) von Bildern ist ein häufig gezeigtes Szenario. Weniger erkundet ist bislang die Multitouch-Interaktion mit dreidimensionalen Objekten, besonders im Kontext intelligenter Heimumgebungen. In diesem Kapitel werden kurz einige Arbeiten vorgestellt, die sich mit neuartigen Interaktionstechniken für dreidimensionale Objekte auseinander setzen. Martinet et al. (Casiez, Martinet und Grisoni 2010) vertraten die These, dass auf Multitouch-Geräten die gemischte, gleichzeitige Steuerung von Translation und Rotation bei 3D-Objekten zu Verwirrungen und geringer Effizienz führt. Ihrer Meinung nach ist eine Trennung der Freiheitsgrade (engl. Degrees Of Freedom, kurz DOF) akkurater und effizienter als eine Steuerung, die die Veränderung von Translation und Rotation in einer Gesten-Technik kombiniert. Nach dieser These erarbeiteten sie eine eigene Technik zur Multitouch-Manipulation von Objekten im Dreidimensionalen, genannt DS3, und verglichen diese im Anschluss mit zwei weiteren Techniken. Durch das Experiment wurde ihre Vermutung bestätigt, dass sich eine Trennung verschiedenartiger Freiheitsgrade positiv auf die Nutzerfreundlichkeit auswirken kann. Die Forscher betrachteten jedoch keinerlei Skalierungstransformation in ihrer Arbeit. Ein Video zur Verwendung ihrer DS3-Technik veröffentlichte man unter (Martinet 2011).

2.2.2 RST Multitouch Experiment

Knoedel und Hachet (Knoedel und Hachet 2011) veröffentlichten einen Artikel, in welchem sie eine spezifische 2D- sowie eine spezifische 3D-RST-Technik vorstellten und evaluierten. In einem Video präsentierten sie ein Experiment zu dieser Technik und zeigen zudem einen Anwendungsfall, in welchem eine Testperson Möbelstücke in einer Wohnung positionieren musste. Auch wenn diese gezeigte Positionierung von Haushaltsgegenständen nicht im Zusammenhang mit einem Ambient Assisted Living System steht, ist es doch eine Aufgabe, die unter Umständen mit einer Nutzerschnittstelle einer intelligenten Heimumgebung durchgeführt werden muss, um beispielsweise Positionen von Gegenständen zu justieren oder gar per Gesten-Steuerung ganze dreidimensionale Modelle intelligenter Heimumgebungen zu designen.

2.2.3 Multitouch-Interaktion mit 3D-Objekten

Breier beschäftigte sich in seiner Master-Thesis (Breier 2010) mit dem Umgang mit 3D-Objekten auf einem Multitouch-Tisch. Da die Multitouch-Steuerung mit 3D-Objekten noch nicht ausgeprägt genug ist, um Standards hervorzubringen, war Breiers Ziel, Möglichkeiten zur Gesten-Steuerung von 3D-Objekten zu finden, mit denen Objekte im dreidimensionalen Raum positioniert und rotiert werden können. Hierfür recherchierte, entwickelte und bewertete er neue Konzepte für die Interaktion mit dreidimensionalen Objekten. Neben Techniken auf Basis von zweidimensionalen Gesten arbeitete Breier auch mit einem separaten Beschleunigungsmesser als indirektem Eingabemedium. Die erarbeiteten Konzepte wurden hauptsächlich für einen Multitouch-Tisch entwickelt und er-

laubten gegebenenfalls die gleichzeitige, kollaborative Interaktion mehrerer Nutzer mit dem Eingabemedium.

2.2.4 Zusammenfassung

Es gibt eine Reihe denkbarer Steuerungstechniken, die man auf Multitouch-Geräten anwenden kann. Ebenso viele Arbeiten lassen sich dazu finden, doch keine bislang allgemein akzeptierten Standards. Es existieren verschiedene RST-Techniken, und die Separierung der Freiheitsgrade bei RST-Techniken kann sich positiv auf die Präzision der Steuerung und damit auf die Nutzerfreundlichkeit auswirken. Nichtsdestotrotz legt die Nichtexistenz allgemein akzeptierter Standards nahe, dass eine konkrete Steuerung einer Nutzerschnittstelle hauptsächlich vom Sachverhalt abhängt, den die Nutzerschnittstelle verkörpert. In der vorliegenden Arbeit wird die Trennung der Freiheitsgrade weitgehend angestrebt, die ausgeübten Gesten werden jedoch hauptsächlich von der jeweiligen Situation im Programm abhängig gemacht.

3 VERWENDETE TECHNOLOGIEN

Bezüglich der Steuerung von Heimgeräten sowie für die Erstellung reichhaltiger grafischer Nutzeroberflächen gibt es bereits einige Technologien, die für die Nutzung in dieser Arbeit in Frage kommen. Im Folgenden werden einige dieser potenziell nutzbaren Technologien verglichen und schließlich die in der Arbeit verwendeten Schlüsseltechnologien näher vorgestellt. Zunächst wird in 3.1 die Multi-Access Service Platform (kurz MASP) vorgestellt, welche eine Java-Schnittstelle, genannt Interface, zur Steuerung von intelligenten Heimumgebungen bietet. In Abschnitt 3.2 werden einige Multi-Touch Frameworks miteinander verglichen und schließlich das für die Arbeit erwählte MT4j Framework näher beschrieben. Gleiches wird im Abschnitt 3.3 für 3D-Modellierungsprogramme geschehen. Diese müssen benutzt werden, um Haushaltsgeräten und dem Haushalt selbst eine adäquate visuelle Repräsentation zu ermöglichen.

3.1 MASP

Die Multi-Access Service Platform (MASP) (Roscher, et al. 2011) stellt eine Schnittstelle zur Kommunikation mit steuerbaren Heimgeräten dar und kapselt dabei verschiedene Steuerungstechnologien und Protokolle vom Klienten ab. Auch kümmert sie sich um die Datenhaltung über den Zustand von Geräten. Dieses Verhalten als Adapter ermöglicht eine Steuerung gleichartiger, doch verschiedener Geräte (z.B. Lampen verschiedener Hersteller) mithilfe der gleichen Schnittstelle, sodass sich der Klient nicht mit unterliegenden Details beschäftigen muss. Weiterhin liefert die Plattform die Daten-Repräsentation einer intelligenten Heimumgebung, welches aus mehreren Räumen und Geräten bestehen kann.

Die MASP ermöglicht im Zusammenhang dieser Arbeit die eigentliche Steuerung einer intelligenten Heimumgebung, was den Implementierungsaufwand auf die Erzeugung von Widgets und die eigentliche Anwendungsprogrammierung reduziert. Entweder kann man die MASP direkt mithilfe eines Interface benutzen, oder aber über eine JSON[1]-Schnittstelle, was den Klienten weitgehend unabhängig von der Implementierungssprache des MASP macht. Die folgende Abbildung 3.1 verdeutlicht noch einmal die Rolle der MASP als Adapter, welcher die Schnittstellen verschiedener Heimgeräte abstrahiert und bündelt.

[1] http://www.json.org/. Abgerufen am 28.05.2012

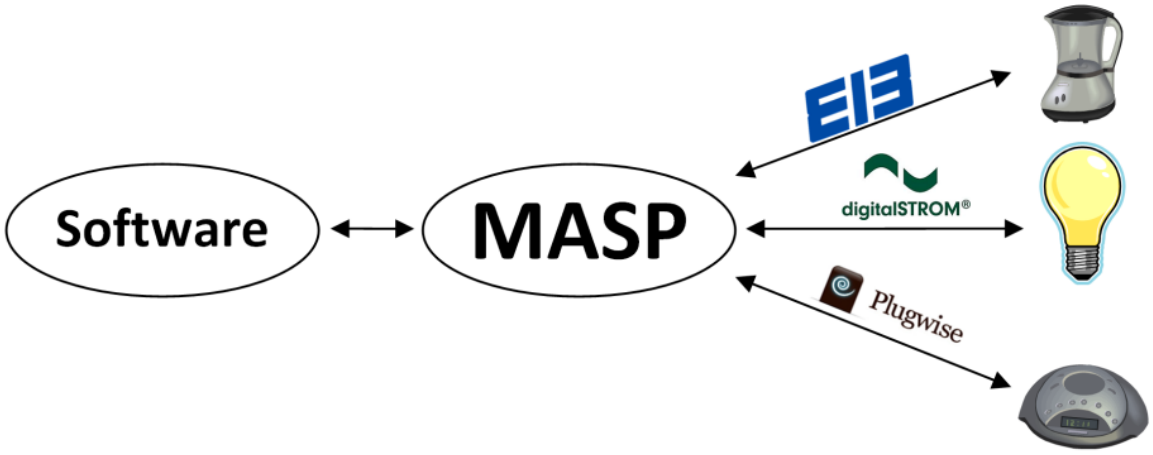

Abbildung 3.1: Die Adapterfunktion der MASP ermöglicht, verschiedene Steuerungsmechanismen zu kapseln und über eine einheitliche Schnittstelle zu nutzen.

3.2 Multitouch-Frameworks

Es gibt eine Reihe von Multitouch-Frameworks für verschiedene Plattformen und Programmiersprachen. Die Wahl des Frameworks bestimmt die Wahl der anzuwendenden Programmiersprache und umgekehrt. Auch bestimmt diese Wahl maßgeblich den Entwicklungsaufwand der Applikation. Obgleich es nicht zwingend notwendig ist, ist es aus Bequemlichkeitsgründen wünschenswert, dass das zu verwendende System eine Schnittstelle gewährleistet, sodass man diese einfach mit der Schnittstelle der MASP kombinieren kann. Andernfalls kann man die JSON-API der MASP mit jeder konventionellen Programmiersprache ansprechen, was jedoch wahrscheinlich den Programmieraufwand steigern würde. Auch sollte das zu nutzende Multitouch-Framework gut dokumentiert und in einer stabilen Version verfügbar sein. Zudem sollte es eine möglichst breite Nutzerbasis bieten, um schnellstmöglich Unterstützung im Falle von Schwierigkeiten im Umgang mit der Software zu erhalten. Als potenzielle Programmiersprachen kamen bei der näheren Auswahl Java, C++ und C# in Frage.

3.2.1 Auswahl eines Multitouch-Frameworks

Drei Software-Bibliotheken, libTisch (Echtler 2010), MT4j (Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO 2010) und Breezemultitouch (Mindstorm Limited 2010) wurden näher betrachtet, um daraus die adäquateste für die vorliegende Arbeit auszuwählen (Tabelle 3.1). Als Kriterien wurden Programmiersprache, Entwicklungsstand/Stabilität, die Verfügbarkeit des Quelltexts und die Lizenz ausgewählt. Weiterhin wurden die Fragen, ob Gesten-Erkennung schon in die Frameworks eingebaut ist, und ob diese Gesten-Erkennungen erweiterbar sind, beachtet. Ein letztes Auswahlkriterium war die Verfügbarkeit von Visualisierungsmöglichkeiten, also ob Widgets für die Nutzung mit den Frameworks existieren. Das erstgenannte Kriterium, die Programmiersprache, sollte möglichst mit Java erfüllt sein, da die MASP selbst eine Java-Schnittstelle zur Verfügung stellt. Andernfalls müsste von der JSON-Schnittstelle der MASP Gebrauch gemacht werden. Das Spektrum der Zielplattformen des zu verwendenden Frameworks sollte möglichst groß sein, um die Implementierung der vorliegenden Arbeit

selbst möglichst plattformunabhängig zu machen. Der Aktivitätsstatus des zu nutzenden Frameworks sollte idealerweise aktiv sein, sodass im Falle von Problemen mit Unterstützung seitens der System-Entwickler zu rechnen sein kann. Natürlich sollte das Framework an sich in einem stabilen Zustand sein. Der Quelltext des Projekts sollte verfügbar sein, um diesen im Falle von Problemen zu Rate zu ziehen und gegebenenfalls manuell Änderungen daran vorzunehmen. Hierfür wird auch eine offene Lizenzierung des Frameworks beansprucht. Es sollten neben der Multitouch-Funktionalität schon grundlegende Gesten-Erkennungen vom Framework implementiert sein, auch sollten die Gesten-Erkennungen vom Nutzer des Frameworks erweiterbar sein. Zuletzt sollte das Framework Unterstützung für bestehende visuelle Komponenten bieten, oder eigene Komponenten Widgets zur Verfügung stellen. In Tabelle 3.1 werden die drei betrachteten Softwarebibliotheken auf Grundlage der gewählten Kriterien verglichen. Grüne Hintergrundfarbe zeigt als positiv gewertete Eigenschaften, Rote Hintergrundfarbe markiert weniger wünschenswerte Ausprägungen.

Kriterium	libTisch	MT4j	Breeze-multitouch
Programmiersprache	C++	Java	C#
Plattform	Windows, Linux, Linux	Windows, Linux, Mac OS	Windows 7
Status / Stabilität	aktiv/stabil	aktiv/stabil	inaktiv/stabil
Quelltext Verfügbar	Ja	ja	ja
Lizenz	LGPL	GPLv3	LGPL
Gesten-Erkennung eingebaut	Ja	ja	Ja
Gesten erweiterbar	Ja	ja	ja
Visualisierungsmöglichkeiten	Ja	ja	ja

Tabelle 3.1: Verschiedene Multitouch-Frameworks und Kriterien an diese.

Das für diese Arbeit adäquateste System, Multitouch for Java, erfüllt einzig alle auferlegten Kriterien. Diese Software wird im Folgenden kurz vorgestellt.

3.2.2 Multitouch for Java

Multitouch for Java, kurz MT4j (Laufs, Ruff und Zibuschka 2010) (Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO 2010), ist ein Open Source Framework für Java, entwickelt zur einfachen Erstellung von grafisch aufwändigen Applikationen. Besonderer Fokus liegt hier auf der Unterstützung diverser Multitouch-Eingabegeräte. Mithilfe von MT4j, welches auf Processing[2] aufbaut, lassen sich zwei- und dreidimensionale Szenen einfach in der Programmiersprache Java aufsetzen. Auch die Verarbeitung von Eingabesignalen ist auf einfache Art und Weise möglich. Die MT4j-Bibliothek stellt verschiedene Möglichkeiten zum Design und zur Ereignisbehandlung bereit, auch existieren

[2] http://processing.org. Abgerufen am 24.05.2012

viele geometrische Objekte und einige GUI-Komponenten. Dreidimensionale Modelle aus Dateien können ebenso geladen und dargestellt werden. Reaktive GUI-Komponenten, die sich besonders für die Steuerung einer Multitouch-Oberfläche eignen und noch nicht vom Framework zur Verfügung gestellt werden, müssten aus den bestehenden Elementen aggregiert werden, doch bietet die Bibliothek wie erwähnt wie erwähnt von Haus aus schon einige Komponenten an. Abbildung 3.2 beschreibt die Schichtenarchitektur von MT4j, wobei Dateneingaben von der untersten Schicht bis zur obersten gesendet werden. Nur die zwei höheren Schichten werden für die Implementierung des Projekts verwendet oder gar erweitert; im „Input Processing Layer“ findet man Eingabeprozessoren, welche Eingabesignale als Gesten interpretieren können, im „Presentation Layer“ sämtliche Bestandteile einer Nutzerschnittstelle und die Klassen, die für das Rendern der Szenen verantwortlich sind.

