Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Philipp Roßberger

Physikbasierte Interaktion in kollaborativen computergestützen Umgebungen

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Philipp Roßberger

Physikbasierte Interaktion in kollaborativen computergestützen Umgebungen

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Master Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Kai von Luck
Zweitgutachter : Prof. Dr. Gunter Klemke

Abgegeben am 21. Januar 2008

Philipp Roßberger

Thema der Bachelorarbeit

Physikbasierte Interaktion in kollaborativen computergestützen Umgebungen

Stichworte

Physikbasierte Interaktion, Collaborative Workspace, Ubiquitous Computing, Ambient Intelligence, Tabletop

Kurzzusammenfassung

Bei gemeinschaftlicher Arbeit an Tischflächen nutzen Menschen häufig bestimmte Arbeitstechniken: das Arbeitsmaterial wird hin- und hergeschoben, gedreht und in Form von Stapeln sortiert. Im Rahmen dieser Masterarbeit wurden diese Interaktionstechniken in Form eines Programm namens DynAmbient mit Hilfe physikalischer Simulationsalgorithmen nachgebildet. Durch Modellierung physikalischer Eigenschaften, wie Masse und Oberflächenbeschaffenheit, verhalten sich digitale Objekte in *DynAmbient* wie es Anwender in der Realität erwarten würden. Dieses neuartige Bedienkonzept *physikbasierter Interaktion* erlaubt die Nutzung vertrauter Arbeitstechniken und verbessert somit die intuitive Bedienbarkeit von Programmen in kollaborativen computergestützen Umgebungen.

Philipp Rossberger

Title of the paper

Physics-based interaction in collaborative computer supported workspaces

Keywords

Physics-based interaction, Collaborative Workspace, Ubiquitous Computing, Ambient Intelligence, Tabletop

Abstract

During cooperative work on tabletops people frequently use certain working techniques: objects are moved around, rotated and sorted by stacks. By using physical simulation algorithms these interaction techniques were recreated in a computer program called *DynAmbient*. Through the modelling of physical properties like mass and surface chararcteristics, digital objects in DynAmbient behave as users would expect them to do in reality. This novel concept of *physics-based interaction* allows the usage of familiar working techniques and consequently improves the usability of programs in collaborative computer supported workspaces.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die von Weiser (1996) prognostizierte Allgegenwart von Computern ist im Begriff stückweise Realität zu werden. Mittlerweile ist der Besitz mehrerer, meist mobiler Computer in Form von Laptops, Smartphones und PDAs, für viele Menschen Normalität. Pierce und Nichols (2007) sprechen von einer Entwicklung weg vom *Personal Computer* hin zu *Personal Information Environments*. Die Informationsumgebung besteht dabei aus den Daten des Anwenders, die über seine heterogene Gerätesammlung verstreut sind.

Analog zu dieser Entwicklung sind in den letzten Jahren gemeinschaftlich genutzte, kollaborative Informationsumgebungen, so genannte *Collaborative Workspaces* ins Forschungsinteresse gerückt. Diese neuartigen Arbeitsräume erlauben die Zusammenarbeit mehrerer Personen unter Nutzung digitaler Dokumente und Informationen.

Ähnlich wie im Falle persönlicher Informationsumgebungen fehlen bisher ausgereifte Konzepte und Anwendungen für Collaborative Workspaces, die eine einfache und bequeme Nutzung verschiedener Daten über mehrere Geräte und Programme hinweg erlauben. Eine besondere Herausforderung stellt dabei die Entwicklung intuititver Interaktionstechniken dar, die eine effiziente Zusammenarbeit mehrerer gleichzeitig agierender Personen erlauben.

Bei gemeinschaftlicher Arbeit mit Papierdokumenten und Fotos auf Tischflächen nutzen Menschen bestimmte Arbeitstechniken sehr häufig: das Arbeitsmaterial wird anderen Personen übergeben, hin- und hergeschoben, von verschiedenen Seiten betrachtet und in Form von Stapeln sortiert. Die Übertragung solcher Handlungsweisen auf die Bedienung von Computer-Programmen, erscheint als viel versprechender Ansatz, da diese nachgebildeten Interaktionstechniken von Menschen als natürlich empfunden werden, weil sie aus der Realität vertraut sind.

Der Umgang mit Gegenständen in der Realität unterliegt physikalischen Gesetzen. Jedes Objekt hat eine individuelle Größe, Masse, Dichte und Oberflächenbeschaffenheit. Menschen lernen im Laufe ihres Lebens, wie sich Objekte mit bestimmten Eigenschaften verhalten. Sie sind es z. B. gewohnt, dass eine Zeitschrift ein gewisses Stück über eine Tischfläche rutscht, wenn sie ausreichend stark angestoßen wird.

Die realitätsnahe Simulation des Verhaltens von Objekten mit physikalischen Eigenschaften unter Krafteinwirkung in Kombination mit der Nachbildung natürlicher Interaktionstechniken bildet die zentralen Bestandteile eines neuartigen Bedienkonzepts mit dem sich diese Masterarbeit unter der Bezeichnung *physikbasierte Interaktion* beschäftigt. Ziel dieser Ausarbeitung ist die Entwicklung einer physikbasierten Computer-Anwendung für Collaborative Workspaces mit dem Namen DynAmbient[1]. Diese Masterarbeit gliedert sich wie im folgenden Abschnitt 1.2 beschrieben.

1.2 Gliederung der Arbeit

Im Anschluss an diesen Abschnitt, werden in Kapitel 2 die Systemarchitektur und Aufbau eines Collaborative Workspaces und unterschiedliche Interaktionskonzepte mit den damit verbundenen Geräten vorgestellt. Danach werden Handlungsweisen betrachtet, die besonders häufig bei menschlicher Zusammenarbeit auf Tischen auftreten und verschiedene Ansätze diskutiert, die es erlauben diese Form der Kollaboration unter dem Einsatz von Computern auszuüben. Schließlich wird die Erstellung von Programmen anhand mentaler Modelle betrachtet. Dieser Abschnitt bezieht sich auf die bereits in 1.1 erläuterte Nachbildung von natürlichen Interaktionstechniken.

Das Analyse-Kapitel (3) dient der Definition bestimmter Funktionalitäten denen DynAmbient genügen soll. Dazu wird zunächst ein Anwendungsszenario beschrieben, aus dem sich Rahmenbedingungen für die Gestaltung von DynAmbient in Form funktionaler und nichtfunktionaler Anforderungen ableiten.

Darauf aufbauend folgt in Kapitel 4 die Entwicklung eines Software-Designs für DynAmbient. Zunächst wird die Bedienoberfläche und grundlegende Funktionalität des Programms erklärt. Danach folgt unter Verwendung geeigneter Software-Entwurfsmuster die Gestaltung der Architektur von DynAmbient. Dabei wird auch diskutiert, wie die Applikation in die Anwendungslandschaft eines Collaborative Workspace integriert werden kann. Im Anschluß wird die gestalterische und technische Umsetzung der Anforderungen vorgestellt und am Ende des Kapitels bewertet.

Der letzte Teil dieser Arbeit, Kapitel 5, enthält eine Zusammenfassung und diskutiert abschliessend verschiedene Möglichkeiten der Weiterentwicklung von DynAmbient.

[1] Die Bezeichnung leitet sich aus den englischen Begriffen *dynamic* und *ambient* ab.

2 Grundlagen

Diese Kapitel beschreibt zunächst in Abschnitt 2.1 die Arbeitsumgebung für die DynAmbient entwickelt wird. Im Anschluss daran wird in 2.2 erläutert, wie Interaktion in einem Collaborative Workspace unter Einsatz spezieller Konzepte und Eingabegeräte erfolgen kann. Die Ausrichtung von Objekten auf horizontalen Oberflächen spielt in diesem Umfeld eine wichtige Rolle und wird daher in 2.3 gesondert betrachtet. Physikbasierte Interaktionstechniken eignen sich in diesem Zusammenhang besonders zur effizienten Interaktion mit Objekten und ermöglichen zudem die Entwicklung intuitiver Bedienoberflächen, wie in Abschnitt 2.4 erläutert wird.

2.1 Co-located Collaborative Workspaces (CCWs)

Unter einem Collaborative Workspace versteht man eine computergestützte Arbeitsumgebung, die für projektorientierte Zusammenarbeit mehrerer Personen konzipiert ist. Ein Co-located Collaborative Workspace (CCW) ist eine Sonderform eines Collaborative Workspaces, bei der sich alle miteinander arbeitenden Personen physisch im selben Raum befinden. Beispiele für diese Art von Arbeitsräumen sind das vom Fraunhofer Institut entwickelte i-LAND (Streitz u. a., 1999) und der iRoom (Fox u. a., 2000) der Universität Stanford, die in Abb. 2.1 dargestellt sind.

Konventionelle Versammlungsräume, wie man sie in Unternehmen und wissenschaftlichen Einrichtungen findet, werden primär für bestimmte Aufgabentypen benutzt. Dazu zählen Brainstormingsitzungen, Designaktivitäten (z. B. Erstellung einer Zeitschriftentitelseite) und planerische Aufgaben wie die Gestaltung der Innenarchitektur eines Hauses oder die Absprache eines Projektzeitplans.