3.3 3D-Modellierer

Für Nutzerschnittstellen, welcher komplexe dreidimensionale Modelle darstellen sollen, ist es wichtig, 3D-Modelle auf einfachem Wege modellieren zu können. Hierfür gibt es eine Reihe von Software. In diesem Abschnitt werden drei kostenlose Modellierungsprogramme – Blender, Google SketchUp und Wings3D – miteinander verglichen, um eine Entscheidungsgrundlage für die Nutzung eines der Programme zu haben. Zu berücksichtigende Aspekte sind die unterstützten Grafikformate, Texturierungsfunktion, einfache Handhabung bzw. Einarbeitungszeit. Generell ist ein einfach zu bedienendes Programm mit Texturierungsfunktion und Unterstützung offener Dateiformate, wie Wavefront .obj[3] oder Autodesk .3ds[4], gewünscht. In Tabelle 3.2 werden die drei betrachteten Programme an den genannten Aspekten gegenübergestellt.

Blender (Blender Foundation 2012) ist ein verbreitetes quelloffenes Modellierungsprogramm mit einer immensen Mannigfaltigkeit an Funktionen, unter Anderem mit der Fähigkeit zur Modellierung und Texturierung von dreidimensionalen Objekten. Ein Nachteil von Blender, der sich aus den vielen Funktionen ergibt, ist eine wenig intuitive Nutzerschnittstelle mit hohem Einarbeitungsaufwand, viele Kommandos müssen über einen komplexen Menü-Baum aufgerufen oder per Tastenkombination getätigt werden.

Google SketchUp (Google Inc. 2012) ist eine dual lizenzierte Software, die in einer kostenlosen sowie kostenpflichtigen Pro-Version existiert. Die Software erlaubt Modellierung und Texturierung und ist auf schnelle und vornehmlich einfache Kreationen ausgelegt. Die Möglichkeit, 3D-Modelle in einem offenen Format wie dem .obj-Format zu speichern, existiert nicht. Lediglich bei der kostenpflichtigen Pro-Version sind Exporte in gemeinhin genutzte Formate, wie dem.3ds-Format möglich.

Wings 3D (Wings3D 2010) ist eine Quelloffene Software, die sich besonders durch ihre vergleichsweise einfache Benutzeroberfläche und ihr geringes Volumen für einfache Modellierungs- und Texturierungsaufgaben eignet. Das Programm wurde in der Sprache

[3] http://en.wikipedia.org/wiki/Wavefront_.obj_file. Abgerufen am 21.05.2012

[4] http://en.wikipedia.org/wiki/.3ds. Abgerufen am 21.05.2012

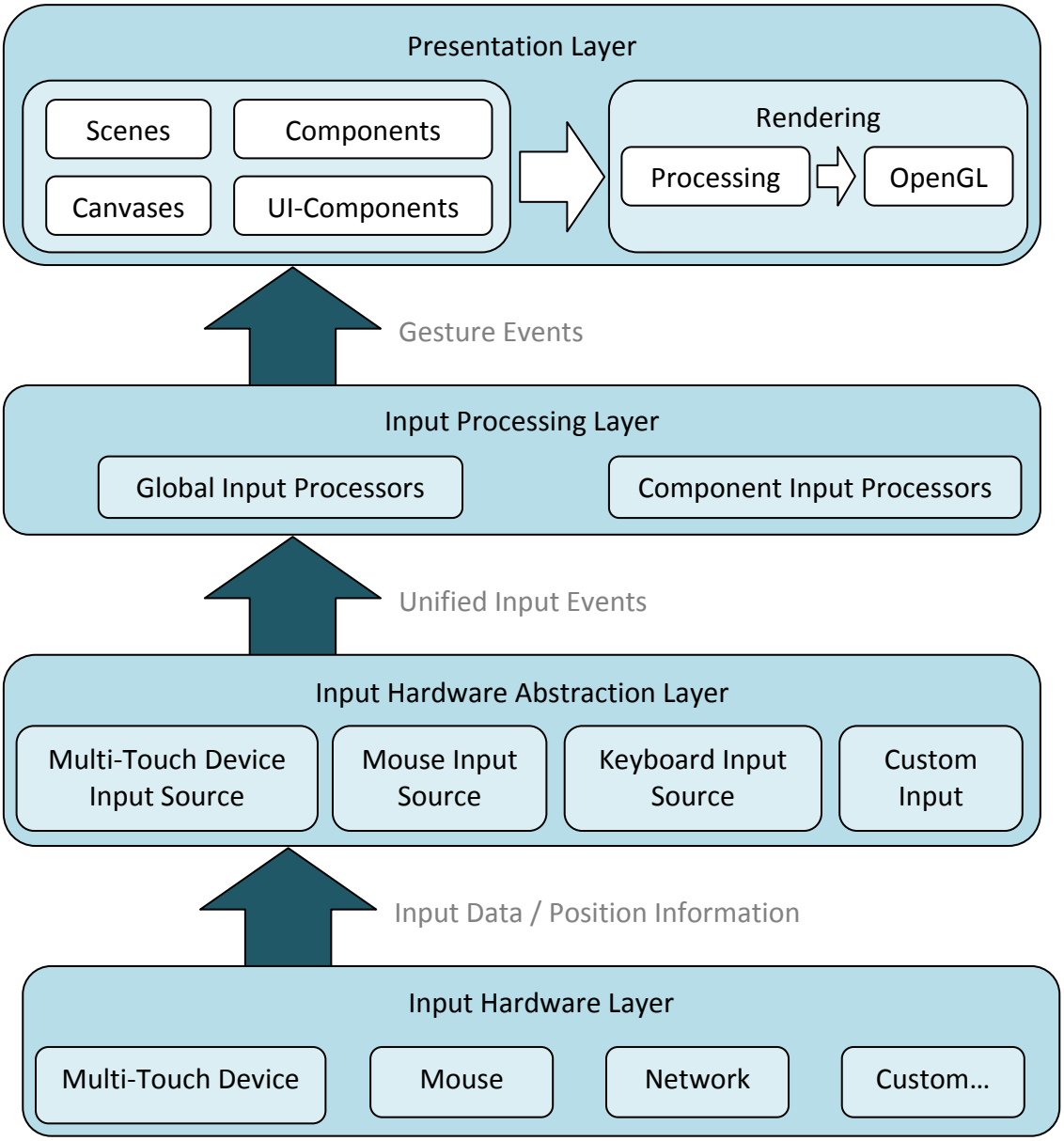

Abbildung 3.2: Die Schichtenartige Aufteilung der MT4j Architektur (Grafik nach (Fraunhofer IAO 2011)).

Kriterium	Blender	Google SketchUp	Wings 3D
Texturierung	Ja	Ja	Ja
Handhabung	Komplex	Einfach	Einfach
Unterstützt .obj	Ja	Nein	Ja
Unterstützt .3ds	Ja	(nur Pro-Version)	Ja

Tabelle 3.2: Modellierungssoftware bewertet nach Kriterien zur Auswahl des geeignetsten Programms.

Erlang[5] geschrieben und ist für alle gängigen Desktop-Betriebssysteme erhältlich. Da sich der Funktionsumfang von Wings 3D auf die Modellierung und Texturierung beschränkt, bietet das Programm eine einfache Nutzeroberfläche zur Steuerung, welche hauptsächlich mithilfe von Kontextmenüs nutzen lässt. Auf diese Art und Weise wird die Einarbeitungszeit eines Nutzers verkürzt. Wings 3D unterstützt eine Reihe offener Formate, darunter auch Wavefront .obj und Autodesk .3ds.

3.3.1 Wings 3D

Da Wings 3D einfache Bedienbarkeit und Unterstützung wichtiger Dateiformate verspricht, wird dieses Programm im Laufe des Projekts für die Erstellung von dreidimensionalen Objekten genutzt. Im Zuge der Arbeit dient Wings 3D als Modellierungstool für die Erstellung von 3D-Objekten, was eine zukünftige Verwendung anderer Modellierungstools jedoch nicht ausschließen soll. Die Objekte selbst werden im offenen Wavefront .obj – Format gespeichert, welches weite Verbreitung in der Computergrafik findet und auch von MT4j (3.2) akzeptiert wird. Neben diesen .obj-Dateien, welche die Geometrie eines Modells speichern, werden auch Wavefront .mtl – Dateien (Material Template Libraries) erzeugt, in denen Oberflächen- & Beleuchtungseigenschaften für Modelle enthalten sind. Texturen werden mithilfe von Dateien gängiger Grafikformate eingebunden. Abbildung 3.3 verdeutlicht die Zusammensetzung eines kompletten 3D-Objekts mittels Wavefront .obj noch einmal.

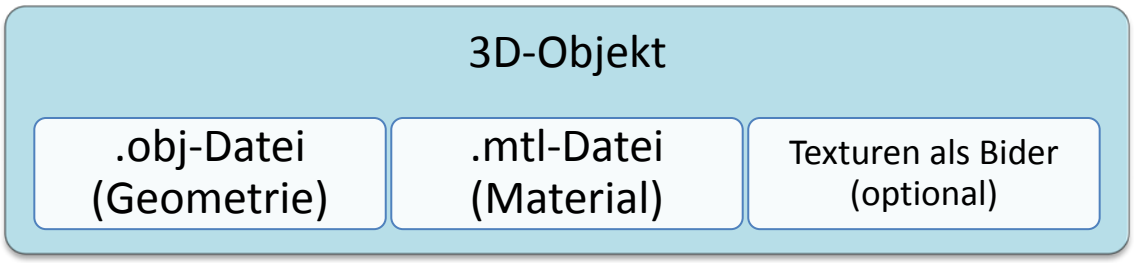

Abbildung 3.3: Repräsentation eines 3D-Modells mittels .obj, .mtl und Bilddatei.

[5] http://www.erlang.org. Abgerufen am 24.05.2012

4 LÖSUNGSANSATZ

Dieser Abschnitt beschreibt die generellen Designideen hinter der Arbeit und beleuchtet, auf welche Aspekte bei der Erstellung der Nutzerschnittstelle besonderer Wert gelegt wird. Zuerst wird im Abschnitt 4.1 die erdachte Designphilosophie in ihrer Gesamtheit erklärt und begründet. In 4.2 werden die beiden konzeptuellen Hauptbestandteile des Programms, ein mehr oder weniger generisches GUI Framework und schließlich die Smart Home GUI selbst, erläutert.

4.1 Architekturphilosophie

Da die Implementierung geschieht in der objektorientierten Programmiersprache Java und die Software wird extensiven Gebrauch von Entwurfsmustern machen, um so eine möglichst modulare und wartbare, doch zugleich dynamische und verständliche Architektur hervorzubringen. Das Ziel der Arbeit ist eine nutzerfreundliche, gegebenenfalls dynamische Nutzeroberfläche, also die Oberfläche und die 3D-Umgebung selbst von einem Entwickler leicht verändert bzw. angepasst werden können. Auch Ansätze von Endbenutzer-Entwicklung (Langenhagen 2010) sollen in der Applikation vorkommen. So soll dem Nutzer beispielsweise die Möglichkeit geboten werden, Designs von Gegenständen in der Heimumgebung selbst fest zu legen und gar eigene Designs zu entwerfen. Auch sollte aufgrund der Wunschidee, dass die entstehende Software auf verschiedenen Systemen mit verschiedenen Displayformaten ästhetisch ansprechend sein soll, die Anpassung des visuellen Designs so einfach wie möglich gemacht werden. Zu einem spezifischen Design gehören Schriftarten, Farbschemata, visuelle Beschaffenheit und Reaktionsverhalten von Widgets. Um eine derartige Anpassung zu leicht bewerkstelligen, muss die Trennung von Programmcode und Design so stark wie möglich sein, also die Verknüpfung von konkreten Ressourcen und der Software unterbunden werden. Dies kann erreicht werden, indem Module entwickelt werden, die die Ressourcenverwaltung vom restlichen Quelltext kapseln und den Entwickler der konkreten GUI vor der expliziten Auseinandersetzung mit einem spezifischen Design bewahren. Auf diese Art und Weise kann zudem agil auf externe Anforderungen bezüglich des Aussehens reagiert werden, ohne die Anwendung selbst neu zu schreiben. Eine solche Ressourcenverwaltung, die wegen der Gewährleistung auf Konsistenz zudem zentral geschehen sollte, legt die Anwendung von Entwurfsmustern wie der abstrakten Fabrik und des Singleton-Entwurfsmuster nahe. Da Ressourcenverwaltung und ihre Kapselung besonders bei grafisch aufwändigen Applikation eine wichtige Rolle spielt, sollte dieser bei der Implementierung der Arbeit besonderes Augenmerk gewidmet werden. Um generell für einen hohen Grad an Abstraktion und Kapselung zu sorgen, wird im ganzen Projekt von umfassendem Einsatz von Java-Interfaces und abstrakten Klassen Gebrauch gemacht. Dessen ungeachtet werden Ausnahmefälle, also Klassen ohne abstrakte Superschnittstelle, zum Zwecke der Komplexitätsreduktion ebenfalls existieren. Diese Klassen werden jedoch soweit nötig vollständig ableitbar sein, sodass vom Konzept der Polymorphie auch in solchen Fällen profitiert werden kann. Instanzen konkreter Objekte sollen mit Verwaltungs-Klassen, sogenannten Fabriken erzeugt werden. Der dadurch aufkommende Mehr-

aufwand bei der anfänglichen Implementierung kann im Folgenden durch schnelle Kreation von Nutzerschnittstellen wieder gut gemacht werden. Schlussendlich sollen die angewendeten Entwurfsmuster in ihrer Art und Weise möglichst oft verwendet werden, sodass verschiedene Subsysteme bis auf wenige Schlüsselaspekte identische Strukturen aufweisen können. Dies hat den Zweck, einem Entwickler schnelle Einarbeitung zu ermöglichen und die Intuition für das System zu erleichtern.

4.2 GUI Framework und Smart Home GUI

Vorangegangene Gedanken führen zu der Betrachtungsweise, die Software grob in zwei Teile zu zerlegen: Zum einen in ein Rahmenwerk, welches die Erstellung der Smart Home GUI so einfach wie möglich macht, zum anderen in die Smart Home GUI selbst. Im Folgenden wird also nunmehr zwischen dem GUI Framework und der Smart Home GUI unterschieden. Ersteres beinhaltet jeglichen Quelltext, welcher sich für eine beliebige (domänenspezifische) Anwendung einsetzen lässt. Dazu gehören Ressourcenmanagement, Design, Widgets und generische Interaktionsmechanismen, wie zum Beispiel die Bewegung eines 3D Objekts durch ein dreidimensionales Koordinatensystem. Weiterhin findet man im GUI Framework Hilfsfunktionen und –Klassen sowie Parser zur Auswertung von persistenten anwendungsunabhängigen Informationen. Die Smart Home GUI steht für die konkrete Implementierung der Arbeit. Die Smart Home GUI ist von interaktiven und individuellen Verhaltensweisen der Widgets abhängig und wird – üblich für eine GUI – dementsprechend viel Gebrauch vom Kommando-Entwurfsmuster machen.