Ein CCW kann als Versammlungsraum betrachtet werden, der speziell für die Arbeit mit digitalen Dokumenten optimiert ist. Dazu sind CCWs in der Regel mit großformatigen Touchscreens ausgestattet. Dabei handelt es sich um berührungsemfpindliche Monitore (siehe Abb. 2.1), die gleichzeitig als Eingabegerät zur Bedienung von Computer-Programmen dienen. Touchscreens erlauben es Anwendern durch Berührung der Anzeigefläche mit Fingern oder speziellen Stiften den Mauszeiger zu steuern.

Abb. 2.1: Links die iLand Arbeitsumgebung des Fraunhofer Instituts (von Fraunhofer-Gesellschaft, 2002) und daneben der iRoom der Universität Stanford (von Winograd, 2003).

Ein Ziel bei der Gestaltung von CCWs wie dem iLand ist die Minimierung von Computerpräsenz im Arbeitsraum: im Gegensatz zu konventionellen Büros werden Anzeigeflächen dazu in Tische und Wände integriert und die zugehörige Computerhardware „versteckt“ bzw. in eigene Räume verlagert. Damit folgen CCWs dem *Ubiquitous Computing*[1]-Gedanken (Weiser, 1991), der von Mark Weiser als *„the age of calm technology, when technology recedes into the background of our lives“* (Weiser, 1996) beschrieben wird. Computer sollen die Menschen bei der Arbeit unterstützen ohne dabei in den Vordergrund zu treten.

Der *Ubiquitous Computing*-Ansatz soll die Konzentration kollaborierender Personen auf ihre „eigentliche“ Arbeit fördern. Neben dem bereits erwähnten Verbergen von Hardware wird dazu spezielle Software eingesetzt, die bestimmte Aufgaben im Hintergrund und ohne menschliches Zutun erledigt. Dazu zählt z. B. der automatisierte Abgleich projektrelevanter Daten zwischen Notebooks, PDAs, Smartphones und anderen, in den CCW mitgebrachten, Geräten. Durch die Übertragung solcher Tätigkeiten auf Computerprogramme lässt sich die kognitive Belastung auf Personen im CCW verringern.

Ein weitere Möglichkeit zur kognitiven Entlastung für Personen, die im CCW arbeiten, ist der Einsatz nahtloser Interaktionstechniken, die im folgenden Abschnitt 2.2 näher betrachtet werden.

2.2 Interaktion in Collaborative Workspaces

Ein bekanntes Szenario aus dem Wirtschaftsalltag ist die Zusammenkunft von Angestellten zweier Firmen zu einer gemeinsamen Besprechung. Dabei kann das Treffen in einem Versammlungsraum einer der Firmen stattfinden, den die Angestellten der anderen Firma noch

[1] ubiquitous = allgegenwärtig

nie betreten haben. Der Aufenthalt im unbekannten Raum beeinträchtigt die Konzentration der „ortsfremden“ Personen, da sich diese zunächst orientieren müssen.

In einem CCW verschärft sich das beschriebene Szenario um die Komponente unbekannter Computerprogramme mit denen Anwender konfrontiert werden. Um die kognitive Belastung so gering wie möglich zu halten, ist es wünschenswert, dass die Benutzeroberflächen aller Programme im CCW so einfach und intuitiv bedienbar sind, wie dies Computerpionier Theodor Holm Nelson in einem seiner Mottos fordert:

> *„A user interface should be so simple that a beginner in an emergency can understand it within ten seconds.“*[2]

Eine Möglichkeit zur erfolgreichen Entwicklung intuitiv bedienbarer Benutzeroberflächen ist der Einsatz nahtloser Interaktionstechniken, die im folgenden Absatz 2.2.1 betrachtet werden.

2.2.1 Nahtlose Interaktion (Seamless Interaction)

Bei der Zusammenarbeit in Versammlungsräumen nutzen Menschen unter anderem Papier, Tafeln oder Whiteboards um Ideen und Informationen auszutauschen und zu dokumentieren. Wie in Abschnitt 2.1 erläutert, kann ein CCW als Versammlungsraum betrachtet werden, der sich besonders zur kollaborativen Bearbeitung digitaler Dokumente eignet. Die Anzeigeflächen im CCW übernehmen dabei die Funktion traditioneller Medien: anstatt mit Stift oder Kreide erfolgt die Erstellung von Textblöcken und Zeichnungen mithilfe eines Touchscreens.

Damit die Anfertigung einer Skizze auf einem Touchscreen genauso einfach wie auf einem Stück Papier erfolgen kann, sollten die dazu notwendigen Arbeitstechniken für den Anwender „nahtlos“ vom Papier auf den Touchscreen übertragbar sein. Dies bedeutet, dass der Anwender seine Finger wie einen Stift benutzen kann um Linien auf dem Touchscreen zu zeichnen.

Das Konzept *nahtloser Interaktion* (Seamless Interaction) eignet sich für den Entwurf von CCW-Interaktionstechniken da hierbei folgende Ziele verfolgt werden, wie Ishii u. a. (1994) schreiben:

1. Seamlessness (continuity) with existing work practices
2. Seamlessness (smooth transition) between functional spaces

[2] http://xanadu.com.au/ted/

Das erste Ziel *Continuity* bezieht sich auf Fähigkeiten und Techniken, die von vielen Menschen beherrscht werden. Dazu zählt Schreiben und Zeichnen mit der Hand wie auch das Verschieben und Drehen von Objekten auf Arbeitsflächen. Nach dem Konzept nahtloser Interaktion sollten diese Tätigkeiten in einem CCW durchgeführt werden können, wie es der Anwender z. B. von der Arbeit mit Papierdokumenten auf Tischen gewohnt ist.

Das zweite Ziel *Smooth Transition* betrifft den Wechsel zwischen unterschiedlichen Arbeitsräumen und -modi. Zum Beispiel überträgt eine Person während einer Brainstormingsitzung ihre persönlichen Ideen von einem Stück Papier zur Gruppendiskussion auf eine Tafel, die für alle im Raum sichtbar ist. Dabei erfolgt ein Wechsel vom privaten Arbeitsraum Papier zum öffentlichen Arbeitsraum Tafel. Gleichzeitig ändert sich der Arbeitsmodus von Einzelarbeit zu Gruppenarbeit. Entsprechende Übergänge sollten in einem CCW einfach und ohne großen Aufwand möglich sein.

An dieser Stelle ist es wichtig hervorzuheben, dass die Erstellung möglichst „naturgetreuer“ Kopien von Interaktionstechniken aus der Realwelt meist nicht sinnvoll ist. Vielmehr sollten dabei Vorteile, die sich aus dem Einsatz von Computern zur Ausführung der Techniken ergeben, einbezogen werden. So ist z. B. das schnelle Bündeln oder Umsortieren von Objekten, wie es viele Computerprogramme ermöglichen, dem manuellen Bewegung reeller Objekte hinsichtlich Fehlerfreiheit und Geschwindigkeit sicherlich überlegen. Bei der Übertragung von Interaktionstechniken muss also ein Abstraktion des Interaktionsprinzips erfolgen, wobei eine Loslösung vom reellen Vorbild und eine verbesserte Übersetzung in die Computerwelt das Ziel sein sollte. Durch diesen Ansatz ist es möglich Interaktionstechniken zu entwickeln, die Vorteile reeller intuitiver Interaktion mit der Geschwindigkeit computergestützer Arbeit vereinen.

Die spezifischen Eigenschaften von Eingabegeräten haben einen entscheidenden Einfluss auf die Entwicklung von Interaktionstechniken für Computersysteme. Um die Anforderungen nahtloser Interaktion zu realisieren, eignen sich besonders Touchscreens, die im folgenden Abschnitt 2.2.2 erläutert werden.

2.2.2 Touch-basierte Interaktion

Der Einsatz von Touchscreens in CCWs bietet eine Reihe von Vorteilen. Zum einen kann die Bedienung eines Touchscreens von Anwendern schnell erlernt werden, wie Benko u. a. (2006) schreiben:

> *„The ability to directly touch and manipulate data on the screen without using any intermediary devices has a very strong appeal to users. In particular, novices benefit most from the directness of touch screen displays. A fast learning*

> *curve and inherent robustness (no movable parts) makes touch screens an ideal interface for interacting with public installations [...]“*

Touchscreens ermöglichen eine relativ „natürliche“ Interaktion: Anwender nach Objekte auf der Bedienoberfläche greifen und sie „anfassen“. Damit tragen Touchscreens dazu bei die kognitive Belastung auf CCW-unerfahrene Anwender zu reduzieren, wie in 2.2 erläutert wurde. Einige weitere Vorteile von Touchscreens gegenüber indirekten Eingabegeräten wie Maus oder Trackball werden von Sears und Shneiderman (1991) beschrieben:

> *„Touchscreens are easy to learn to use, require no additional work space, have no moving parts, and are very durable.“*

Ferner können mit Hilfe von Touchscreens reelle Interaktionstechniken digital nachgebildet werden. Die Erstellung einer Zeichnung mit einem Stift auf einem Stück Papier kann auf einem Touchscreen analog z. B. mit einem Finger durchgeführt werden. Ebenso können digitale Objekte mithilfe eines Touchscreens und geeigneter Programme durch „Berührung“ verschoben werden. In der Realität funktionieren entsprechende Bewegungsvorgänge ebenfalls durch Berührung. Damit unterstützen Touchscreens die *Continuity* von Anwendungen, welche in Abschnitt 2.2.1 als Teilziel nahtloser Interaktion beschrieben wird.