Zu dem GUI Framework gehören Widgets, wie man sie von herkömmlichen WIMP[6]-System kennt. MT4j bringt eine kleine Palette an solchen Widgets mit; Buttons und Slider lassen sich in der Bibliothek beispielsweise finden. Jedoch können in der GUI weitere, komplexere Widgets zum Einsatz kommen. Auch ist der programmatische Umgang der in MT4j vorhandenen Widgets wenig homogen. Das bedeutet, die Schnittstelle der Slider-Klasse und der Button-Klasse unterscheidet sich in vielerlei Hinsicht, was den Implementierungsaufwand für den Anwendungsentwickler dramatisch erhöhen kann. Aus diesem Grunde wird eine Palette bekannter und neuer Widgets entworfen, die ein gemeinsames Java-Interface implementieren. Fabrik-Klassen werden sich für die Instanziierung konkreter Widgets verantwortlich zeichnen. Neben einer Fabrik für Widgets wird es noch Fabrik-Klassen für 3D Modelle, Schriftarten und Grafiken geben. All diese Fabriken implementieren das Singleton-Entwurfsmuster. Um eine hohe Dynamik des Programms zu bieten, soll die konkrete Implementierung einer jeden genutzten Fabrik zur Laufzeit ausgetauscht werden können, sodass zum Beispiel Schriftstile oder die Erscheinung von Widgets jederzeit geändert werden können. Dies geschieht durch eine modifizierte Variante des klassischen Singleton-Entwurfsmusters. Die Modifikation erlaubt, dass das Singleton in einer `setup`-Methode seine künftig zu nutzende konkrete Implementierungsklasse als Parameter entgegen nimmt. Abbildung 4.1 beschreibt diese Singleton-Variante in UML-Form.

[6] Das Akronym WIMP steht für Window, Icon, Menu, Pointer und fasst die gebräuchlichen Elemente heutiger GUIs zusammen.

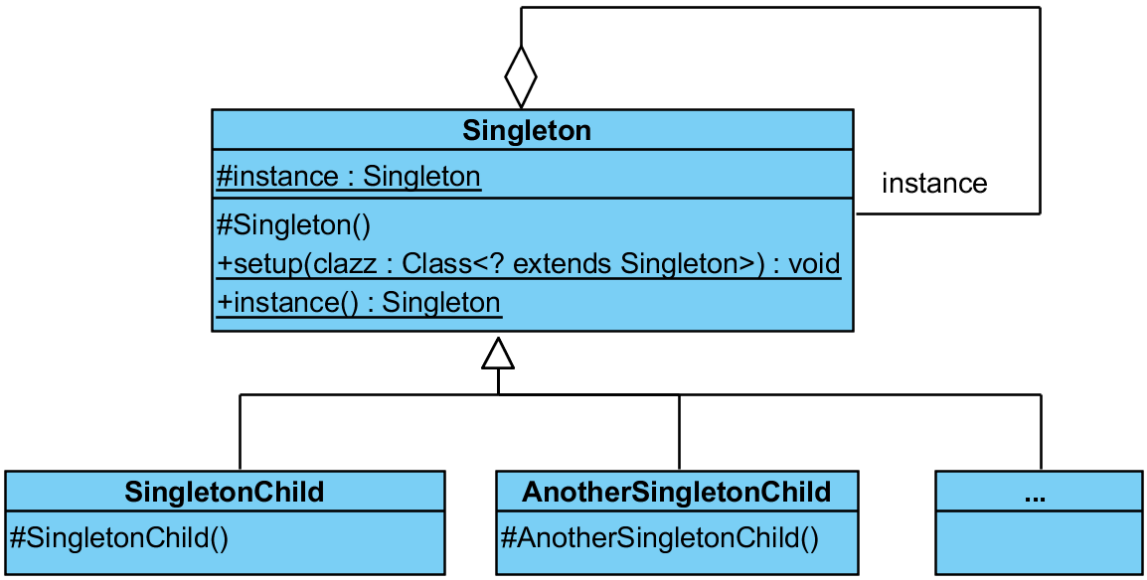

Abbildung 4.1: Die modifizierte Version des Singleton-Entwurfsmusters in UML-Repräsentation.

In der Smart Home GUI ist die interaktive Programmlogik verankert, die die Nutzerschnittstelle selbst ausmacht. Dazu gehört eine Objekt-Repräsentation einer Heimumgebung und seiner Geräte und die konkrete dreidimensionale Szene, welche die Heimumgebung widerspiegelt. Auch gehören zur Nutzerschnittstelle Dialoge, die aus mehreren Widgets aggregiert und mit konkreter, interaktiver Logik verknüpft sind. Diese interaktive Logik wird im Kontext der Arbeit durch das Kommando-Entwurfsmuster umgesetzt. Kommandos stellen in dieser Arbeit zumeist eine Interaktionsmöglichkeit mit spezifischen Geräten im Smart Home dar und können beispielsweise Dialogmenüs für weitere Anweisungen aufrufen. Kommandos sollen nicht zwingend vom Anwendungsprogrammierer explizit gestartet werden, vielmehr muss der Anwendungsentwickler eine Fabrik-Klasse für die Generierung verschiedener Kommandos erstellen. Diese Fabrik-Klasse kann bei der Objekt-Repräsentation der Heimumgebung registriert werden, sodass die korrekten Kommandos für die entsprechenden Gerätetypen automatisch aufgerufen werden und potenzielle Programmierfehler vermieden werden. Auf diese Weise kann der Anwendungsentwickler mit wenigen Befehlszeilen die eigentliche GUI schreiben und wird darüber hinaus dahin geleitet, modularen und wieder verwendbaren Quelltext zu verfassen. Nichtsdestotrotz sollen Kommandos auch vom Entwickler explizit aufgerufen werden können, sofern sich dies als nötig erweisen sollte. Da spezifische Kommandos mit spezifischer Logik verbunden sind, wird hier vom GUI Framework keine Trennung von Logik und Design vorgeschrieben. Da Kommandos in der Lage sein sollen, ganze Dialoge zu generieren, sollte es auch die Möglichkeit geben, diese Dialoge auf eine einheitliche Art und Weise wieder zu deaktivieren. Zu diesem Zwecke wird das gewöhnliche Interface vom Kommando-Muster um eine Abschalt-Funktion erweitert. Mithilfe der `cease()`-Methode soll die Arbeit eines laufenden Kommandos – zum Beispiel das Rendern von Menüelementen – eingestellt werden. Hierbei soll das Kommando-Objekt nicht vernichtet werden, sondern für den späteren Gebrauch weiterhin bereit stehen. Ein Sequenzdiagramm für die mögliche Nutzung eines Kommando-Objekts in der Smart Home GUI wird in Abbildung 4.2 dargestellt. In der Grafik wird ein Kommando von

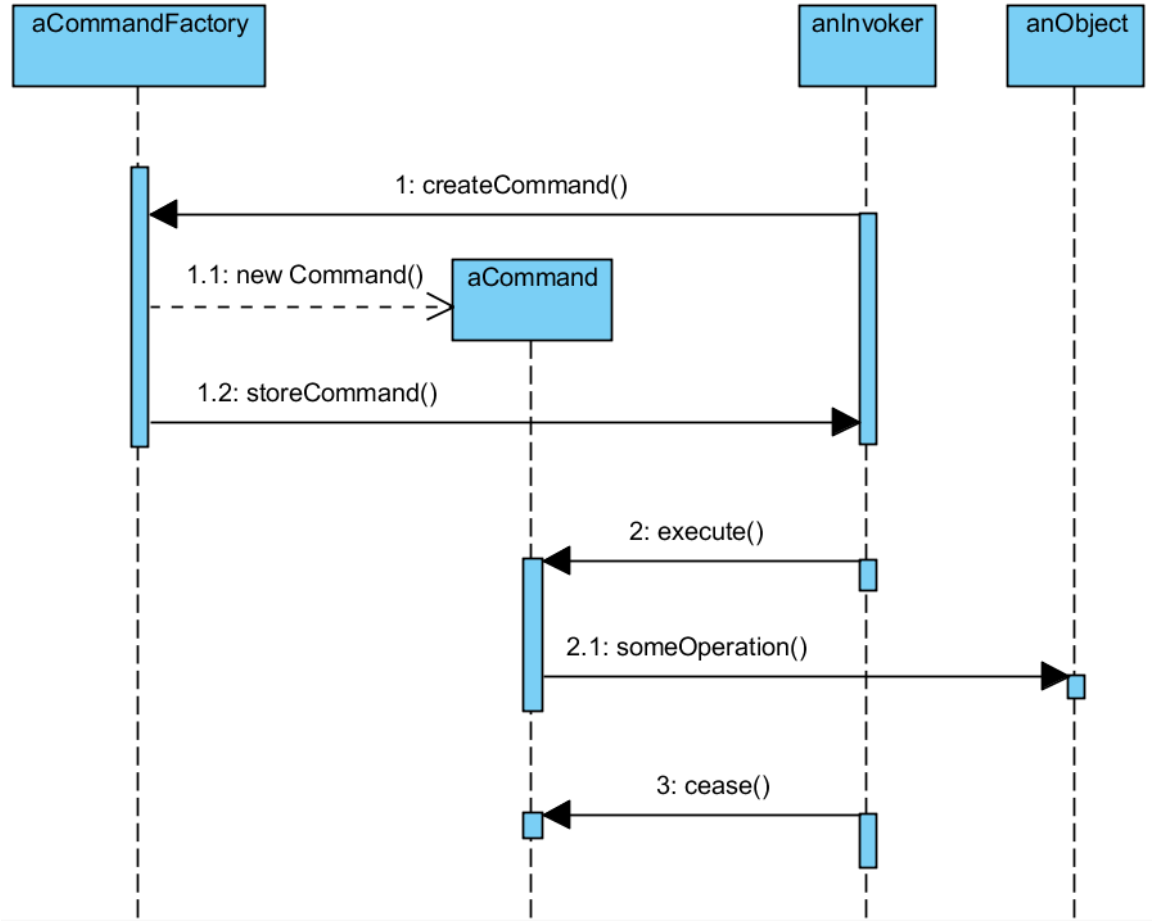

Abbildung 4.2: Sequenzdiagramm für die Verwendung von Kommando-Objekten.

einer Fabrik erstellt und nimmt nach dem Funktionsaufruf `execute()`seine Arbeit auf, bis es von einem Klienten mit `cease()` beendet wird. In der Praxis steht es einer konkreten Kommando-Implementierung natürlich frei, sich nach ausgeführter Arbeit selbst mit `cease()` zu deaktivieren oder von einem anderen Objekt abgeschaltet zu werden.

Zwei weitere Bestandteile der Smart Home GUI existieren. Die ersten sind Visualisierungsmodule, welche die grafischen Bestandteile der GUI entsprechend des Zustands des Smart Home manipulieren können, die zweiten sind anwendungsspezifische Persistenzklassen. Die ersteren, sogenannten *StateVisualizer* benannten Module können etwa die Helligkeit der Textur eines dreidimensionalen Lampenmodells an den Zustand des realen Pendants anpassen. Auch hier wird die Verwaltung der verschiedenen *StateVisualizer*-Entitäten anhand von Fabrik-Klassen gehandhabt, gleich nach dem Schema wie bei den Kommando-Fabriken. Die Persistenzklassen, die *StateReaders* und *StateWriters*, sollen Informationen bezüglich der GUI speichern und laden können. Hierbei soll nicht vorgeschrieben werden, wie die Daten gespeichert bzw. gelesen werden. So sind Datenzugriffe auf die Festplatte als auch wie die Übertragung als Datenstrom via Netzwerk möglich. Ebenso ist aber jede weitere Art der Datenablage bzw. Datenerfassung denkbar. Die *StateReaders*/*StateWriters* sollen jedoch nicht die Zustandsinformationen der Heimumgebung selbst verarbeiten, diese Informationen werden von der MASP (3.1) verwaltet.

5 IMPLEMENTIERUNG

In diesem Abschnitt werden die konkreten Java-Implementierungen des GUI Framework und der Smart Home GUI beschrieben. Dabei wird Modul für Modul auf die einzelnen abstrakten Klassen und Interfaces sowie die deren Implementierungen eingegangen. Für die konkreten Javadoc[7]-Dokumentationen ist der entsprechende digitale Anhang zu Rate zu ziehen.

5.1 GUI Framework

Das GUI Framework ist eine generische Klassenbibliothek, welche sich nicht nur für die Entwicklung von Smart Home GUIs eignet. Aufbauend auf dem MT4j Framework ergänzt das GUI Framework dieses um neue Widgets, wie zum Beispiel Graph-Plotter und Checkbuttons, aber auch um einfach zu nutzende 3D-Objekte. Hinzu kommen dynamische Ressourcenmanagement-Klassen und Fabrik-Klassen für Schriftarten, Texturen, 3D-Objekte und Widgets. Zwei weitere Bestandteile sind die so genannten *AnimationListeners* und eine abstrakte Implementierung einer Szenen-Klasse. Erstere sind in der Lage, Veränderungen von Widgets, zum Beispiel Positions- oder Farbveränderungen, über eine gewisse Zeit zu interpolieren und so eine flüssige Animation abzuspielen. Szenen hingegen stellen die Grundlage für eine sichtbare Nutzerschnittstelle dar. Sie bieten eine Kamera in den virtuelle 3D-Raum und eine Oberfläche, auf die die Widgets gezeichnet werden können. Eine abstrakte Implementierung einer Szenen-Klasse wurde im GUI Framework realisiert, um durch polymorphe Subklassen Aussehen und Handhabung der einzelnen Szenen und Dialoge in der GUI zu vereinheitlichen.

5.1.1 Ressourcenmanagement & Provider

Ressourcenmanagement ist ein wichtiger Bestandteil des GUI Framework und ermöglicht die reibungslose Arbeit mit Schriften, Bildern und 3D-Objekten. Zwar gibt es in MT4j eingebaute Ressourcenmanager, doch sind diese sehr generisch und keineswegs auf die Arbeit mit dem GUI Framework ausgelegt. Die zum Framework gehörigen Ressourcenmanager – deren Bezeichnung im Projekt *Provider* lautet – dienen vor allem der konsistenten Nutzung eines gemeinsamen Ressourcenpools. *Provider* abstrahieren von individuellen, durch Namen identifizieren Ressourcen hin zu Ressourcen für bestimmte Zwecke und Klienten. Beispielsweise gibt es eine Schriftart für Buttons und eine andere Schriftart für Textfelder, doch wird die Schriftart nicht über ihren Namen identifiziert, sondern über ihre Zugehörigkeit, zum Beispiel über ihre Zugehörigkeit zu Button-Widgets. Man übergibt einem Provider nur einen Enumerationswert, welcher den Zweck bzw. das Ziel der angeforderten Ressource angibt. Welche spezifische Ressource im einzelnen Fall verwendet wird, legt die konkrete Implementierung des Providers fest. Dieses Prinzip lässt sich auf alle anzutreffenden Provider übertragen. Im GUI Framework gibt es drei Provider-Arten: je Provider für 3D-Modelle, für Texturen und für Schriftarten. Sie

[7] http://www.oracle.com/technetwork/java/javase/documentation/javadoc-137458.html. Abgerufen am 31.05.2012

liegen im Paket `provider` und tragen der Kürze wegen die Namen `GUI3DModels`, `GUIFonts` und `GUITextures`. Diese drei Provider implementieren Standardverhalten und können für verändertes Verhalten abgeleitet werden. Alle Provider implementieren die in 4.2 vorgestellte Variante des Singleton Musters und die Fabrikmethode. Es liegt für die beiden Provider `GUI3DModels` und `GUITextures` bereits je eine Subklasse vor, um die Austauschbarkeit von Ressourcen zu demonstrieren. Eine weitere Eigenschaft der Provider ist, dass nicht spezifiziert ist, wie und woher Ressourcen bezogen werden. So können diese beispielsweise vom Dateisystem geladen, aus dem Netzwerk als Datenstrom bezogen, oder gar zur Laufzeit prozedural erstellt werden. In den konkret existierenden Implementierungen werden zwar Dateien vom lokalen Pfaden geladen und verwendet, doch wird für mögliche Erweiterungen viel Freiheit im Umgang mit Ressourcen gelassen.

Da das Laden von Ressourcen fühlbar viel Zeit in Anspruch nehmen kann, wird in den gegebenen Implementierungen der Provider jede Ressource, die von der GUI verwendet werden könnte, bei der Initialisierung des Providers vom Dateisystem in den Arbeitsspeicher geladen. Dieser Initialisierungsprozess findet in der Implementierung der Smart Home GUI bei Programmstart statt. Auf diese Weise werden alle notwendigen Ressourcen am Anfang in einem mehr oder minder zeitaufwändigen Prozess geladen, sind dann jedoch ohne Latenz zur Laufzeit verfügbar, sobald sie benötigt werden. Je nach Anzahl, Komplexität und Größe der Texturen, 3D-Modelle und Schriftarten kann beliebig viel Arbeitsspeicher beansprucht werden. Die Menge des benötigten Speichers muss der zugelassenen Speichergröße seitens der virtuellen Maschine von Java entsprechen und kann je nach Hardware erhebliche Ladezeiten bewirken. Aufgrund dessen wurden im Laufe der Arbeit neben hochauflösenden Textursätzen und komplexen 3D-Modellen auch Ressourcensätze mit geringer Qualität und Größe entwickelt, welche wesentlich schneller geladen werden können und somit den Entwicklungs- und Testprozess beschleunigten.

5.1.2 Widgets

Widgets sind die sichtbaren GUI-Komponenten, die im GUI Framework existieren können. Dazu gehören herkömmliche Widgets wie Buttons, Checkbuttons, Auswahl-Container Slider und Tab-Container, doch findet man auch 2D-Manipulatoren – eine Art zweidimensionalen Slider, 3D-Objekte und Graph-Plotter. All diese Widgets haben eine abstrakte Oberklasse: das `AbstractGUIWidget`, welche wiederum vom der MT4j-Klasse `MTComponent` erbt. Die Klasse `AbstractGUIWidget` implementiert einfache Hilfsfunktionen für den Umgang mit Widgets und spezifiziert eine Java-Schnittstelle zur einfachen und einheitlichen Positionierung von Widgets auf dem Bildschirm.

`AbstractGUIWidget` führt das Konzept der Modi ein, die den Zustand und das Verhalten der Widgets markieren können. Diese Modi sind Enumerationswerte und heißen *Normal*, *Disabled*, *Tap*, *Signal* und *Signal2*. Verschiedene Modi können das Aussehen und Verhalten der Widgets beeinflussen und so dem Nutzer ein gewisses Interaktionsfeedback geben oder Aufmerksamkeit auf sich lenken. Beispielsweise markiert der Modus *Normal* das herkömmliche Verhalten eines Widgets, während ein Widget im Modus *Disabled* nicht verwendet werden kann. *Tap* stellt den Zustand bei momentaner Fingerberührung dar und eignet sich, um dem Nutzer programmseitige Resonanz auf eine

Geste zu geben. *Signal* und *Signal2* stellen verschiedene Modi dar, die die Aufmerksamkeit des Nutzers auf das Widget ziehen sollen, indem sie eine Komponente etwa farblich von den anderen abheben. Was das konkrete Verhalten bei Anwendung der verschiedenen Modi auf die einzelnen Widgets bewirkt, hängt von deren konkreten Implementierungen dieser ab.

Von der Klasse `AbstractGUIWidget` leiten weitere abstrakte Klassen ab, die das Interface und Verhalten der verschiedenen Widget-Typen weiter spezifizieren. Von diesen abstrakten Klassen können schließlich konkrete Klassen für die einzelnen Widget-Kategorien erstellt werden. Zu jeder abstrakten Widget-Spezifikation liegt dem GUI-Framework eine Implementierung bei.

Zur einfachen und konsistenten Kreation der Widgets in der GUI wird bei den Widgets wie beim Ressourcenmanagement auf die Nutzung einer Singleton Fabrik zurück gegriffen. Die polymorphe Fabrik kann wie die Ressourcenprovider zur Laufzeit ausgetauscht werden. Da mehrere konkrete Implementierungen eines Widget-Typen existieren können, besteht die Gefahr, dass ein unachtsamer Programmierer etwa verschiedene Button-Typen in der GUI einsetzt oder generell das Design der GUI nicht konsequent handhabt. Die Nutzung einer Fabrik reduziert das Risiko für derartige Designfehler. Nichtsdestotrotz kann man Widget-Objekte auch direkt instanziieren, sodass der Entwickler alle Freiheiten hat, sollte er sie benötigen.

5.1.3 GestureListeners

Gesture Listeners sind ein Konzept, welches mit MT4j eingeführt wird. Implementierungen des `IGestureEventListener`-Interface können an den ebenfalls in MT4j benutzten Gesten-Prozessoren registriert werden und ermöglichen das Triggern von Ereignissen, die beim Tätigen gewisser Gesten geschehen sollen. Spezifische *Gesture Listeners* können mit bestimmten GUI-Komponenten, zum Beispiel Widgets, verbunden werden und kapseln die Logik, die ausgeführt werden soll, wenn man mit einem Widget oder einem 3D-Modell interagiert. Wenngleich *Gesture Listeners* für die meisten Aufgaben speziell implementiert werden müssen, so gibt es doch einige wenige Listeners, die immer wieder verwendet werden können. Diese wurden in dem Paket `gestureListeners` in den Unterpaketen `interaction` und `transformation` abgelegt. Im Unterpaket `interaction` liegen generische *Gesture Listeners*, die verschiedene allgemeine Aufgaben mit dem Fokus auf Interaktion übernehmen können. So findet man zum Beispiel einen Listener für fingergesteuerte Zoom-Aktionen und einen Listener, der bei Fingerdruck auf ein Widget dessen Zustand kurzzeitig in den *Tap*-Modus überführt. Einige der vorzufindenden Listener werden nicht in der endgültigen Smart Home GUI verwendet, sind jedoch aus der Entwicklungsphase des Projekts bestehen geblieben. *Gesture Listeners* im Unterpaket `transformation` dienen dazu, GUI-Komponenten im dreidimensionalen bzw. zweidimensionalen Koordinatensystem zu transformieren. Da die Manipulation von Objekten im 3D-Raum auf einer zweidimensionalen Projektionsfläche wie einem Multitouch-Display zwangsläufig zu Unklarheiten führt, gibt es verschiedene Listeners für die Transformation von Objekten im dreidimensionalen Raum. Mithilfe des `Displacement3DPlaneListener` können Objekte auf einer zuvor spezifizierten virtuellen Ebene im 3D-Raum bewegt werden. Hierbei

wird einfach das betreffende Objekt mit einer Wisch-Geste auf der Virtuellen Ebene verschoben. Diese virtuelle Ebene muss nicht zwangsläufig parallel zur Bildebene sein. Durch die Reduktion auf zwei Freiheitsgrade in einer Bewegungsebene ist die Verschiebung von Objekten bis auf die Situation, in der die Kamera selbst in der Bewegungsebene liegt, immer eindeutig. Für diesen Sonderfall deaktiviert der `Displacement3DPlaneListener` seine Fähigkeiten, um undefiniertem Verhalten vorzubeugen. Abbildung 5.1 veranschaulicht diese Situation noch einmal. Eine weitere denkbare Art der Positionsveränderung im dreidimensionalen Raum lässt sich mit dem `Displacement3DLineListener` umsetzen, mit dessen Hilfe Objekte auf einer Geraden im dreidimensionalen Raum verschoben werden können. Hier interagiert der Anwender mit einem Objekt wie beim `Displacement3DPlaneListener`, jedoch beschränkt sich die Bewegungsrichtung in nur eine zuvor festgelegte Richtung. Die Anzahl der Freiheitsgrade dieser Aktion ist eins, somit ist auch diese mit einfachen Gesten auf einem Eingabegerät wie einem Multitouch-Display umsetzbar. Wieder kann es zu einer undefinierten Situation kommen, sollte die Gerade, auf der sich ein Objekt verschieben lässt, durch den Positionspunkt der Kamera verlaufen. In solchen Situationen wird ebenfalls die Funktionalität des Listeners schlicht deaktiviert, um undefiniertem Verhalten vorzubeugen. Das Problem ist in Abbildung 5.1 noch einmal veranschaulicht.

Weiterhin gibt es Listener für die 3D-Rotation um einen gewünschten Vektor, die dreidimensionale Skalierung von Objekten, die zweidimensionale Translation von Komponenten sowie einen Freihand-Rotationslistener, bei dem die Richtung einer einfachen Wisch-Geste die Rotationsebene beschreibt. Einige transformationsspezifische Listener lassen sich nicht nur auf alle Widgets aus dem GUI Framework anwenden, sondern können in Kombination mit den generischen GUI-Komponenten von MT4j, den Unterklassen vom Typ `MTComponent` verwendet werden.

5.1.4 AnimationListeners

Die *AnimationListeners* implementieren ein MT4j-Interface, das Ablaufverhalten für Animationen bereit stellt. Animationen interpolieren eine reelle Zahl von einem spezifizierten Anfangs- zu einem spezifizierten Zielwert und rufen in jedem Interpolationsschritt eine Methode auf, die vom Entwickler gestaltet werden kann. Dabei kann der Programmierer auf den interpolierten Zahlenwert zurück greifen, um kleine Subroutinen davon abhängig zu machen. So können beispielsweise automatische Bewegungen animiert und Farbübergänge erzeugt werden. Es existieren zwei *AnimationListener* zum GUI-Framework. Erstere, `AnimMoveAction` ermöglicht das flüssige animierte Bewegen von GUI-Komponenten im dreidimensionalen Raum. Der zweite *AnimationListener*, `AnimRotate3DAction` ermöglicht das Rotieren eines 3D-Objektes um sein Zentrum entlang einer beliebigen Koordinatenachse. Da Animationen zwar ein für die Softwareergonomie ein unterstützendes Mittel sind, jedoch für die Smart Home GUI selbst eine weniger wichtige Rolle spielen, werden sie in der Smart Home GUI selbst nicht verwendet, sind aber für eventuelle spätere Nutzung verfügbar. Es existiert wegen der geringen Anzahl von nur zwei *AnimationListeners* keine Fabrik-Klasse, jedoch könnte sich eine solche bei Erweiterung des `AnimationListeners`-Paktes als sinnvoll erweisen.

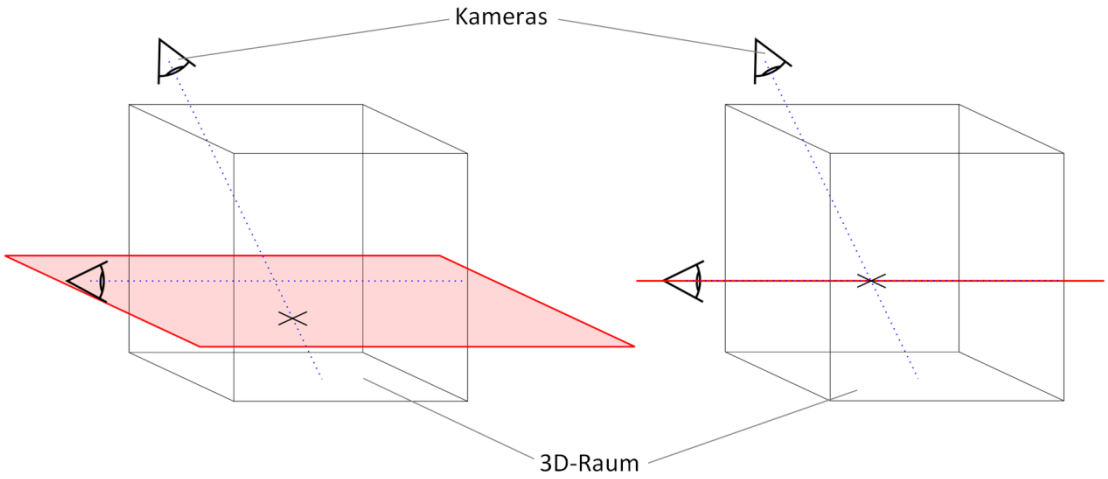

Abbildung 5.1: Zwei Situationen, in denen sich die Freiheitsgrade des `Displacement3DPlaneListener` (rechts) und des `Displacement3DLineListener` (links) durch Überlagerung der Bewegungsebene, bzw. Bewegungsgerade (jeweils rot dargestellt) mit der Kameraposition reduzieren. Der Quader stellt jeweils den dreidimensionalen Raum dar, in den die Kamera, dargestellt als Auge, hinein sieht. Liegt die Kamera nun innerhalb der Bewegungsebene, beziehungsweise innerhalb der Bewegungsgerade, sieht der Nutzer im projizierten Bild durch die Kamera keine Ebene bzw. Gerade mehr, der Nutzer kann dementsprechend keinen Punkt mehr auf der Ebene/Geraden eindeutig auswählen, wodurch die gewohnte Handhabung nicht mehr möglich ist. Dies trifft für die unteren beiden Kameras zu. Die beiden oberen Kameras stellen jeweils das Normalverhalten dar.

5.1.5 Scenes-Paket

Im `scenes`-Paket ist vor allem die abstrakte Klasse `AbstractGUIScene` von großer Bedeutung. `AbstractGUIScene` stellt eine Basisklasse für alle Szenen bereit. Szenen in MT4j sind vergleichbar mit Fenstern in den sogenannten WIMP-Interfaces. Alles, was visuell dargestellt wird, wird in einer Szene gerendert. Eine Szene legt essentielle Eigenschaften wie die Hintergrundfarbe fest und bietet die Programmschnittstelle für Tastatursignale. `AbstractGUIScene` spezialisiert die MT4j-eigene Klasse `AbstractScene`, indem sie diese um neue Hilfsfunktionen erweitert, die vor allem bei komplexeren Szenen hilfreich sind. So existiert etwa eine Hilfsfunktion zur Suche von in der GUI verwendeten Widgets anhand ihrer Namen. Auch gewisse Tastenbefehle werden von `AbstractGUIScene` mit Logik verknüpft, hauptsächlich für Debug-Zwecke. Welche Tastenbefehle existieren, wurde in der Javadoc-Dokumentation vermerkt. Jede in einer konkreten GUI verwendete Szene, die von `AbstractGUIScene` erbt, verfügt über die gleiche besprochene Logik und hält sich an dieselben Designrichtlinien wie etwa Hintergrundfarbe und Kameraeinstellung. Neben der `AbstractGUIScene` existieren weitere Szenen im Paket. Diese wurden für Testzwecke implementiert und werden als Hilfsmittel für den Programmierer benötigt.

5.1.6 Util-Paket

Im `util`-Paket liegen mehrere unterstützende Klassen, die sich nicht in die vorhandenen Pakete einordnen lassen. Hauptsächlich beinhaltet das Paket einige nützliche Enumera-

tionen, wie die `DeviceType` Enumeration, welche eine Auflistung aller in der Smart Home GUI unterstützten Gerätetypen beinhaltet.