Trotz der genannten Vorteile weisen gegenwärtig kommerziell erhältliche Touchscreens eine Reihe von Mängeln auf. Dazu zählen geringe Präzision, hohe Fehlerraten und durch längere Benutzung bedingte Ermüdung der Arme (Albinsson und Zhai, 2003). Weiterhin kann aufgrund technischer Beschränkungen wie fehlender Unterstützung durch aktuelle Betriebssysteme und Software mit den meisten kommerziell verfügbaren Touchscreens nur ein Berührungspunkt verfolgt werden. Inzwischen existieren jedoch eine Reihe von Prototypen (Han, 2005; Dietz und Leigh, 2001; Rekimoto, 2002), die Multi-Touch unterstützen und daher die Verfolgung mehrerer Berührungspunkte erlauben. Zu den wenigen kommerziellen Produkten die über Touchscreens mit Multi-Touch verfügen, zählt das 2007 erschienene iPhone von Apple (Abb. 2.3) und der Microsoft Surface Tisch (Abb. 2.2).

Die Arbeit auf horizontalen Arbeitsflächen wie dem Microsoft Surface Tisch bringt spezifische Anforderungen an die Gestaltung von Bedienoberflächen und Interaktionstechniken mit sich. Diese werden im folgenden Abschnitt 2.2.3 dargestellt.

2.2.3 Tabletops

Wie in Abbildung 2.1 zu sehen, lassen sich digitale Arbeitsflächen in CCWs anhand ihrer Orientierung unterscheiden. Vertikale Monitore mit berührungssensitiven Oberflächen nennt man Interactive Walls. Ebenfalls vertikale Anzeigeflächen, die eine sehr hohe Auflösung

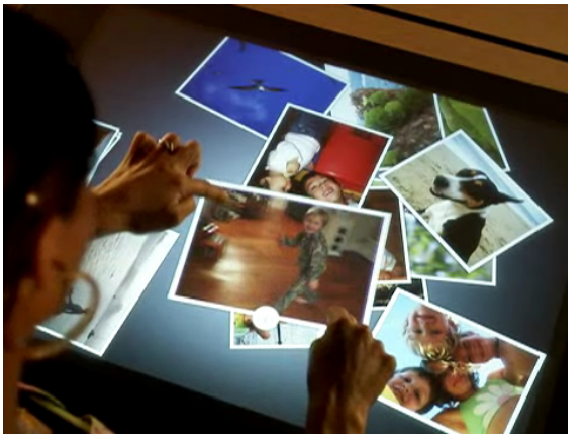


Abb. 2.2: Microsoft Surface Tisch (von Microsoft Corporation, 2007a).

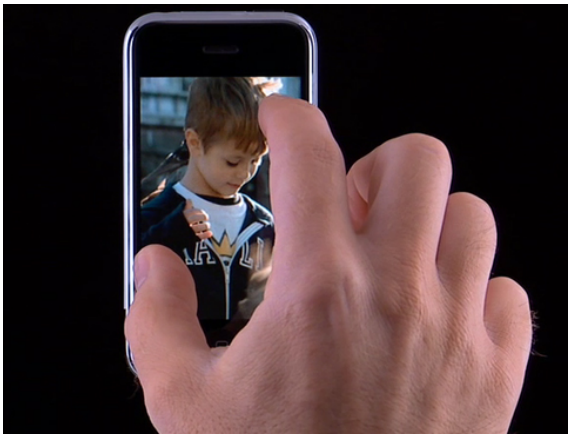


Abb. 2.3: Apple iPhone (von Apple Inc., 2007).

durch die Verbindung mehrerer Monitore bieten, heissen Powerwalls. Horizontale Interaktionsflächen, die in Tische eingelassen sind bzw. darauf projiziert werden, bezeichnet man als Tabletops.

Bei Besprechungen in kleinen Personengruppen werden Tische häufig als zentrale Arbeitsfläche benutzt. Die Gruppierung der Gesprächsteilnehmer um einen Tisch hat mehrere Vorteile:

- Alle Gesprächsteilnehmer können leicht Augenkontakt zueinander aufnehmen, da sie einander zugewandt sind.
- Dokumente und Objekte, die sich auf dem Tisch befinden sind für alle sichtbar.
- Der Tisch dient gleichzeitig als Ablage- und Präsentationsfläche.
- Tätigkeiten von anderen Gruppenmitgliedern am Tisch können gut beobachtet werden.

Besonders der letzte Punkt spielt für die simultane Zusammenarbeit mehrerer Personen eine wichtige Rolle, wie Scott (2005) schreibt:

> *„The ability to monitor the artefacts and the interactions of others [...] helps group members anticipate when assistance may be needed and helps them to understand their collaborators' motivations for actions that they may perform later [...]"*

Die Erforschung von Tabletop-Arbeitsflächen und -Software ist ein relativ junges Forschungsfeld (einer der ersten internationalen Workshops zu diesem Thema ist die TABLETOP [3]). Ge-

[3] http://www.ieeetabletop2007.org/

genwärtig besteht ein Schwerpunkt der Forschung auf diesem Gebiet grundlegende Interaktionsschemata zu entdecken und zu verstehen, wie sie Menschen bei kooperativer Arbeit an Tischen einsetzen. Darauf aufbauend kann die Entwicklung geeigneter Tabletop-Displays und -Software erfolgen, wie von Scott (2005) gefordert wird:

> *„[...] the interface components and interaction techniques that will provide the basic building blocks for tabletop groupware designers must first be developed – similar to interface components such as buttons, sliders, and drop-down menus used in standard desktop applications – before effective tabletop groupware systems can be developed.“*

Inzwischen wurden mehrere Forschungsarbeiten veröffentlicht, die sich mit Arbeitstechniken von Menschen an gewöhnlichen Tischen beschäftigen. Der folgende Abschnitt 2.2.3 stellt einige der bisher gewonnenen Erkenntnisse dar.

Gruppeninteraktion an Tischen

Wie Liu u. a. (2006) beschreiben, können bei kollaborativer Arbeit mehrerer Personen an Tischen eine Reihe von häufig auftretenden Verhaltensmustern beobachtet werden. Dazu zählen die in den Abbildungen 2.4 – 2.7 (alle Abbildungen aus Liu u. a. (2006)) gezeigten Interaktionstechniken.

Abb. 2.4: Anfordern eines entfernten Objekts durch Zeigen.

Abb. 2.5: Greifen nach entfernten Objekten.

Bei näherer Betrachtung der gezeigten Situationen fällt auf, dass die Arbeitsfläche von der Arbeitsgruppe in Teilbereiche zerlegt wurde. Scott (2005) bezeichnet diese als *personal* (persönliche Arbeitsbereiche), *group* (Gruppenarbeitsbereiche) und *storage territories* (Ablageflächen). Persönliche Arbeitsbereiche umschließen in der Regel ein halbkreisförmiges Territorium direkt vor jeder Person an der Arbeitsfläche, während sich Gruppenarbeitsbereiche

Abb. 2.6: Verlagerung des Arbeitsbereichs in die Tischmitte.

Abb. 2.7: Initiierung einer Gruppendiskussion durch Zeigen eines Objekts.

meist in der Mitte des Tisches befinden und da sie dort für alle gleichermaßen gut erreichbar sind (siehe Abb. 2.6). Ablagebereiche werden meist entweder nur von einer Person oder von der ganzen Gruppe benutzt. Zum besseren Verständnis sind die drei erläuterten Territorialtypen in Abb. 2.8 zusammengefasst.

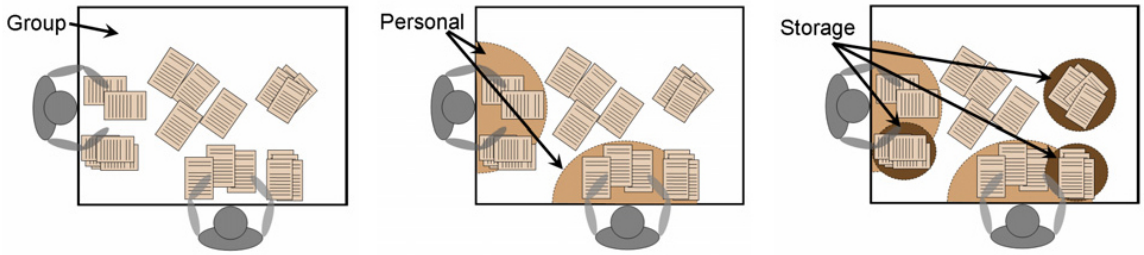


Abb. 2.8: Territorialtypen bei kollaborativer Arbeit an Tischen (von Scott, 2005).

Die Ausrichtung der Objekte auf horizontalen Arbeitsflächen wird von unterschiedlichen Faktoren beeinflusst und spielt eine wichtige Rolle bei der Kollaboration, wie im nächsten Abschnitt 2.3.1 dargelegt wird.

2.3 Objektausrichtung auf horizontalen Arbeitsflächen

2.3.1 Objektausrichtung als Kollaborationshilfsmittel

In Abbildung 2.7 ist deutlich erkennbar, dass Zeitschriften und Bilder auf dem Tisch unterschiedlich ausgerichtet sind. Zum Beispiel sind Objekte in persönlichen Bereichen so rotiert,

dass sie aufrecht vor dem Mitglied der Arbeitsgruppe liegen, das gerade mit ihnen arbeitet. Die Anpassung der Objektausrichtung an die Sitzposition der Personen ist jedoch nur ein Teilaspekt, der die Rotation von Objekten auf dem Tisch beeinflusst. Wie Kruger u. a. (2003) in ihrer Beobachtungsstudie erläutern, umfasst die Rolle der Objektausrichtung folgende Funktionen:

Auffassung Objekte die nicht auf dem Kopf stehen, können besser erkannt, betrachtet und gelesen werden.

Koordination Die Ausrichtung signalisiert welche Person gerade mit einem Objekt arbeitet. Weiterhin dient sie zur Abtrennung der unterschiedlichen Arbeitsbereiche (*personal*, *group* und *storage*).

Kommunikation Die Ausrichtung eines Objekts hin zu anderen Mitarbeitern unterstützt die Initiierung einer Gruppendiskussion.

Die Ausrichtung der Puzzleteile in den Abbildungen 2.4 – 2.6 zeigt relativ klar, welche Person sich gerade für welche Teile auf dem Tisch zuständig fühlt. Dies ist ein Beispiel für die Koordinationsfunktion.

Die Kommunikationsfunktion ist in Abbildung 2.7 sichtbar: die Person am linken Tischrand hält eine Zeitschrift so gedreht, dass sie von den übrigen Mitarbeitern am Tisch aufrecht betrachtet werden kann und macht sie damit zum Diskussionsgegenstand.

Die Ausrichtung der Objekte spielt bei der Ausführung gemeinschaftlicher Arbeiten an Tischen also eine entscheidende Rolle. Entsprechend sollte es möglich sein, digitale Objekte auf Tabletops in CCWs einfach und mit geringem kognitiven Aufwand zu drehen. Bei der Entwicklung von Rotationstechniken auf Tabletop-Systemen lassen sich verschiedene Ansätze beobachten, die diesen Anforderungen unterschiedlich gut gerecht werden. Diese werden im folgenden Abschnitt 2.3.2 untersucht.

2.3.2 Objektausrichtungstechniken auf Tabletops

Abbildung 2.9 zeigt einige, in den letzten Jahren entwickelte Herangehensweisen an den Themenkomplex Objektrotation und -translation auf Tabletops, wie sie teilweise auch von Kruger u. a. (2003) und Kruger u. a. (2005) beschrieben werden.

Wie in der Grafik zu sehen, lassen sich vier Ansätze unterscheiden: *3 Degrees of Freedom (DOF)*, *Umgebungsbasiert*, *Situationsbasiert* und *2 DOF mit Physiksimulation*. Diese Ansätze werden in den folgenden Abschnitten 2.3.2 – 2.3.2 vorgestellt.

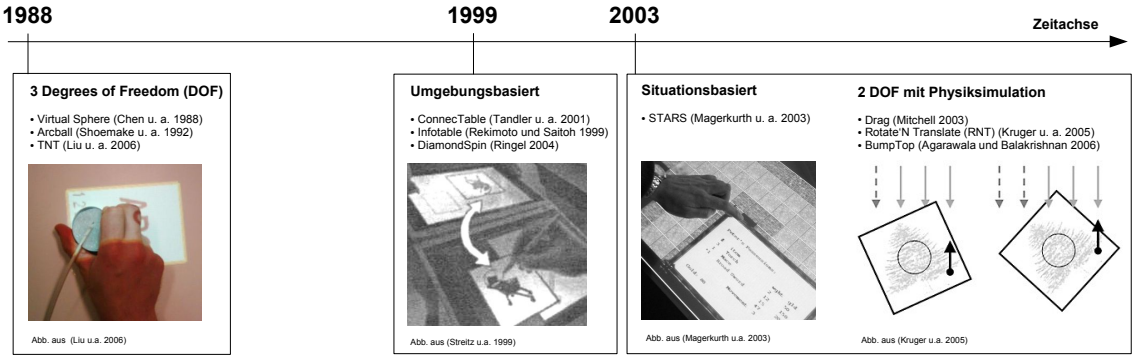


Abb. 2.9: Zeitliche Entwicklung der Forschungslandschaft Objektrotation/-translation auf Tabletops.

3 Degrees of Freedom (DOF)

Konventionelle Eingabegeräte wie Mäuse oder Trackballs erlauben eine Bewegung des Cursors in X- und Y-Richtung und verfügen damit über zwei Freiheitsgrade (degrees of freedom). Die gleichzeitige Rotation eines Objekts während einer Translation ist hier nicht möglich, da durch die Bewegung bereits alle Freiheitsgrade ausgeschöpft sind. Dieses Problem umgehen speziell entwickelte Eingaberäte wie Arcball (Shoemake, 1992) oder Virtual Sphere (Chen u. a., 1988), die drei Freiheitsgrade bieten und es damit erlauben Translationen und Rotationen in einer Bewegung auszuführen. Eine der neueren Arbeiten auf diesem Forschungsgebiet stammt von Liu u. a. (2006) und trägt den Namen TNT. Wie in Abb. 2.9 ganz links zu sehen, kommt bei TNT ein Zylinder als Eingabegerät zum Einsatz. Dieser kann vom Anwender während der Bewegung gedreht werden.

Umgebungsbasiert

Bei umgebungsbasierten Techniken erfolgt die Drehung von Objekten in Abhängigkeit von bestimmten Rahmenbedingung automatisch. Auf dem InfoTable von Rekimoto und Saitoh (1999) werden Objekte aufrecht zur nächstgelegen Kante des Tabeltops gedreht, wenn sie von einer Person zu sich gezogen werden. Bei DiamondSpin (Ringel u. a., 2004) werden Gegenstände auf dem Tabletop auch ohne Ziehbewegung immer zur nächstgelegen Kante rotiert. Dagegen erfolgt die Objektdrehung beim ConnecTable von Tandler u. a. (2001) nach dem Verschieben eines Objekts: die erste Person die das Objekt im Anschluss berührt, wird als Empfänger betrachtet und daher wird das Objekt so rotiert, dass es für diese Person aufrecht steht.

Situationsbasiert

Bei rundenbasierten kollaborativen Tätigkeiten, wie z. B. Spielen, können Objekte automatisch zum aktiven Anwender gedreht werden, wie es beim STARS-System von Magerkurth u. a. (2003) der Fall ist. Diese Technik lässt sich, wie in Abb. 2.9 zu sehen, gut für Brettspiele einsetzen.

2 DOF mit Physiksimulation

Mehrere aktuelle Forschungsarbeiten versuchen den Mangel von Eingabegeräten mit 2 Freiheitsgraden (siehe Abschnitt 2.3.2) durch Simulation physikalischer Kräfte zu beseitigen. Sowohl *Drag* (Mitchell, 2003) als auch *Rotate'N Translate (RNT)* von Kruger u. a. (2005) erzielen durch Berechnung von Reibungskräften (siehe Abb. 2.9 ganz rechts), die bei der Bewegung von Objekten auf Tischflächen auftreten, eine simultane Translation und Rotation von Objekten. Beide Verfahren verwenden dafür speziell entwickelte Algorithmen.

Im Gegensatz dazu kommt beim virtuellen Desktop *BumpTop* (Agarawala und Balakrishnan, 2006) eine Physik-Engine zur Berechnung der physikalischen Kräfte zum Einsatz. Dabei handelt es sich um eine Software-Bibliothek, die es zum einen erlaubt, 3-dimensionale Körper anhand physikalischer Parameter wie Dimension, Dichte, Gewicht usw. zu modellieren. Zum anderen kann mit der Physik-Engine das Verhalten der modellierten Körper unter Krafteinwirkung berechnet werden. Hauptanwendungsgebiete für Physik-Engines sind moderne Computerspiele und Software zur Simulation physikalischer Prozesse. Ein Beispiel für ein Programm, das eine Physik-Engine nutzt, zeigt Abb. 2.10, in der eine Pyramide aus mehreren Würfel durch den Aufprall einer heranfliegenden Kugel zusammenstürzt.

Der Einsatz einer Physik-Engine in BumpTop erlaubt es nicht nur Reibungskräfte, sondern auch die Reaktion von digitalen Objekten untereinander und mit der Umgebung zu simulieren. Beispielsweise prallen Objekte voneinander ab und lassen sich über die Tischfläche werfen oder durch Anstoßen verschieben.