5.2 Smart Home GUI

Die Smart Home GUI ist eine Implementierung für die Steuerung einer intelligenten Heimumgebung, welche die Schnittstelle der MASP (3.1) nutzt, um mit dieser Heimumgebung zu kommunizieren. Aufbauend auf dem GUI Framework nutzt sie MT4j als Vehikel zur Darstellung des Schauraums des DAI Labors sowie seiner Geräte. Generell kann die Smart Home GUI jedoch mit sehr wenig Aufwand an eine beliebige intelligente Heimumgebung angepasst werden. Die Smart Home GUI besteht im Wesentlichen aus einer Szene, die ein `Home`-Objekt instanziiert. Dieses `Home`-Objekt wird mit einem Modell einer intelligenten Heimumgebung mithilfe der MASP verbunden und kontrolliert die visuelle Darstellung sowie die Logik des der Heimumgebung und seiner Geräte, innerhalb der GUI so genannten *Items*. Es existieren zusätzlich Komponenten zum Speichern und Laden von Zuständen sowie zur zustandsabhängigen Veränderung der visuellen Darstellung von Geräten. Kommando-Entwurfsmuster implementieren den Programmablauf bei Interaktion mit gewissen Gerätetypen.

5.2.1 Home

Die `Home` Klasse repräsentiert selbst eine Wohnung im Sinne einer MT4j GUI. Ein `Home` verwaltet *Items*, Instanzen der Klasse `Item` (5.2.2). `Home` beherbergt die Ansichten der Wohnung sowie der Geräte und verbindet diese mit ihren physischen Gegenparts. Eine weitere Aufgabe des `Home` ist die Ermöglichung der Steuerung der Geräte. Sämtliche 3D-Modelle der Haushaltsgeräte werden hier mit den Fähigkeiten ausgestattet, die sie für die umfangreiche Nutzung in einer GUI benötigen, jedoch geschieht dies versteckt und für den Anwendungsprogrammierer unsichtbar. Dieser muss lediglich die anfängliche Position des 3D-Modells der Wohnung in der Szene sowie den nach oben zeigenden Vektor, den Oben-Vektor, spezifizieren, um die Steuerung vollständig zu initialisieren. Der Oben-Vektor ist nötig, um die Interaktion mit der GUI entsprechend der Orientierung der 3D-Objekte im Raum gestalten zu können, etwa um Rotationen um die vertikale Achse zu ermöglichen. Da die Wohnung und die Geräte, die *Items*, in Hinblick auf ihre grafischen Repräsentationen vom Nutzer konfiguriert werden können, existiert auch eine Schnittstelle zum Speichern und Laden eben solcher Konfiguration. Die *StateReaders* & *StateWriters* (5.2.3), können jederzeit an ein `Home`-Objekt angeschlossen werden um den Zustand der GUI zu persistieren oder von einer externen Ressource auszulesen.

5.2.2 Item

Instanzen der Klasse `Item` spiegeln Haushaltsgeräte in der Wohnung wieder. *Items* können mit Modellen ihres physikalischen Gegenparts mithilfe der MASP verbunden werden und adaptieren je nach Typ des realen Gerätes deren Aussehen, sofern der spezifische Typ in der Enumeration `util.DeviceType` gelistet ist und ein entsprechendes 3D-Modell vom Ressourcen-Provider geladen wurde. Das Modell muss hier den gleichen Namen tragen, wie der Gerätetyp in `util.Devicetype`. Sollte einem gewissen Gerätetyp kein 3D-Modell zugewiesen werden können, wird stattdessen ein generischer

Platzhalter angezeigt. Entsprechend des konkreten Gerätetyps verbinden *Items* auch automatisch *DeviceCommands* (5.2.4) und *StateVisualizer*s (5.2.5) mit einem Gerät. Sollte hier der spezifische Typ nicht in der Enumeration `util.DeviceType` vorkommen, wird ein Dummy-Kommando bzw. ein Dummy-Visualizer als Platzhalter eingefügt. Generell kapseln die *Items* die Fähigkeiten der konkreten Geräte weitgehend automatisch, ein Anwendungsprogrammierer muss lediglich die 3D-Modelle der *Items* selbstständig zur Szene hinzufügen und kann diese eingangs mit beliebigen *GestureListeners* versehen. In der entwickelten GUI wurden *Items* mit `Displacement3DPlaneListeners` und `Rotation3DListeners` (5.1.3) versehen. Auf diese Weise kann man etwa durch eine Wisch-Geste auf einem Geräteabbild durch die 3D-Umgebung navigieren.

5.2.3 StateReaders & StateWriters

StateReaders und *StateWriters* in dem Paket `stateReadersWriters` sind Schnittstellen, die die Aufgabe haben, Informationen bezüglich der GUI zu persistieren, beziehungsweise von einer Quelle auszulesen. Hierbei werden lediglich die Methoden `save()`, beziehungsweise `load()` vorgeschrieben. Existierende Implementierungen der `StateReader` und `StateWriter`-Schnittstellen sind die Klassen `XMLReader` und `XMLWriter`. Diese schreiben, beziehungsweise laden, XML-Dateien mithilfe des jdom-Frameworks[8]. In diesen Dateien werden Informationen zur Darstellung der Heimumgebung gespeichert. Da der Nutzer beispielsweise Positionen und Darstellung von Geräten zur Laufzeit verändern kann, werden diese gespeichert und bei Neustart ebenfalls neu geladen. Gespeicherte Dateien, die mithilfe von `XMLReader`- und `XMLWriter`-Instanzen erstellt wurden, werden im Projektunterverzeichnis ./saves gespeichert. Generell ist jedoch aufgrund der einfachen Schnittstelle jede Form der Datenpersistenz denkbar.

5.2.4 DeviceCommands

Die Logik, die beim tippen auf dreidimensionale Gerätemodelle in der GUI ausgeführt wird, wird in den so genannten *DeviceCommands* verankert. Im Falle der konkreten GUI wird hier etwa das Öffnen eines bestimmten Dialogwidgets eingeleitet. Dieses bestimmte Widget kann in der Folge mehr Möglichkeiten zur Interaktion mit sich bringen. Etwa An/Aus-Schalter oder auf das Gerätemodell spezialisierte Funktionalitäten (wie beispielsweise die Temperaturregulierung einer Herdplatte). Für jede unterstützte Art von Gerätetyp muss eine konkrete Subklasse der abstrakten Mutterklasse `DeviceCommand` aus dem Paket `home.deviceCommands` erstellt werden. Sollte der spezifische Typ nicht in der Enumeration `util.DeviceType` vorkommen, wird ein Dummy-Kommando als Platzhalter eingefügt. Die konkreten *DeviceCommands* können als Ausgangspunkt für weitere Interaktionsmöglichkeiten angesehen werden. Eine bestehende Gruppe von einfachen *DeviceCommands* befindet sich im Paket `home.deviceCommands.std`. Diese Gruppe von *DeviceCommands* erzeugt bei Antippen von Geräten einfache Dialoge am rechten Bildschirmrand, in denen grundlegende Informationen über den Zustand der Geräte angezeigt und verändert werden können.

[8] http://www.jdom.org. Abgerufen am 24.05.2012

DeviceCommands sollten in aller Regel nicht vom Anwendungsprogrammierer direkt instanziiert werden. Stattdessen sollte eine Fabrik-Klasse implementiert werden, die von der Klasse `CommandFactory` im Paket `home.deviceCommands.factories` erbt. Es muss lediglich die Methode `createCommand(Item, PhysicalDevice)` überschrieben werden. Diese Methode erstellt zu einem gegebenen `Item` und einem gegebenen Modell eines physikalischen Gerätes entsprechend des Gerätetyps ein zugehöriges `DeviceCommand` und liefert es zurück. Das `Item`-Objekt beschreibt den Kontext des Kommandos, und vermittelt indirekt Informationen über das 3D-Modell und weitere grafische Eigenschaften der visuellen Geräterepräsentation, wohingegen `PhysicalDevice` das wahrhaftige physikalische Modell seitens der MASP liefert. Die Klasse `CommandFactory` wurde als Singleton-Klasse nach dem gleichen Muster wie die anderen Fabriken im GUI-Framework realisiert. Auch die `CommandFactory` muss also vor Verwendung initialisiert werden. Ein gegebener Zeitpunkt ist am Anfang des Programms oder beim Laden der ersten Szene, die auf ein `Home`-Objekt zurück greift.

5.2.5 StateVisualizers

Um die grafischen Repräsentationen der vom GUI-Framework unterstützten Geräte in der GUI abhängig vom Zustand der echten Geräte machen zu können, wurden die *StateVisualizer* eingeführt. Instanzen dieser Klasse können über Zustandsänderungen von ihnen zugeordneten Geräten manuell oder mit Hilfe der MASP informiert werden. So kann man beispielsweise die Helligkeit eines 3D-Modells einer Lampe innerhalb der GUI anpassen, wenn sich auch ihr realer Gegenpart ändert. *StateVisualizer* für spezielle Geräte sollen von der abstrakten Klasse `StateVisualizer` im Paket `home.stateVisualizers` erben. Hier muss lediglich die Methode `notifyVisualizer()` überschrieben werden, welche bei Aufruf die visuelle Repräsentation, wie etwa die Helligkeit oder die Größe eines Gerätes anpasst. Zurzeit existieren drei *StateVisualizer*. Die ersten Beiden sind der `LampStateVisualizer` und der `BlindStateVisualizer`. Der Dritte ist der `DummyVisualizer`. Sollte der spezifische Gerätetyp für einen Visualizer nicht in der Enumeration `util.DeviceType` vorkommen, also von der GUI nicht unterstützt werden, wird ein `DummyVisualizer` als Platzhalter eingefügt, der die visuelle Repräsentation von 3D-Objekten nicht verändert. Wie für *DeviceCommands* gilt für die Handhabung mit *StateVisualizers*, dass der Anwendungsprogrammierer in der Regel nicht mit ihnen in Berührung kommt. Die Kreation konkreter *StateVisualizer* wird gleich wie bei den *DeviceCommands* in eine Fabrik-Klasse ausgelagert. Eine solche Fabrik-Klasse muss eine Subklasse vom Typ `StateVisualizerFactory` sein, welche im Paket `home.stateVisualizers.factories` liegt. Es muss lediglich die Methode `createStateVisualizer(Item, DeviceType)` überschrieben werden, welche entsprechend des gegebenen `DeviceType`-Enumerationswertes einen *StateVisualizer* erstellt. Das übergebene `Item` stellt den Kontext für den *StateVisualizer* dar, sodass dieser Informationen bezüglich des zu verändernden 3D-Modells beziehen kann.

6 DIE ANWENDUNG

Die fertig gestellte Smart Home GUI besteht aus einer Szene, die das 3D-Modell der Heimumgebung und seiner ihm innewohnenden Geräte darstellt. Am linken Fensterrand existieren zwei Slider. Der Obere dient dem Kamerazoom per Ein-Finger-Geste, der untere ermöglicht die Neigung des Blickwinkels. Das Zoom lässt sich zusätzlich mit einer auf dem Modell der Wohnung ausgeführten Zwei-Finger-Geste betätigen. Hierbei ist zu beachten, dass beide Finger das 3D-Modell der Wohnung berühren müssen, um den entsprechenden MT4j-Eingabeprozessor zu aktivieren. Mithilfe einer Wisch-Geste mit einem Finger lässt sich das 3D-Modell in der horizontalen Ebene verschieben, mit zwei Fingern kann man neben dem Zoom gleichzeitig eine Rotation um die vertikale Achse des Modells bewirken. Damit ist eine Trennung von Rotation und Translation erreicht, was nach (2.2.1) eine höhere Anwenderfreundlichkeit zur Folge hat, als eine Vermischung beider Aktionen mit einer Geste. Lediglich das Zoom wurde nicht von der Rotation getrennt, was sich jedoch nicht als hinderlich erwies, da in der Smart Home GUI keine absolut akkuraten Bewegungen durchgeführt werden müssen.

Die Interaktion mit Geräten kann durch einfaches Tippen auf einen Gegenstand initiiert werden. Durch das Tippen wird ein mit dem Gerät, bzw. seinem `Item`, verbundenes Kommando aktiviert, welches ein Dialogmenü öffnet. Das Menü erscheint als Leiste am rechten Fensterrand. Diese Leiste zeigt den Namen des Gerätes an, blendet essentielle Zustandseigenschaften ein und stellt Widgets zur Manipulation dieser Eigenschaften bereit. Abbildung 6.1 zeigt diese Leiste nach antippen einer Lampe. Man sieht einen An/Aus-Schalter sowie einen Slider, mit welchem man die Helligkeit der Lampe dimmen kann. Ändert man den Zustand der realen Lampe – ungeachtet, ob per herkömmlichen Schalter oder der Smart Home GUI – wird ebenso die Helligkeit der Lampe in der 3D-Umgebung angepasst. Haushaltsgeräte können mehrere untergeordnete Geräte bündeln. Beispielsweise besteht eine Dunstabzugshaube aus einer Lampe und einem Ventilator. Um solch aggregierte Geräte zu nutzen, tippt man auf das 3D-Modell des übergreifenden Gerätes (etwa die Dunstabzugshaube) und wählt dann im rechten Menü das entsprechende untergeordnete Element aus (Abbildung 6.2). Sollte der Anwender die Position von Geräten verändern wollen, kann er nach Antippen eines Geräte-Modells mit einem Druck auf den Button mit dem Wort „Edit“ ein Menü aufrufen, welches ihm die Neupositionierung des Gerätes ermöglicht. Die Edit-Funktion innerhalb der Smart Home GUI verkörpert den Ansatz der Endbenutzer-Entwicklung, wie er in Kapitel 1 erwähnt wurde. Bei Nutzung der Edit-Funktion kann der Anwender auf der rechten Bildschirmseite einen oder mehrere Checkbuttons bedienen, um das 3D-Modell auf verschiedene Weisen im Raum zu bewegen. Er kann das Objekt (1) in der Horizontalen Ebene bewegen, (2) Vertikal verschieben, (3) Rotieren und (4) Skalieren. Es ist auch möglich, mithilfe des Buttons „Change Style“ das 3D-Modell durch ein anderes auszutauschen. Sollte beispielsweise die Darstellung einer Lampe der realen Lampe sehr unähnlich sein, kann man eine Darstellung wählen, die der physischen Lampe ähnlicher ist. Die Einstellungen, die im Edit-Menü getroffen wurden, können gespeichert werden und werden bei Neustart der Anwendung automatisch wieder geladen.

Abbildung 6.1: Die Smart Home GUI. Am linken Rand sieht man den Slider für die Zoom-Einstellung (oben) und den Slider für die Neigung des 3D-Modells (unten). Am rechten Rand sieht man ein Menü für die Interaktion mit einer Leuchte.