Diskussion

Die drei vorgestellten Ansätze (2.3.2 – 2.3.2) zur Rotation von Objekten auf Tabletops weisen eine Reihe von Nachteilen auf.

Speziell entwickelte, in Abschnitt 2.3.2 beschriebene, 3 DOF-Eingabegeräte sind bisher nicht weit verbreitet und machen die Rotierbarkeit von Objekten von spezieller Hardware abhängig.

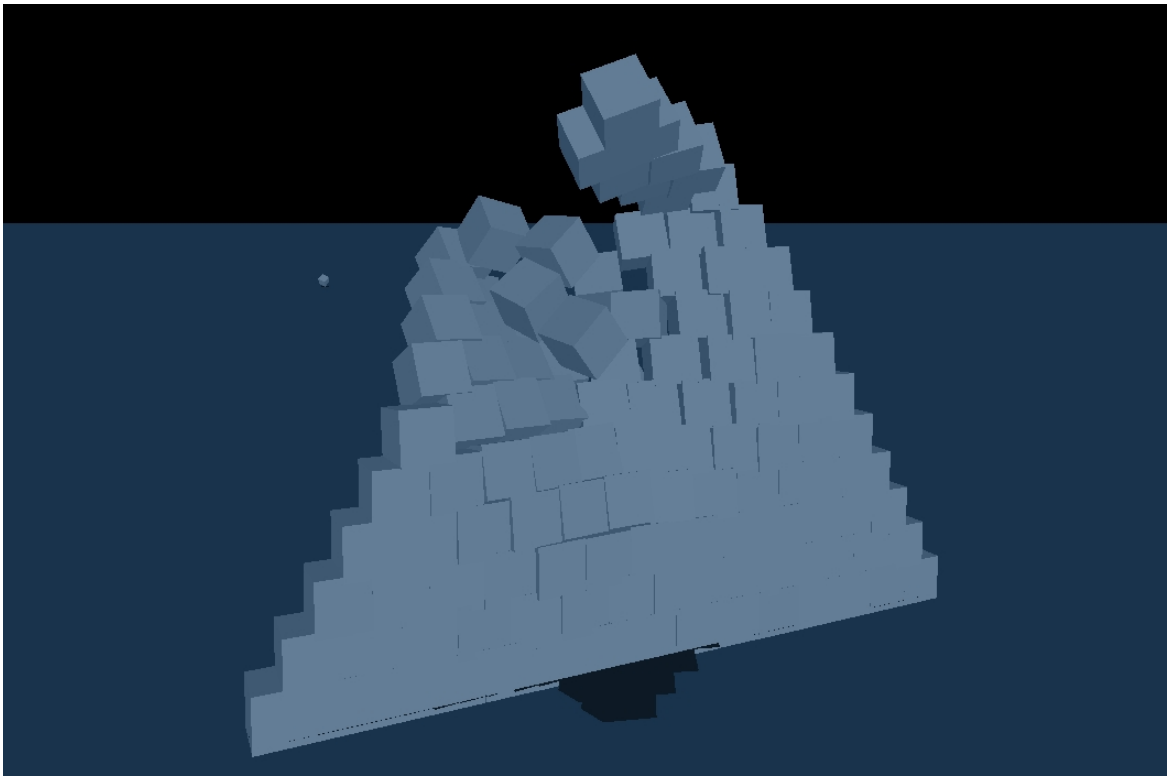


Abb. 2.10: Simulation in der das Verhalten mehrerer Körper mit Hilfe einer Physik-Engine berechnet wird.

Bei den gezeigten umgebungsbasierten Verfahren (siehe Abschnitt 2.3.2) wird davon ausgegangen, dass es optimal ist, Objekte stets zur Person die mit ihnen arbeitet bzw. den Objekten am nächsten sitzt, aufrecht zu rotieren. Wie in 2.2.3 beschrieben ist dies aber nicht immer der Fall. Will ein Anwender z. B. über ein Objekt auf der Arbeitsfläche diskutieren, ist es wichtig, dass seine Mitarbeiter das Objekt aufrecht betrachten können. Mit den vorgestellten umgebungsbasierten Methoden ist eine entsprechende Rotation nicht möglich.

Das in 2.3.2 beschriebene Verfahren eignet sich gut für rundenbasierte Interaktion, wie sie häufig bei Spielen verwendet wird. Da jedoch viele andere Anwendungen ein simultanes Agieren mehrerer Personen erfordern, ist dieser Ansatz in vielen Fällen nicht anwendbar.

Die abschließend in 2.3.2 vorgestellten physikbasierten Verfahren weisen die Nachteile der anderen Ansätze nicht auf: sie sind unabhängig von spezieller Hardware, verzichten auf automatische Rotation und geben dem Anwender somit volle Kontrolle. Aus diesen Gründen bieten sich physikbasierte Verfahren zur Rotation von Objekten auf Tabletop-Systemen besonders an. Aus der zeitlichen Einordnung der Verfahren in Abb. 2.9 lässt sich erkennen, dass sich der Forschungsfokus in letzter Zeit verstärkt auf physikbasierte Verfahren richtet.

2.3.3 Physikbasierte Interaktionstechniken

Wie in 2.3.2 beschrieben, unterscheiden sich die entwickelten physikbasierten Verfahren in ihrem Funktionsumfang, der u.a. durch die Art der Physikberechnung bestimmt wird. Das folgende Kapitel betrachtet zwei der vorgestellten Arbeiten näher. Dabei handelt es sich um *RNT* und *BumpTop*.

Rotate'N Translate (RNT)

Mit RNT kann ein Objekt mithilfe eines Kontaktpunktes in einer Bewegung gleichzeitig rotiert und verschoben werden.

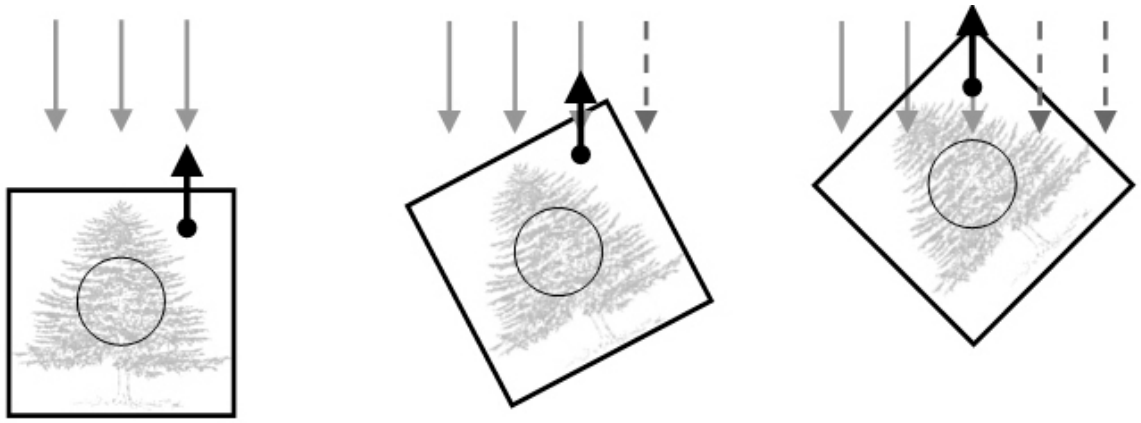


Abb. 2.11: Simultane Rotation und Translation über einen Berührungspunkt bei RNT (von Kruger u. a., 2005).

Abbildung 2.11 veranschaulicht das Verfahren. Kruger u. a. (2005) beschreiben die Funktionsweise von RNT wie folgt:

> *„Imagine a current that acts against the object always in direct opposition to the object's movement vector. If the direction of movement changes, so too does the current, maintaining its direct opposition. When the object is stationary, no current exists. As the object is manipulated, the current acts against the object to produce rotational changes, while the movement vector yields positional changes."*

Zur Berechnung des Drehverhaltens nutzt der RNT-Algorithmus den Mittelpunkt[4] des Objekts (C), die Position des Cursors zu Beginn (O) und am Anfang der Bewegung (T). Aus einer

[4] Der Schwerpunkt ist bei Objekten mit gleichmässiger Masseverteilung gleich dem Objektmittelpunkt.

Translation ergibt sich damit der Translationsvektor *OT* und der Rotationswinkel θ, wie in Abb. 2.12 zu sehen.

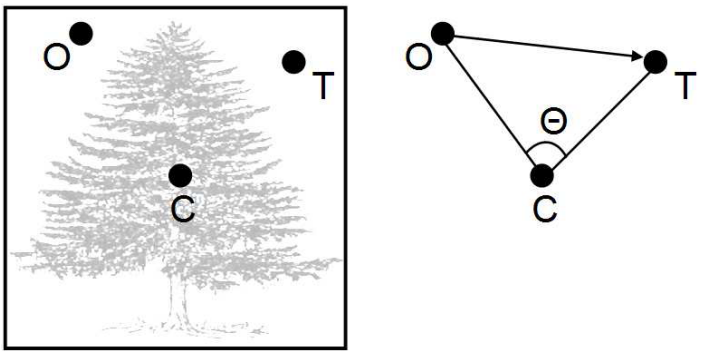


Abb. 2.12: Funktionsweise des RNT-Algorithmus (von Kruger u. a., 2005).