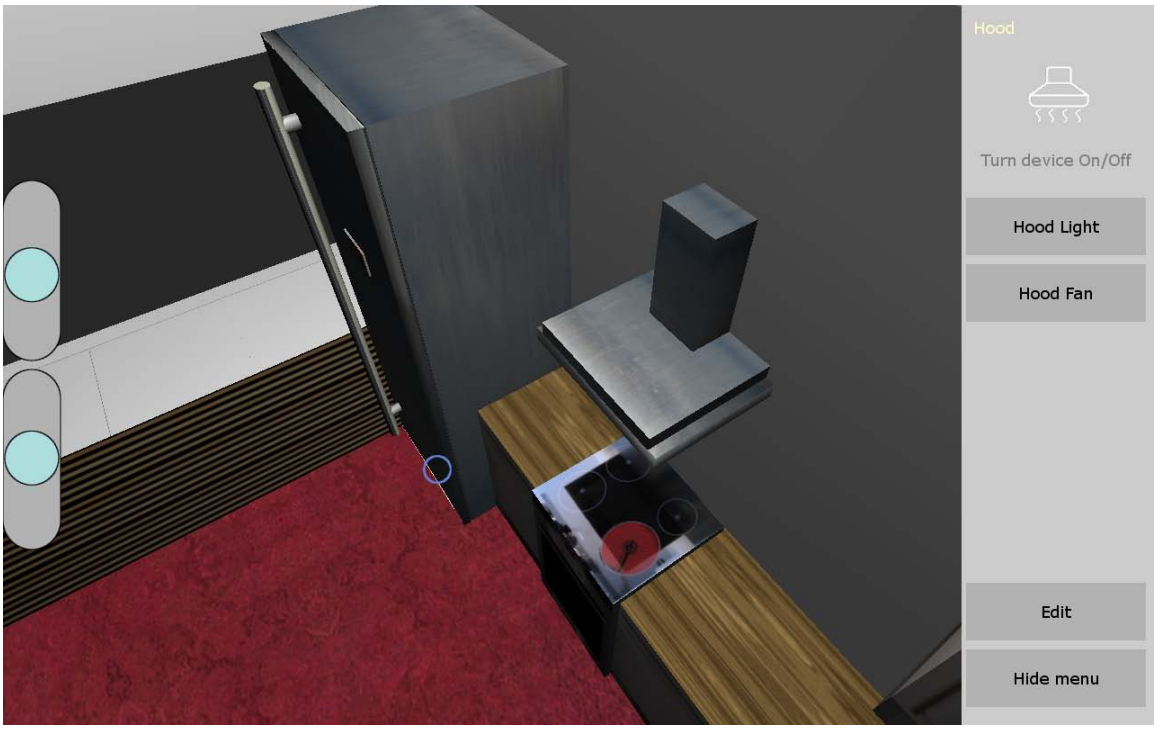

Abbildung 6.2: Die Dunstabzugshaube bündelt eine Leuchte und einen Ventilator, die in einem Auswahlmenü am rechten Fensterrand selektiert werden können.

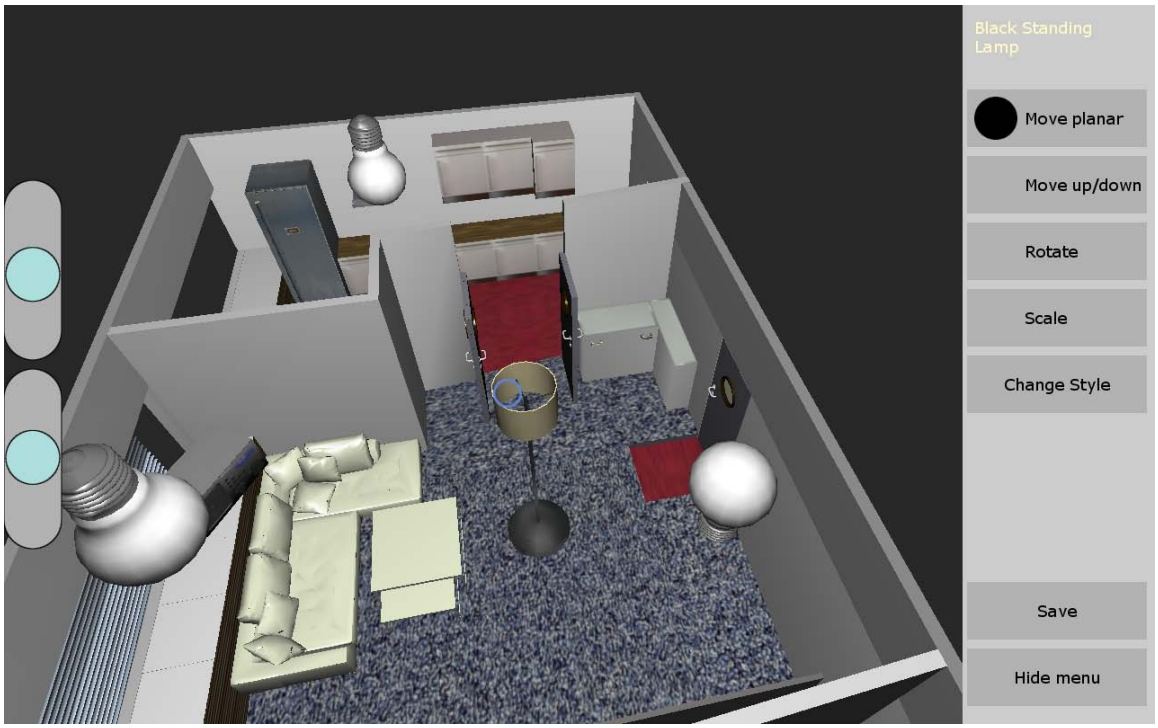

Abbildung 6.3: Die Edit-Ansicht für Geräte.

7 EVALUATIONSSTUDIE

Um die Nutzbarkeit der grafischen Oberfläche zu evaluieren, wurde eine Studie durchgeführt, in der die Smart Home GUI mit einer existierenden, zweidimensionalen Nutzeroberfläche, das Home Operating System (kurz Home OS) (Weingarten, Blumendorf und Albayrak 2010), verglichen wurde. Für die Studie wurde die Geschwindigkeit einfacher Aktionen, wie das An- und Ausschalten von Geräten, für je die dreidimensionale Smart Home GUI und für die zweidimensionale Home OS GUI gemessen. Weiterhin wurde ein Fragebogen (Anhang A - Evaluations-Fragebogen) mit Fragen zur Akzeptanz und Intuitivität der Smart Home GUI ausgewertet. Die nicht repräsentative Studie zeigt eine geringfügig verbesserte Performanz bei Nutzung der Smart Home GUI gegenüber ihrem zweidimensionalen Pendant. Auch wurde mithilfe von Fragebögen die Akzeptanz der 3D GUI belegt. Abschnitt 7.1 beschreibt den Aufbau der Studie, 7.2 ihre Durchführung und Abschnitt 7.3 geht auf ihre Ergebnisse ein.

7.1 Ziel der Studie & Versuchsaufbau

Das Ziel der Studie war, heraus zu finden, ob die Nutzung einer dreidimensionalen Nutzeroberfläche für intelligente Heimumgebungen sich vom ergonomischen Aspekt aus lohnt und vor allem, ob sie Akzeptanz bei den Anwendern findet. Zu diesem Zweck wurde ein Versuch mit Testpersonen in der intelligenten Heimumgebung des Schauraums des DAI-Labors an der Technischen Universität Berlin durchgeführt, sowie jeweils ein Fragebogen mit Fragen betreffend die Akzeptanz der Smart Home GUI beantwortet.

Für den Versuch wurde eine Reihe einfacher Aufgaben in der intelligenten Heimumgebung erdacht, die die Interaktion mit drei verschiedenen Geräten in Schauraum vorsahen. Bei diesen Geräten handelte es sich um Wandleuchten im Wohnzimmer, die Jalousien im Wohnzimmer und den Ventilator der Dunstabzugshaube in der Küche. Die Position der Geräte und der Schauraum selbst sind schematisch in Abbildung 7.1 dargestellt. Betrachtungs- und Eingabemedium war ein Tablet-Computer des Typs Motion J3500 der Firma MotionComputing[9]. Der Rechner war per WLAN mit der MASP (3.1) verbunden und stellte die dreidimensionale Smart Home GUI als Vollbild-Java-Applikation (Abbildung 7.2) sowie die zweidimensionale Home OS GUI im Vollbild-Webbrowser (Abbildung 7.3) dar. Mithilfe der beiden Nutzerschnittstellen sollten nun 6 Teilaufgaben bewältigt werden. Zuerst sollten die Wandleuchten an, dann wieder ausgeschaltet werden, hierauf die Jalousien auf eine beliebige Höhe eingestellt und danach wieder zurück in Ausgangsstellung gebracht werden. Schließlich sollte der Ventilator der Dunstabzugshaube aktiviert und wieder deaktiviert werden. Zwischen den einzelnen Schalt-Aktionen, die von einem Probanden ausgeführt wurden, wurden die beiden Nutzerschnittstellen jeweils wieder in eine Ausgangsposition gebracht. Für die Zeitmessung wurde eine manuell betätigte Stoppuhr genutzt.

[9] http:/www.motioncomputing.de. Abgerufen am 30.05.2012.

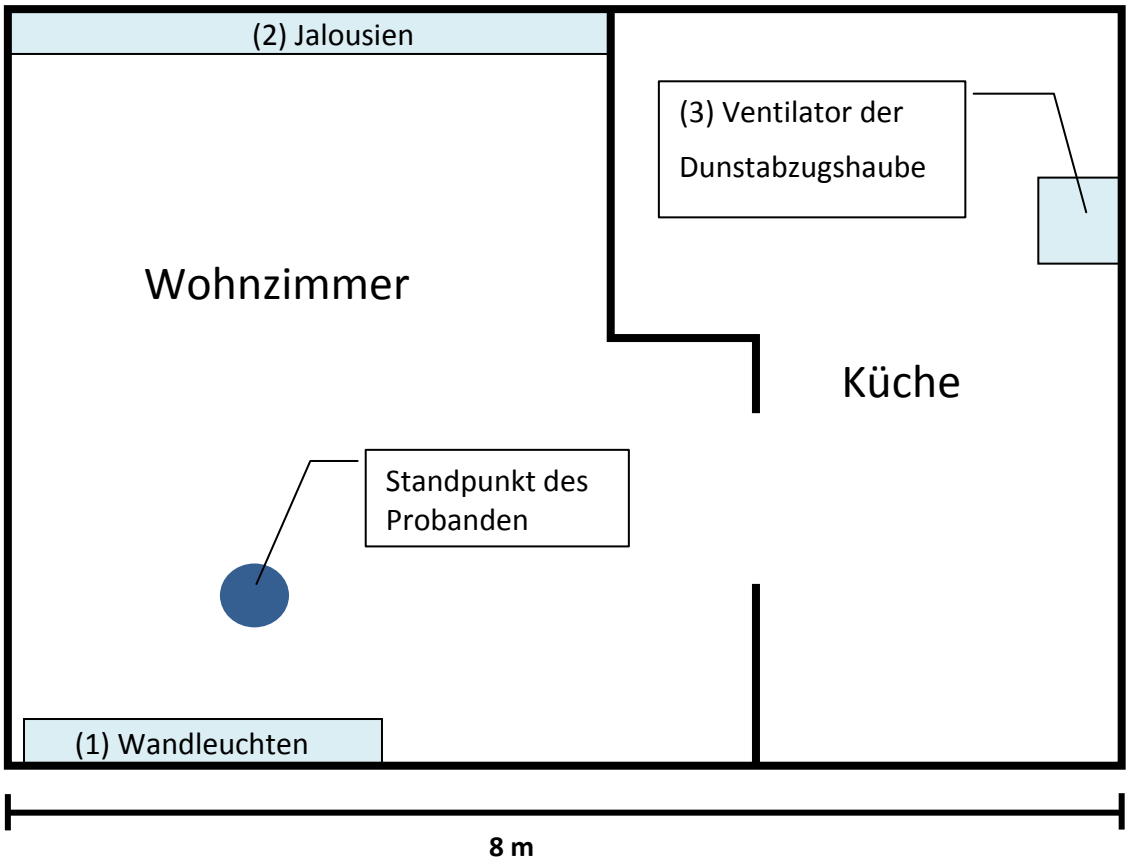

Abbildung 7.1: Schematischer Grundriss des Schauraums sowie die drei in der Studie bedienten Geräte, nummeriert in der Reihenfolge ihrer Benutzung. Der dunkelblaue Punkt markiert den ungefähren Standpunkt der einzelnen Probanden während der Studie.

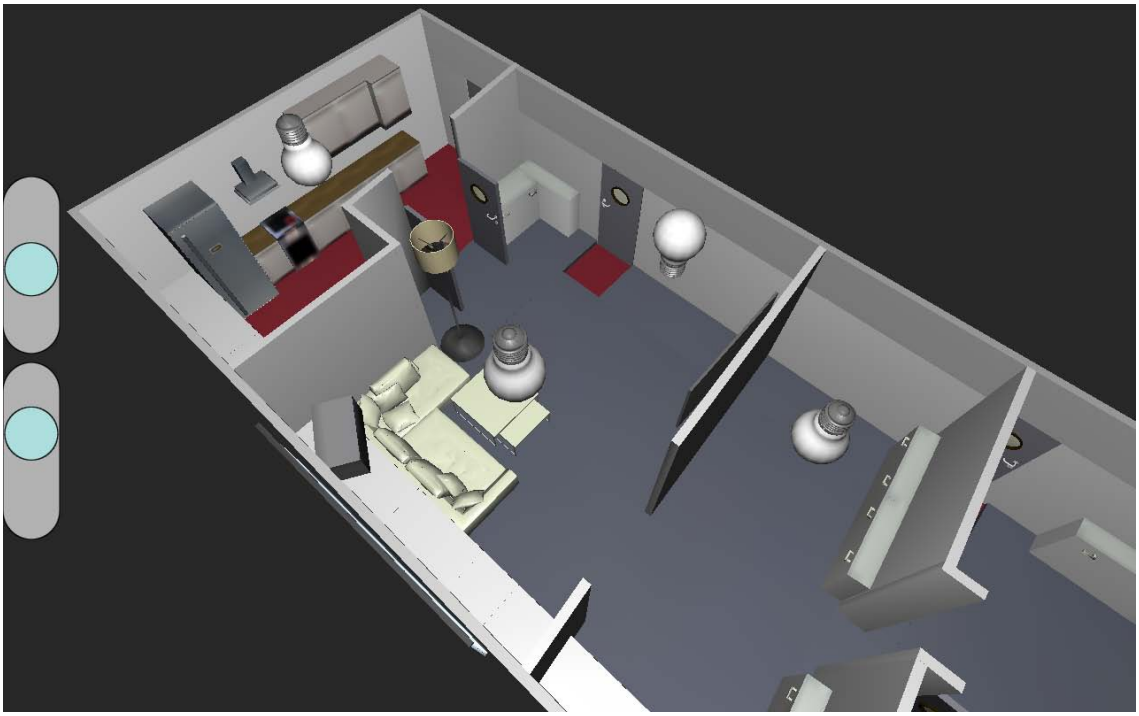

Abbildung 7.2: Smart Home GUI in der Ausgangssituation während der Studie.

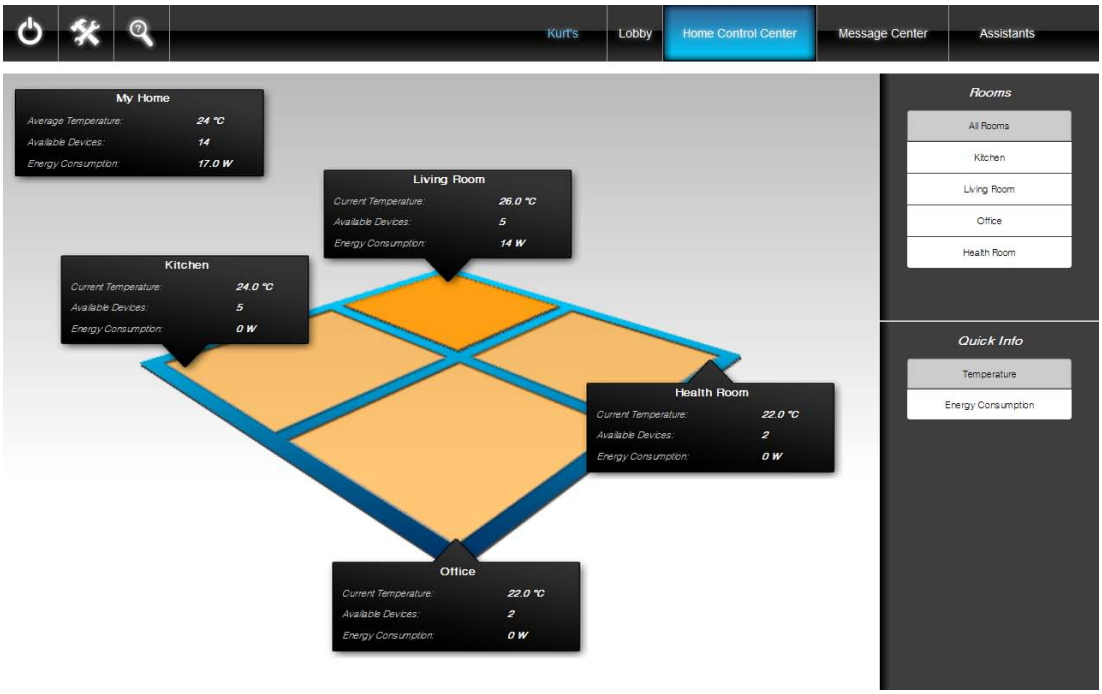

Abbildung 7.3: Zweidimensionale Home OS GUI in der Ausgangssituation während der Studie.