RNT wurde anhand mehrerer Aufgaben mit der von Desktoprogrammen bekannten Rotationstechnik Corner-to-Rotate (CTR) verglichen. Bei CTR kann eine Rotation nach einem expliziten Befehl durch Interaktion mit quadratischen Interaktionsbereichen, die an den Ecken des Objekts angezeigt werden, ausgelöst werden. CTR wird z. B. bei vielen Grafikprogrammen eingesetzt.

Beim Vergleich zwischen RNT und CTR hatten Probanden unter anderem die Aufgabe gemeinschaftlich eine Rätselaufgabe an einem Tabletop zu lösen. Bei deren Bearbeitung mussten sich die Versuchspersonen Objekte auf einem Tabletop zuschieben und rotieren. Zusammenfassend ergab sich aus den Untersuchungen, dass bei RNT im Vergleich zu CTR weniger Berührungen eines Objektes nötig sind und dass Objekte über eine kürzere Strecke (gemessen in Pixel) hinweg gehalten werden müssen.

Bei RNT wird die Berechnung physikalischer Kräfte lediglich zur simultanen Rotation und Translation von Objekten genutzt. Das im folgenden Abschnitt vorgestellte BumpTop stellt dem Anwender darüber hinaus einen physikbasierten Desktop zur Verfügung, auf dem Objekte, wie in der Realität, auf die Einwirkung physikalischer Kräfte reagieren.

BumpTop

BumpTop (Agarawala und Balakrishnan, 2006) greift die, von aktuellen Betriebssystemen wie Windows oder OS X her bekannte, Desktop-Metapher auf. Datenobjekte werden dabei auf einer abgegrenzten Oberfläche durch grafische Objekte repräsentiert, wie in Abbildung 2.13 zu sehen.

Abb. 2.13: Der virtuelle Schreibtisch BumpTop (von Agarawala und Balakrishnan, 2006).

BumpTop unterstützt wie RNT simultane Rotationen und Translationen. Dies ist allerdings eher ein Nebenprodukt, welches sich durch den Einsatz einer Physik-Engine[5] zur Simulation der physikalischen Kräfte auf dem virtuellen Desktop ergibt.

Bei BumpTop besitzt jedes Objekt auf der Bedienoberfläche physikalische Charakteristiken, wie Masse, Größe und Oberflächenbeschaffenheit, die im Kontakt mit dem Untergrund oder anderen Objekten in einem Reibungskoeffizienten resultiert. Ferner können Objekte miteinander kollidieren und dadurch bei entsprechend hoher Geschwindigkeit und Masse andere Objekte verschieben.

Wie die Autoren beschreiben, waren Probanden in der Lage die Bedienung von BumpTop größtenteils selbst zu erlernen. Viele Interaktionstechniken wurden von den Versuchspersonen „spielerisch“ entdeckt, ohne explizit darauf aufmerksam gemacht worden zu sein. Dazu zählt z. B. die Möglichkeit Objekte zu stapeln, aufeinander zu häufen oder durcheinander zu werfen. Dies ist vermutlich darin begründet, dass es mit Hilfe physikbasierter Interaktion möglich ist, Interaktionstechniken aus der Realwelt auf den virtuellen BumpTop-Schreibtisch

[5] Ageia Physics SDK (http://www.ageia.com/).

zu übertragen. Dies verringert die Einarbeitungszeit in die Programmbedienung und stellt einen Vorteil physikbasierter Bedienoberflächen dar.

Das Bedienkonzept von BumpTop findet sich ebenfalls in dem kommerziell verfügbaren Programm Real Desktop, dessen Bedienoberfläche Abb. 2.14 zeigt.

Abb. 2.14: Bedienoberfläche von Real Desktop.

Zukunftsperspektive physikbasierter Programme

Die Vorzüge physikbasierter Interaktionstechniken, welche in Abschnitt 2.3.3 und 2.3.3 beschrieben werden, sprechen für eine Fortsetzung der Forschungsbemühungen auf diesem Gebiet. Weil die dabei verwendeten Interaktionstechniken aus der Realwelt stammen und somit nahezu allen potentiellen Anwendern bekannt sein sollten, stellt sich die Frage, warum physikbasierte Interaktionstechniken bei der Bedienung von Computersystemen nicht weiter verbreitet sind.

Bei vielen aktuellen Spieletiteln, wie z. B. Half-Life 2 oder Cellfactor, nimmt die realitätsnahe Darstellung einer physikbasierten Spielumgebung einen hohen Stellenwert ein, weil hierdurch der Immersionsgrad und die Glaubwürdigkeit des Spiels weiter erhöht werden kann. Die Simulation sehr komplexer physikasierter Umgebungen mit vielen Objekten bringt jedoch einen hohen Leistungsbedarf mit sich, der die Kapazität aktueller Prozessoren stark beansprucht. Neben der Nutzung von Multi-Core-Prozessorenvon, nennt Luban (2007) den Einsatz spezieller Hardware-Beschleunigerkarten[6] als Möglichkeit um den Prozessor bei der Berechnung physikalischer Effekte zu entlasten.

Wie die Verbreitung von 3D-Grafik und 3D-Grafikkarten in den letzten Jahren, könnte auch der Durchbruch physikbasierter Anwendungsumgebungen und spezieller „Physik-Hardware“ durch die rasch voranschreitende Entwicklung bei Computerspielen beschleunigt werden. So zählen 3D-Grafikkarten inzwischen zu den Standardkomponenten aktuell verfügbarer Computersysteme, da selbst auf Betriebssystemebene 3D-Effekte eingesetzt werden, wie z. B. bei Windows Vista.

Aufgrund der stetig wachsenden Leistungsfähigkeit von Computersystemen für Endanwender, der zunehmenden Verbreitung spezieller Hardware zur Physikbeschleunigung und der leicht zu erlernenden Bedienbarkeit, haben physikbasierte Interaktionstechniken gute Chancen integraler Bestandteil zukünftiger Bedienoberflächen zu werden. Ein weiteres entscheidendes Argument für den Einsatz physikbasierter Anwendungsumgebungen und Bedienoberflächen in Computerprogrammen wird im folgenden Abschnitt 2.4 beschrieben.

2.4 Software-Gestaltung auf Basis mentaler Modelle

Im Forschungsfeld der Human-Computer-Interaction (HCI) hat das Konzept mentaler Modelle in den letzten Jahren zunehmend an Bedeutung gewonnen. Die generelle Idee des Konzepts beschreiben Jacko und Sears (2003) wie folgt:

> *„[...] as the user interacts with the computer, he or she receives feedback from the system that allows him/her to develop a representation of how the system is functioning for a given task.“*

Die subjektiv empfundene Bedienqualität eines Programms ist dabei eng mit dem mentalen Modell des Anwenders verknüpft, wie Sasse (1997) erläutert:

> *„[...] a well-designed system and user interface will allow the user to develop an appropriate internalised model of that system, which in turn facilitates users' learning of, and interaction with, the system.“*

[6] siehe http://www.ageia.com/

Dieses Zitat beschreibt den Kerngedanken des konzeptionellen Designansatzes von Norman (1988), der davon ausgeht, dass Menschen basierend auf ihren Erwartungen mentale Modelle von Systemen entwickeln. Ein mentales Modell ist eine Sammlung von Annahmen, die hinsichtlich der Funktionalität eines Systems getroffen werden. Sind die Annahmen richtig, dann reagiert das System so, wie es der Anwender erwartet. Sein mentales Modell stimmt in diesem Fall mit der Funktionalität des Systems überein.

Ein Hauptproblem bei der Gestaltung von Computer-Anwendungen wird in Abb. 2.15 illustriert. Der Entwickler (Designer) macht sein mentales Modell in Form eines Programms (System image) greifbar, indem er dessen optische Erscheinung und Reaktion auf Benutzereingaben entsprechend gestaltet. In der Regel unterscheidet sich das in Form des Programms manifestierte mentale Modell des Entwicklers mehr oder weniger stark vom mentalen Modell des Anwenders (User). Je mehr es dem mentalen Modells des Anwenders entspricht, desto leichter und intuitiver kann das Programm von ihm bedient werden.

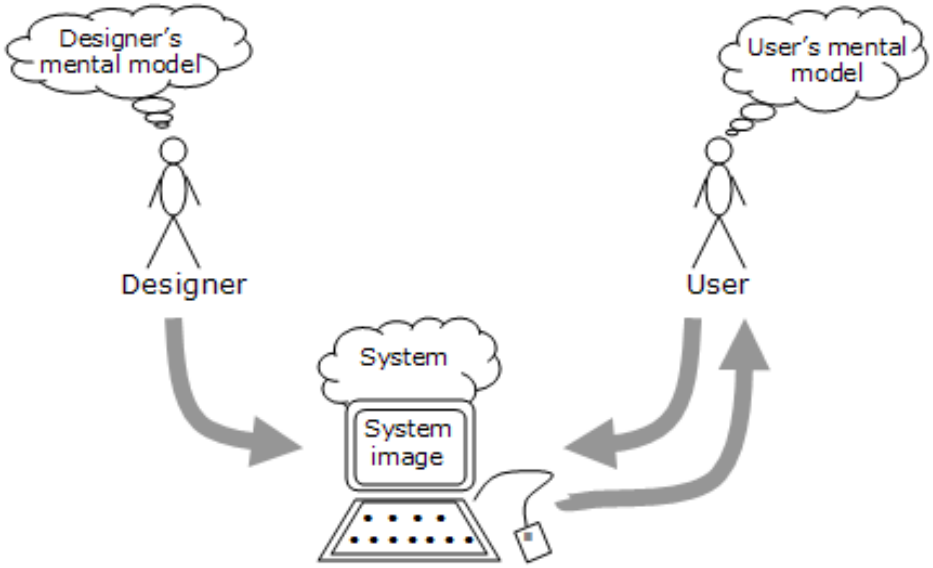


Abb. 2.15: Unterschiedliche mentale Modelle führen zu unterschiedlichen Annahmen hinsichtlich der Funktionalität von Anwendungen (von Soegaard, 2007 in Anlehnung an Norman, 1988).

Zur erfolgreichen Entwicklung mentaler Modelle, die den Vorstellungen von Anwendern entsprechen, empfiehlt Tognazzini (1992) die Verwendung von Metaphern oder Analogien. Sasse (1997) definiert beide Begriffe wie folgt:

> *„An analogy [...] provides an explicit, referentially isomorphic mapping between objects in two domains; valid analogies can therefore only be constructed between similar domains. A metaphor is a looser type of mapping, which points*

> *out similarities between two domains, without making explicit links between individual objects, or involving all objects. A metaphor is therefore referentially arbitrary and open-ended; its primary function is to initiate a process of active learning in the user.“*

Ein Computer-Programm wie DynAmbient, kann anhand dieser Beschreibung als Implementation einer Analogie bezeichnet werden, da Objekte aus der Realität mit ihren physikalischen Eigenschaften direkt in die Anwendung übertragen werden. Gleiches gilt für die Bereitstellung physikbasierter Interaktionstechniken, die sich eindeutig auf analoge Techniken in der Realität beziehen.

Menschen entwickeln im Laufe ihres Lebens mentale Modelle vom Verhalten physikalischer Objekte unter dem Einfluss externer Kräfte. Aufgrund weltweit ähnlicher Umweltbedingungen (Schwerkraft, Luftwiderstand, etc.) ist anzunehmen, dass bestimmte mentale Modelle von einem Großteil der Menschheit geteilt werden. Folglich besitzt vermutlich sowohl der Entwickler als auch der Anwender eines physikbasierten Programms ein sehr ähnliches mentales Modell vom Verhalten physikalischer Objekte in einer Arbeitsumgebung wie sie z. B. von BumpTop (vgl. Abschnitt 2.3.3) bereit gestellt wird. Dieser Sachverhalt wird in Abb. 2.16 nochmals verdeutlicht.

Mittlerweile existieren eine Vielzahl ausgereifter Software-Bibliotheken, die eine Nachstellung komplexer reeller Szenerien ermöglichen und es erlauben die Wirkung von Kräften auf Objekte mit physikalischen Eigenschaften innerhalb dieser Szenerien zu simulieren. Computer-Anwendungen, die solche Bibliotheken in geeigneter Weise zur Nachbildung der Realität einsetzen, werden mit hoher Wahrscheinlichkeit als intuitiv empfunden, da sie mentalen Modellen der Anwender entsprechen, die aus der Realität abgeleitet sind. Dies ist ein entscheidender Vorteil physikbasierter Anwendungen.

Nach Einführung in den Kontext dieser Masterarbeit und Darstellung einiger Vorteile physikbasierter Programme und Interaktionstechniken, werden im folgenden Kapitel 3 Anforderungen definiert, die eine physikbasierte Computer-Anwendung zum Einsatz in einer CCW-Umgebung erfüllen sollte.

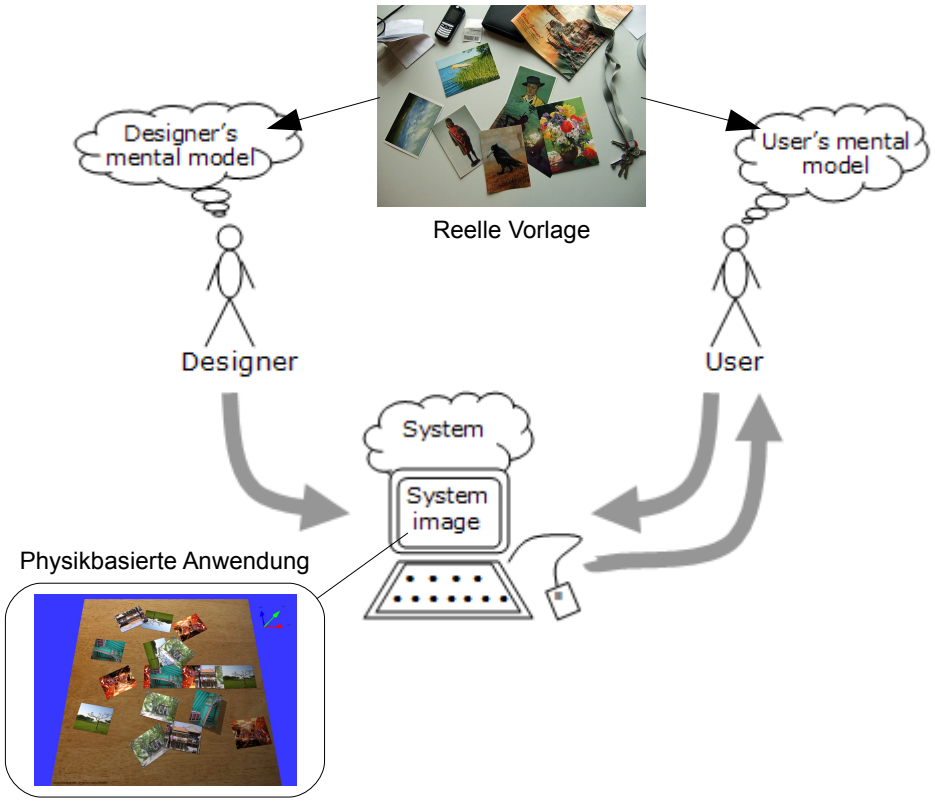


Abb. 2.16: Entwicklung einer Anwendung, auf Basis mentaler Modelle mit gleicher Vorlage unter Einsatz physikalischer Simulationsalgorithmen (abgeänderte Darstellung von Soegaard, 2007 in Anlehnung an Norman, 1988).

3 Analyse

Die Anforderungsanalyse in diesem Kapitel orientiert sich an einem Beispielszenario, das zunächst im folgenden Abschnitt 3.1 vorgestellt wird. Auf Basis des Szenarios wird eine Softwareapplikation namens DynAmbient entwickelt, die bestimmten funktionalen und nichtfunktionalen Anforderungen genügen soll, welche in Abschnitt 3.2 und 3.3 formuliert werden.

3.1 Beispielszenario

Ein aktueller Forschungsschwerpunkt des UbiComp-Projekts[1] der HAW Hamburg befasst sich mit der funktionalen und technologischen Entwicklung von Geräten und Arbeitsumgebungen für Rettungseinsätze. Konkret werden Teilaspekte eines Feuerwehreinsatzes bei einem Großbrand (Rescue-Szenario) betrachtet. Dazu zählt die Ausstattung der Feuerwehrleute mit tragbaren Computersystemen (Wearable Computing) (Hinck, 2007) und die Überwachung brennender Gebäude durch Ad-hoc Sensornetzwerke (Davids, 2007).

Ein weiterer wesentlicher Bestandteil des Szenarios ist die Entwicklung eines Leitstands (Piening, 2007) zur Einsatzkoordination. Der Leitstand dient dabei unter anderem als zentrale Sammelstelle für Audio-, Video- und Sensordaten, die kontinuierlich von Rettungskräften und unterstützenden Systemen (Sensoren, Infrarotkameras etc.) am Brandort gesendet werden. Eine wichtige Aufgabe der Arbeitskräfte im Leitstand ist die Sichtung und zielgerichtete Weiterleitung der eingehenden Daten.

Bei der Beurteilung der Lage durch die Einsatzleitung können Fotos und Videos, die z. B. aus dem Inneren eines brennenden Gebäudes stammen, ein entscheidendes Hilfsmittel darstellen. Entscheidungen, die auf Basis dieser Medien getroffen werden, erfordern unter Umständen das Urteil mehrerer Verantwortlicher und Experten (z. B. Statiker, Ärzte etc.). Daher ist es erforderlich, dass eingehende Datenobjekte im Leitstand gleichzeitig von mehreren Personen begutachtet und diskutiert werden können.

Der Leitstand kann als spezielle Variante eines CCW (vgl. Abschnitt 2.1) betrachtet werden, der für die kooperative Arbeit von Rettungskräften konzipiert ist. Im Kontext des Leitstands

[1] http://users.informatik.haw-hamburg.de/ ubicomp/

bietet sich der Einsatz von Tabletops zur Betrachtung eingehender Fotos und Videos an, da diese besonders für die kooperative Zusammenarbeit mehrerer Personen geeignet sind, wie in Abschnitt 2.2.3 erläutert.