7.2 Probandenwahl und Durchführung

Zehn Personen im Alter zwischen 23 und 32 Jahren dienten als Probanden. Insgesamt betrug das Durchschnittsalter 27,5 Jahre. Unter den Versuchsteilnehmern befanden sich zwei Frauen und acht Männer. Da die Probanden in Bezug auf Alter und Geschlecht nicht entsprechend der demografischen Verteilung Deutschlands ausgewählt wurden, ist die Studie selbst nicht repräsentativ. Acht Personen studierten Informatik, eine Person studierte theoretische Informatik und eine Teilnehmerin war Studentin der Biologie. Vier der Probanden hatten keine bis wenig Erfahrung mit Tablet-Computern, sechs Teilnehmer hatten bereits Erfahrung mit solchen Geräten gesammelt. Keine der Testpersonen hatte Erfahrung mit einer der beiden verwendeten Nutzerschnittstellen. Die Probanden hatten etwa gleich verteilt keine, wenig und gute Kenntnis über den Schauraum und seine Einrichtung.

Die Probanden wurden in zwei Gruppen à fünf Personen aufgeteilt. Versuchsteilnehmer der Gruppe 1 steuerten die Geräte der intelligenten Heimumgebung zuerst mithilfe der dreidimensionalen Smart Home GUI und danach mithilfe der zweidimensionalen Home OS GUI. Die Probanden der Gruppe 2 bedienten zuerst die zweidimensionale GUI und darauffolgend die dreidimensionale Variante. Die Durchschnittsalter beider Gruppen der Probanden beider Gruppen waren etwa identisch. Nach einer kurzen Einweisung eines jeden Probanden, ohne, dass ihnen die grafischen Oberflächen selbst näher erläutert wurden, wurden ihnen die Schalt-Aufgaben Schritt für Schritt aufgetragen. Es wurden keine Hilfestellungen, außer bei essentiellen Fragen, gegeben. Sobald eine Startmeldung zu vernehmen war, durfte der Proband mit der Bewältigung einer Aufgabe beginnen und die Zeitmessung begann. Nach dem Bedienen der richtigen Komponenten in den GUIs wurde die Zeit gestoppt. Die Reihenfolge, in der die Geräte zu bedienen waren, war im-

mer die gleiche. Zuerst sollten die Wandleuchten an, dann wieder aus geschaltet werden, hierauf die Jalousien auf eine beliebige Höhe eingestellt und danach wieder zurück in Ausgangsstellung gebracht werden. Schließlich sollte der Ventilator der Dunstabzugshaube aktiviert und wieder deaktiviert werden. Zwischen jeder Schaltaktion wurden beide grafische Nutzeroberflächen auf einen Initialzustand zurück befördert. In der Oberfläche des Home OS wurde die Ansicht zurück auf eine Wohnungs-Übersicht geschaltet, in der dreidimensionalen GUI wurde die Kamera auf eine ursprüngliche Übersicht über die gesamte Wohnung zurückgesetzt und sämtliche offenen Menüs wurden geschlossen. Nach Beendigung aller Aufgaben wurde den Versuchsteilnehmern ein Fragebogen ausgehändigt, der folgende 10 Fragen an die Teilnehmer stellte:

(1) Fanden sie die Steuerung mit der 3D-GUI intuitiv?
(2) Fanden sie die Steuerung mit der 3D-GUI schnell?
(3) Finden sie die Art der Steuerung der 3D-GUI leicht erlernbar?
(4) Würden sie diese Art der Steuerung anderen Eingabemethoden gegenüber bevorzugen?
(5) Würden sie die 3D-GUI Steuerung auch zu Hause benutzen?
(6) Welche Steuerung empfanden sie als schneller?
(7) Welche Steuerung empfanden sie als einfacher?
(8) Welche Steuerung empfinden sie als einfacher zu erlernen?
(9) Welche Steuerung fanden sie intuitiver?
(10) Welche Steuerung würden sie bevorzugen?

Die Fragen konnten durch einfache Auswahl einer aus je 5 Antwortmöglichkeiten beantwortet werden. Für die Fragen eins bis fünf stand hierbei eine gewichtete Wahl zwischen "ja" und "nein" zur Auswahl, für die Fragen fünf bis zehn gaben je fünf gewichtete Möglichkeiten zwischen "3D GUI" und "2D GUI" die Auswahlpalette vor. Des Weiteren ließ der Fragebogen Platz für textuelle Kommentare. Der komplette Fragebogen lässt sich im Anhang A - Evaluations-Fragebogen wieder finden.

7.3 Ergebnisse

Nur zwei der zehn Versuchsteilnehmer lösten die 6 Teilaufgaben in der Summe mit der zweidimensionalen Home OS Nutzerschnittstelle um jeweils weniger als eine halbe Sekunde schneller als mit der Smart Home GUI. Beide Probanden stammten aus Gruppe 1, die den Test zuerst mit der 3D-Nutzeroberfläche vollzog. Die anderen acht Probanden lösten die Aufgaben mithilfe der Smart Home GUI insgesamt durchschnittlich um mehrere Sekunden schneller. Gruppe 1, welche den Test zuerst mit der dreidimensionalen Nutzeroberfläche vollführte, löste die Aufgaben mithilfe der 3D GUI im Durchschnitt um 2 Sekunden schneller, im Median um 0,22 Sekunden marginal schneller. Gruppe 2 hingegen löste die Aufgaben mit der 3D GUI durchschnittlich sogar um 5,31 Sekunden schneller, im Median um 4,43 Sekunden schneller. Im Durchschnitt über alle Probanden und Teilaufgaben wurden die Aufgaben mit der Smart Home GUI um je 3,66 Sekunden schneller gelöst als mit der Home OS Oberfläche, im Median um 2,89 Sekunden schneller. Tabelle 7.1 gibt einen näheren Überblick über die gemittelten Zeiten zur Bewältigung der Teilaufgaben. Deutlich wird ein großer Zeitunterschied zwischen der 2D GUI und 3D GUI bei der ersten Aktion „Wandleuchten An“. Bei dieser Aktion waren die Probanden

mit der Smart Home GUI im Mittel um 16,6 Sekunden schneller als mit Home OS GUI. Es ist zu beachten, dass durch die manuellen Zeitmessungen Fehler im Sekundenbruchteilbereich nicht zu vermeiden waren.

Die Ergebnisse des Fragebogens besagen, dass die Smart Home GUI als sehr intuitiv, leicht erlernbar und schnell betrachtet wird. Die Mehrheit der Probanden würde die Smart Home GUI einer anderen Art der Steuerung gegenüber bevorzugen. Auf die Frage hin, ob die Testperson eine solche Art der Steuerung bei sich zu Haus nutzen würden, antworteten drei Personen entschieden mit nein, die Mehrheit antwortete jedoch tendenziell mit ja. Im Vergleich mit der zweidimensionalen Home OS Schnittstelle wurde die Smart Home GUI vorwiegend als schneller und einfacher bewertet. Alle Versuchsteilnehmer gaben an, die Smart Home GUI gegenüber der Home OS Schnittstelle zu bevorzugen. Die Anzahl der gewählten Antworten der Probanden können Tabelle 7.2 und Tabelle 7.3 entnommen werden.

Zwei der Probanden bemängelten, dass die Repräsentation der Wandleuchten in der Smart Home GUI nicht dem realen Bild der Leuchten entsprach, sondern diese durch eine generische Glühbirne dargestellt wurden. Dies sorgte nach Aussage der Versuchspersonen für Verwirrung, ob es sich dabei tatsächlich um die Wandleuchten handele. Ihrer Aussage nach hätte eine visuelle Imitation der echten Leuchten effizienteres Handeln ermöglicht. Eine Person merkte an, dass sie bei der Multitouch-Steuerung für die Jalousien annahm, diese direkt durch eine Wisch-Geste, ohne die Nutzung von Widgets, bewegen zu können. Dies ist in der aktuellen Implementierung der Smart Home GUI nicht möglich. Zwei Probanden schlugen vor, in der Smart Home GUI angewählte Objekte farblich hervorzuheben. Eine weitere Person empfahl eine automatische Kamerafokussierung auf ein gerade ausgewähltes Gerät.

7.4 Interpretation der Ergebnisse

Eine Auffälligkeit ist die erhöhte zeitliche Leistungssteigerung bezüglich der 3D GUI bei Gruppe 2 gegenüber Gruppe 1 (siehe Tabelle 7.1). Dies ist möglicherweise darauf zurück zu führen, dass Versuchspersonen nach der eingängigen Nutzung der zweidimensionalen GUI mit den Positionen und Geräten der Heimumgebung bereits vertraut waren. Die Probanden wussten somit, wo die zu bedienenden Geräte in der Heimumgebung zu finden waren. Eine solche Leistungssteigerung prägt sich bezüglich der zweidimensionalen GUI bei Gruppe 1 weniger stark aus, da die zweidimensionale Home OS-Nutzerschnittstelle die Positionen der Geräte stark abstrahiert; die Geräte werden in Form einer Liste repräsentiert, was deren wahre Positionen im Raum weitgehend irrelevant erscheinen lässt.

Eine weitere Beobachtung ist die erhöhte Aktionsgeschwindigkeiten bei der Aufgabe (1) „Wandleuchten An“. Dies lässt sich daher herleiten, dass Aufgabe (1) die erste auszuführende Aufgabe war. Die langen Durchführungszeiten in der zweidimensionalen GUI lassen sich auf anfängliche Verwirrung der Probanden zurück führen, die in den darauf folgenden Aktionen mit zunehmender Kenntnis der Steuerung nachließ. Eine solche anfängliche Verwirrung schien auch bei der Smart Home GUI zu existieren (siehe Tabelle 7.1), jedoch nicht in so ausgeprägtem Maße wie bei der zweidimensionalen Oberfläche.

Generell lassen sich bei den Aktionen (1), (3) und (5), bei denen das erste Mal mit einem neuen Gerät interagiert wurde, große zeitliche Unterschiede zwischen 3D und 2D GUI feststellen. Der exakte durchschnittliche Differenzwert über alle Probanden beträgt hier 6,22 Sekunden, der für die Aufgaben (2), (4), (6) lediglich 1,11 Sekunden. In beiden Fällen ist jedoch die Nutzung der Smart Home GUI effizienter gewesen. Daraus kann man ableiten, dass die 3D-Repräsentation durch ihre visuelle Ähnlichkeit mit der realen Welt besonders beim Einstieg in die Bedienung hilft, wohingegen die Einarbeitungszeit in die Home OS GUI durch abstrakte Grafiken und Texte negativ beeinflusst wird. Offen bleibt die Frage, welche GUI nach einer Langzeitnutzung die zeitlich effektivere ist.

Auffällig ist auch die im Mittel und Median schnellere Steuerung der Jalousien mithilfe der Home OS GUI. Dies kann mehrere Ursachen haben. Während in der Home OS GUI die Steuerung der Jalousien mittels vier Buttons für vier verschiedene Höhen bewerkstelligt wurde, wurde in der Smart Home GUI ein Slider zur Höhenverstellung gebraucht. Der bei allen Gerätetypen zu findende An/Aus-Schalter wurde bei den Jalousien deaktiviert und als deaktiviert dargestellt. Da die Probanden jedoch nicht mit dem verwendeten Farbschema der GUI vertraut waren, wussten sie nicht, wie ein deaktiviertes GUI-Widget aussieht. Einige Testpersonen versuchten, den deaktivierten An/Aus-Schalter zu bedienen und stellten erst nach etwas Zeit fest, dass dieser deaktiviert war. Abbildung 7.4 zeigt einen Button im Modus *Normal* und einen Button im Modus *Disabled*. Ein zweiter Grund für die geringe Effizienz der Steuerung der Jalousien war, dass einige Nutzer die Jalousien nicht auf Anhieb auswählen konnten, weil ihnen das Bewusstsein für die Kamera-Neigung mithilfe des Sliders im linken Teil des GUI-Fensters fehlte. Da die Probanden den Slider nicht nutzten, versuchten sie, die Jalousie ohne Veränderung der Kameraneigung, aus einem ungünstigen Blickwinkel, auszuwählen, was in mehreren Versuchen nicht funktionierte. Während man in der Smart Home GUI optimalerweise ein neues Steuerelement (den Neigungs-Slider) hätte nutzen sollen, war dies in der Home OS GUI nicht der Fall, die Steuerung war weitgehend ähnlich zur Steuerung in Aufgabe (1) und (2). Entsprechendes Vorwissen über die Smart Home GUI hätte die Aktionen (3) und (4) fraglos beschleunigt, war jedoch nicht in der Studie erwünscht.

Die nichtrepräsentative Evaluationsstudie sowie die Antworten zu den 10 Fragen zeigen insgesamt eine hohe Akzeptanzbereitschaft für eine dreidimensionale grafische Nutzeroberfläche im Kontext der intelligenten Heimumgebung. Beide Versuchsgruppen erreichten mit der Smart Home GUI sowohl im arithmetischen Mittel als auch im Median höhere Aktionsgeschwindigkeiten als mit der zweidimensionalen Home OS-Nutzerschnittstelle. Demnach kann neben einer subjektiven Effizienzsteigerung im Sinne der Einfachheit auch von einer effektiven Leistungssteigerung im Sinne des Zeitaufwands erkannt werden. Beide Erkenntnisse lassen sich dadurch erklären, dass die dreidimensionale Repräsentation der intelligenten Heimumgebung wie in Kapitel 1 behauptet eher dem mentalen Modell eines Nutzers entspricht als dies eine zweidimensionale Nutzerschnittstelle vermag. Der von den Probanden zwei Mal angesprochene Wunsch nach realitätsnäheren Visualisierungen der Wandlampe legt ebenfalls nah, dass das nutzerseitige Bedürfnis nach einer weniger abstrakten Interaktionsschnittstelle tatsächlich besteht. Die von drei Personen gewünschte Kenntlichmachung von ausgewählten Geräten zeugt weiterhin vom Wunsch nach verstärkter Eingaberesonanz seitens der Smart Home GUI.

			(1) Wandleuchten An	(2) Wandleuchten Aus	(3) Jalousien Runter	(4) Jalousien Hoch	(5) Lüfter Dunstabzugshaube An	(6) Lüfter Dunstabzugshaube Aus	Ø
Gruppe 1 - 3D zuerst	Ø	2D	28,42s	5,59s	11,16s	6,42s	15,96s	5,76s	**12,22s**
		3D	16,58s	2,54s	19,16s	8,38s	10,32s	4,32s	**10,22s**
	Median	2D	27,4s	5,79s	10,7s	5,5s	8,7s	3,9s	**10,33s**
		3D	17s	2s	17,3s	9s	11,3s	4,1s	**10,12s**
Gruppe 2 - 2D zuerst	Ø	2D	27,96s	5,38s	9,66s	5,6s	9,44s	6,24s	**10,71s**
		3D	6,58s	2,82s	7,44s	5,96s	5,29s	4,32s	**5,4s**
	Median	2D	20,9s	5,4s	9,2s	5,4s	8,6s	6,9s	**9,4s**
		3D	4,3s	2,9s	7,7s	6,7s	4,2s	4s	**4,97s**
Gruppe 1 und Gruppe 2	Ø	2D	28,19s	5,49s	10,41s	6,01s	12,7s	6s	**11,47s**
		3D	11,58s	2,68s	13,3s	7,17s	7,81s	4,32s	**7,81s**
	Median	2D	24,15s	5,6s	9,55s	5,45s	8,65s	5,45s	**9,81s**
		3D	10,65s	2,5s	9,85s	7,1s	7,38s	4,05s	**6,92s**

Tabelle 7.1: Das arithmetische Mittel und der Median der Zeiten für die Lösung der einzelnen Aufgaben der Evaluationsstudie, für die einzelnen Teilaufgaben (horizontal) und Gruppen (vertikal). Die vertikale Unterteilung wurde weiter in arithmetisches Mittel und Median, jeweils für die 3D Nutzeroberfläche (Smart Home GUI) und die 2D Nutzeroberfläche (Home OS) gegliedert. Rot hinterlegte Felder markieren höheren Zeitaufwand gegenüber der vergleichbaren Zelle der jeweils anderen GUI-Variante. Grün hinterlegte Felder markieren geringeren Zeitaufwand. In der rechten Spalte sind die arithmetischen Mittel der Werte jeder Teilaufgabe zu finden. Rot umrandete Felder markieren hier noch einmal geringeren Zeitaufwand. Man kann sehen, dass im Durchschnitt über alle Teilaufgaben die Smart Home GUI immer schneller als ihr zweidimensionales Pendant, die Home OS GUI, ist.

#	Frage	ja				nein
1.	Fanden sie die Steuerung mit der 3D-GUI intuitiv?	8	1	1	0	0
2.	Fanden sie die Steuerung mit der 3D-GUI schnell?	6	3	1	0	0
3.	Finden sie die Art der Steuerung der 3D-GUI leicht erlernbar?	10	0	0	0	0
4.	Würden sie diese Art der Steuerung anderen Eingabemethoden gegenüber bevorzugen?	5	2	2	0	1
5.	Würden sie die 3D-GUI Steuerung auch zu Hause benutzen?	4	3	0	0	3

Tabelle 7.2: Die Anzahl der gewählten Antwortmöglichkeiten zu den Fragen 1 bis 5.

#	Frage	3D				2D
6.	Welche Steuerung empfanden sie als schneller?	6	3	1	0	0
7.	Welche Steuerung empfanden sie als einfacher?	6	4	0	0	0
8.	Welche Steuerung empfinden sie als einfacher zu erlernen?	5	2	2	1	0
9.	Welche Steuerung fanden sie intuitiver?	7	3	0	0	0
10.	Welche Steuerung würden sie bevorzugen?	7	3	0	0	0

Tabelle 7.3: Die Anzahl der gewählten Antwortmöglichkeiten zu den Fragen 6 bis 10.

Abbildung 7.4: Ein aktivierter Button (links) und ein deaktivierter Button (rechts) im in der Studie verwendeten Designschema. Das helle Grau an den Rändern ist die Hintergrundfarbe von Dialogen im verwendeten Designschema.

8 ZUSAMMENFASSUNG & SCHLUSS

Im Zuge dieser Arbeit wurde die Vermutung aufgestellt, dreidimensionale grafische Multitouch-Nutzerschnittstellen seien zweidimensionalen GUIs im Kontext der intelligenten Heimumgebung in Hinblick auf Nutzerfreundlichkeit und Effizienz überlegen. Auf diesem Gedanken fußend wurde eine solche dreidimensionale Nutzerschnittstelle, die Smart Home GUI, entworfen und implementiert. Dafür wurden zunächst verwandte Arbeiten und relevante Forschungen, wie die Multitouch-Interaktion mit 3D-Objekten, näher betrachtet und teils mit der geplanten Arbeit in Relation gesetzt. Nach einem Auswahlprozess von Technologien, die bei der Entwicklung der Smart Home GUI behilflich sein konnten, wurde zunächst ein mehr oder minder generisches GUI-Framework aufbauend auf dem Multitouch-Framework MT4j implementiert und dokumentiert. Dieses erweitert MT4j um Widgets und führt wie geplant für reichhaltige grafische Oberflächen wichtiges und dynamisches Ressourcenmanagement ein. Die ausgiebige Verwendung von Entwurfsmustern zum Segen der Wiederverwendbarkeit stand bei der Architektur im Vordergrund und wurde dementsprechend realisiert. Zudem gelang durch die Implementierung des GUI Frameworks, wie angestrebt, eine möglichst starke Trennung von Logik und Design. Schließlich wurde die Smart Home GUI selbst implementiert. Mit ihrer Hilfe kann der Nutzer nun eine intelligente Heimumgebung virtuell begutachten und mithilfe der Software MASP mit Haushaltsgeräten interagieren. In einer Evaluationsstudie mit 10 ungeübten Probanden wurde die Smart Home GUI mit der zweidimensionalen Home OS GUI in Hinblick auf Effizienz und Anwenderzufriedenheit verglichen. Hierbei konnte die Smart Home GUI durchweg positive Resonanz erfahren. Auch die Geschwindigkeit einfacher Aktionen war im arithmetischen Mittel circa drei Sekunden schneller als die Geschwindigkeit äquivalenter Aktionen mithilfe der Home OS GUI. Die positiven subjektiven Bewertungen der Probanden sowie die niedrige Ausführungszeit von Befehlen bestätigen die eingangs vorgebrachte Vermutung. Jedoch, die Evaluationsstudie lässt zwar Rückschlüsse eine hohe Nutzerakzeptanz bezüglich der Smart Home GUI zu und zeigt auch eine hohe Effizienz bei ungeübten Nutzern, dennoch bietet sie keinen Vergleich bei geübten, eingearbeiteten Nutzern, sodass hierfür keine Erkenntnisse gewonnen wurden. Die Studie offenbarte auch Fehlbarkeiten der Smart Home GUI. So hat sich das farbliche Designschema der GUI als nicht optimal herausgestellt, doch ist dies durch die umgesetzte Abstraktion von Design und Logik schnell änderbar. Weiterhin bemängelten einige Probanden die teils zu geringe grafische Ähnlichkeit der Geräte mit ihren 3D-Repräsentationen. Es wurden Verbesserungsvorschläge gemacht, die das Kenntlichmachen von fokussierten Geräten empfahlen. Trotz der Schwachstellen konnte die GUI insgesamt ein hohes Maß an Akzeptanz erreichen und zeitgleich Effizienz beweisen. Schlussendlich konnte erfolgreich sichtbar gemacht werden, dass die eine dreidimensionale GUI für intelligente Heimumgebungen eine mächtige Alternative gegenüber konventionellen zweidimensionalen GUIs darstellt.

9 AUSBLICK

In Zukunft können weitere Studien durchgeführt werden, mit deren Hilfe die Effizienz der Smart Home GUI gegenüber zweidimensionalen Alternativen auch über eine gewisse Dauer von mehr als einer Sitzung verglichen werden kann. Auf diese Weise kann man einen Vergleich der beiden Ansätze bei routinierten Anwendern ziehen, denn die Frage der Effizienz einer GUI hängt stark von den Gewohnheiten des Nutzers ab. Die im Rahmen dieser Arbeit erstellte Software selbst lässt ebenfalls Freiraum für Weiterentwicklung. Das GUI Framework kann mit neuen Widgets und standardisierten Szenen bestückt werden, um noch schneller eine beliebige multitouchfähige 3D-Anwendung zu realisieren. Auch können neue, ansprechende und ergonomische Designs für die Widgets entworfen werden. Eine weitere Idee ist, das 3D-Modell der Heimumgebung nicht explizit im 3D-Modellierungsprogramm zu erstellen, sondern dies implizit mithilfe der MASP zu generieren; mithilfe der MASP-Schnittstelle können Positionen und Ausdehnungen von Räumen erfasst werden. Diese Informationen können wiederum verwendet werden, um zur Laufzeit ein automatisch generiertes 3D-Modell für jede beliebige Heimumgebung darzustellen. Schließlich kann auch Smart Home GUI einigen Verbesserungen unterzogen werden. Etwa kann man die von einigen Probanden in der Evaluationsstudie gewünschte Markierung des fokussierten Gerätes implementieren. Auch können neue Dialogmenüs, angepasst auf verschiedene Zielplattformen, wie Multitouch-Tische und Mobilgeräte mit kleinen Displays, erstellt werden. Mit all diesen Ideen und den verschiedensten Varianten für mögliche Designs sind die Möglichkeiten zur Weiterentwicklung der Software schließlich und endlich schier grenzenlos.

10 DANKSAGUNG

An erster Stelle möchte ich meinem Betreuer Mathias Wilhelm für seine Unterstützung und Geduld danken, die er mir für die gesamte Zeit der Arbeit zuteilwerden ließ. Weiterhin möchte ich Prof. Dr. Dr. Sahin Albayrak für seinen starken Glauben in meine Arbeit und seine wertvolle Motivation meinen Dank aussprechen. Auch danke ich Bianca Scheibel für ihre Zurverfügungstellung eines 3D-Modells des Schauraums sowie Martin Zöllig für das Gegenlesen und Beurteilen der Arbeit. Für ihre Zeit und Meinung möchte ich zu guter Letzt auch den zehn Teilnehmern der Evaluationsstudie mein Lob spenden.

11 REFERENZEN

Apache Software Foundation. *Maven Getting Started Guide.* 12 30, 2010. http://maven.apache.org/guides/getting-started/index.html (accessed Januar 8, 2011).

Blender Foundation. *blender.org - Home.* Blender Foundation. 2012. http://www.blender.org/ (Zugriff am 13. 05 2012).

Borodulkin, L., H. Ruser, und H.-R. Tränkler. *3D Virtual "Smart Home" User Interface.* Neubiberg, Germany: Universität der Bundeswehr München, 2002.

Breier, Florian. *Multitouch 3D Interaktion mit 3D Objekten.* Stuttgart, Mai 2010.

Echtler, Florian. *TISCH - Tangible Interactive Surfaces for Collaboration between Humans.* 2010. http://tisch.sourceforge.net/ (Zugriff am 28. Mai 2012).

Fraunhofer IAO. *Architecture - MT4j.* 2011. http://www.mt4j.org/mediawiki/index.php/Architecture (Zugriff am 24. 05 2012).

Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO. *MT4j - Multitouch For Java.* 2010. http://www.mt4j.org/mediawiki/index.php/Main_Page (Zugriff am 14. Januar 2011).

Gamma, E., R. Helm, R. Johnson, und J. Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software.* Indianapolis: Addison-Wesley, 1994.

Gira GmbH & Co. KG. *Das KNX/EIB System - Intelligente Haustechnik.* 2012. http://www.gira.de/gebaeudetechnik/systeme/knx-eib_system.html (Zugriff am 18. 05 2012).

Google Inc. ***Google SketchUp.*** Google Inc. 2012. http://sketchup.google.com/intl/de/ (Zugriff am 13. 05 2012).

Knoedel, Sebastian, und Martin Hachet. *Technote à IEEE 3DUI 2010.* 9. Januar 2011. http://anr-instinct.cap-sciences.net/?q=node/41 (Zugriff am 13. März 2011).

Langenhagen, Andreas. *End-User Development mit Fokus auf grafische Nutzeroberflächen.* Berlin: DAI-Labor, 2010.

Laufs, Uwe, Christopher Ruff, und Jan Zibuschka. „MT4j – A Cross-platform Multi-touch Development." *ACM EICS 2010, Workshop: Engineering patterns for multi-touch interfaces*, Dezember 2010: 52-57.

Martinet, Anthony. *Anthony Martinet's Home Page.* 7. 3 2011. http://www.lifl.fr/~martinea/ (Zugriff am 13. 3 2011).

Martinet, Anthony, Géry Casiez, und Laurent Grisoni. „The Effect of DOF Separation in 3D Manipulation Tasks with Multi-touch Displays." In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, von George Baciu, Rynson Lau, Ming Lin, Taku Komura und Qunsheng Peng, 111-118. Hong Kong: ACM, 2010.

Mindstorm Limited. *breezemultitouch.* 17. März 2010. http://code.google.com/p/breezemultitouch/ (Zugriff am 28. Mai 2012).

Roscher, Dirk, Grzegorz Lehmann, Veit Schwartze, Marco Blumendorf, und Sahin Albayrak. „Dynamic Distribution and Layouting of Model-Based User Interfaces." In

Model-Driven Development of Advanced User Interfaces, von Heinrich Hussmann, Gerrit Meixner und Detlef Zuehlke, 171-197. Berlin/Heidelberg: Springer, 2011.

Seifried, Thomas, et al. „CRISTAL: A Collaborative Home Media and Device Controller Based on a Multi-touch Display.“ In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, 33-40. New York: ACM, 2009.

Smart, John Ferguson. *An introduction to Maven 2.* 05. December 2005. http://www.javaworld.com/javaworld/jw-12-2005/jw-1205-maven.html (Zugriff am 08. January 2011).

Weingarten, Florian, Marco Blumendorf, und Sahin Albayrak. „Towards multimodal interaction in smart home environments: the home operating system.“ In *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, 430-433. New York: ACM, 2010.

Wings3D. *Wings 3d.* 2010. http://www.wings3d.com/ (Zugriff am 17. Januar 2011).

12 ABBILDUNGSVERZEICHNIS

13 TABELLENVERZEICHNIS

14 ANHANG A - EVALUATIONS-FRAGEBOGEN

Fragebogen zur multimodalen Steuerung intelligenter Heimumgebungen

Geschlecht: m ☐ w ☐

Alter:.....

Studiengang:.....

		ja				nein
1.	Fanden sie die Steuerung mit der 3D-GUI intuitiv?	☐	☐	☐	☐	☐
2.	Fanden sie die Steuerung mit der 3D-GUI schnell?	☐	☐	☐	☐	☐
3.	Finden sie die Art der Steuerung der 3D-GUI leicht erlernbar?	☐	☐	☐	☐	☐
4.	Würden sie diese Art der Steuerung anderen Eingabemethoden gegenüber bevorzugen?	☐	☐	☐	☐	☐
5.	Würden sie die 3D-GUI Steuerung auch zu Hause benutzen?	☐	☐	☐	☐	☐
		3D GUI				**2D GUI**
6.	Welche Steuerung empfanden sie als schneller?	☐	☐	☐	☐	☐
7.	Welche Steuerung empfanden sie als einfacher?	☐	☐	☐	☐	☐
8.	Welche Steuerung empfinden sie als einfacher zu erlernen?	☐	☐	☐	☐	☐
9.	Welche Steuerung fanden sie intuitiver?	☐	☐	☐	☐	☐
10.	Welche Steuerung würden sie bevorzugen?	☐	☐	☐	☐	☐

11. Anmerkungen