Damit die Medien-Dateien auf dem Tabletop schnell begutachtet und weitergegeben werden können, sollte die eingesetzte Software auf dem Tabletop möglichst intuitiv und effizient bedienbar sein. Die Entwicklung einer entsprechenden Anwendung unter dem Namen DynAmbient ist Ziel dieser Masterarbeit. Dabei sollen die, in Abschnitt 2.2.1 erläuterten, Ziele nahtloser Interaktion durch den Einsatz physikbasierter Interaktionstechniken (siehe Abschnitt 2.3.3) umgesetzt werden.

Die Bedienoberfläche des Programms soll ähnlich wie in Abb. 3.1 gestaltet sein: auf einem Tabletop werden mehrere digitale Bilder dargestellt, die sich auf einer horizontalen Ebene befinden. Die optische Erscheinung des Programms gleicht damit einer Tischfläche auf der gedruckte Fotos liegen.

Abb. 3.1: Beispielhafte Applikation zur Betrachtung digitaler Fotos auf einem Tabletop. Durch Drehen und Verschieben können die Fotos von jeder Tischposition aus betrachtet werden.

Die geplante DynAmbient-Applikation ähnelt dem von Hollatz (2007) entwickelten Leuchttisch, konzentriert sich jedoch auf die Bereitstellung physikbasierter Interaktionstechniken

zur simultanen Rotation und Translation von Fotos und Videos. Die Anwendung soll primär zur Sichtung von Medien-Dateien und nicht zur Layout-Erstellung dienen, wie dies beim Leuchttisch der Fall ist.

Eine detailierte Beschreibung der Anforderungen, die an DynAmbient gestellt werden, folgt in den anschliessenden Abschnitten 3.2 und 3.3.

3.2 Funktionale Anforderungen

In diesem Abschnitt werden zunächst in 3.2.1 mehrere Anwendungsfälle beschrieben, die bei der Nutzung von DynAmbient auftreten. Aus den Anwendungsfällen werden anschliessend in 3.2.2 konkrete Interaktionstechniken abgeleitet.

3.2.1 Anwendungsfälle

Im eben beschriebenen Notfall-Szenario sollen mehrere Personen an einem Tabletop mit Hilfe einer Software-Anwendung fortlaufend eingehende Fotos und Videos begutachten und auf deren Basis gemeinschaftlich Entscheidungen treffen. Wie in Abschnitt 2.2.3 erläutert, lassen sich bei kollaborativer Gruppenarbeit an Tischen bestimmte Verhaltensmuster häufig beobachten. Übertragen auf das Notfall-Szenario, kann man davon ausgehen, dass bestimmte Anwendungsfälle ebenfalls bei der kollaborativen Nutzung von DynAmbient gehäuft auftreten. Aufgrund der Aufgabenstellung mit der die Einsatzkräfte konfrontiert sind, lassen sich mehrere Anwendungsfälle unterscheiden. Die Bezeichnung „Datenobjekt“ wird dabei als Oberbegriff für digitale Bild- und Video-Dateien verwendet.

Datenobjekt heranholen Dem Anwender wird ein entferntes Foto oder Video übergeben bzw. greift er danach und holt es heran.

Datenobjekt betrachten Details können vom Anwender dabei durch Vergrösserung genauer betrachtet werden.

Datenobjekt zeigen Ein Foto oder Video wird z. B. im Rahmen einer Diskussion zu den Mitarbeitern am Tisch rotiert (vgl. Abschnitt 2.3.1).

Datenobjekt an andere Person übergeben Ist notwendig wenn ein Foto oder Video von einem Mitarbeiter betrachtet werden soll.

Datenobjekt beiseite legen Ein Foto oder Video, das zu einem späteren Zeitpunkt relevant sein könnte, wird beiseite gelegt.

Datenobjekt weiterleiten Ein Foto oder Video wird vom Leitstand an andere Hilfsdienste (z. B. Sanitäter, Polizei) gesendet.

Datenobjekt ablegen Zur Archivierung wird ein Foto oder Video abgespeichert und verschwindet dabei von der Bedienoberfläche.

Nachdem ein Datenobjekt von DynAmbient empfangen wurde und angezeigt wird, können die Anwender damit interagieren. Die dabei vorhandenen Übergänge zwischen den beschriebenen Anwendungsfällen sind in Abb. 3.2 in Form eines Aktivitätsdiagramms dargestellt.

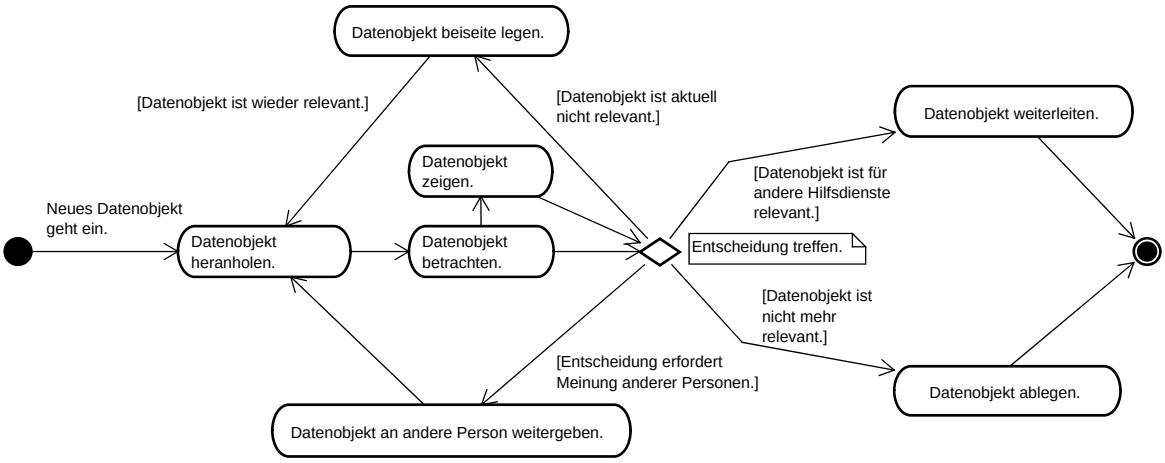


Abb. 3.2: Übergangsmöglichkeiten zwischen den Anwendungsfällen von DynAmbient.

Aus dem Diagramm wird ersichtlich, dass die Handlungsreihenfolge

Datenobjekt heranholen → *Datenobjekt betrachten*

eine zentrale Rolle spielt, da anzunehmen ist, dass diese bei jedem neu eingegangenen Foto bzw. Video ausgeführt wird, bevor eine Entscheidung[2] getroffen werden kann. Daher wird bei der Benutzung von DynAmbient vermutlich ein Großteil der Arbeitszeit auf das Heranholen und vor allem Betrachten von Datenobjekten entfallen.

Unter dem Aspekt eingesetzter Interaktionstechniken fällt auf, dass bei allen Anwendungsfällen, außer *Datenobjekt betrachten* und *Datenobjekt zeigen*, das Foto bzw. Video bewegt wird. Eine Rotation des Datenobjekts ist bei den Anwendungsfällen *Datenobjekt zeigen*, *Datenobjekt heranholen* und *Datenobjekts an andere Person weitergeben* wahrscheinlich, da sich die kollaborierenden Personen am verschiedenen Tischseiten befinden können. Die Reskalierung des Datenobjekts ist lediglich beim Anwendungsfall *Datenobjekts betrachten*

[2] im Diagramm als Raute dargestellt

sinnvoll. Zusammengefasst sollte dem Anwender das Bewegen und Rotieren von Fotos und Videos besonders einfach und intuitiv gestaltet werden, da diese Interaktionstechniken vermutlich sehr häufig genutzt werden.

3.2.2 Interaktionstechniken

Wie im vorherigen Absatz erläutert, beinhalten die sieben identifizierten Anwendungsfälle bestimmte Interaktionstechniken, wie Verschieben, Rotieren und Skalieren (siehe Abb. 3.3). Wie in Abschnitt 2.3.3 erläutert, ermöglichen physikalische Simulationverfahren eine simultane Rotation und Bewegung von Objekten in Computerprogrammen. Wie in Abschnitt 2.3.3 beschrieben, kann eine Verbindung beider Techniken zu einem geringeren Arbeitsaufwand bei der Translation von Objekten beitragen und bietet sich daher zur Verwendung bei allen genannten Anwendungsfällen an. Zusammenfassend lassen sich folgende Interaktionstechniken unterscheiden:

1. Datenobjekt skalieren
2. Datenobjekt verschieben
3. Datenobjekt rotieren
4. Datenobjekt verschieben und gleichzeitig rotieren

Die Interaktionstechniken *Datenobjekt verschieben*, *Datenobjekt rotieren* und *Datenobjekt verschieben und gleichzeitig rotieren* können ebenfalls bei der Betrachtung von gedruckten Bildern auf einer Tischfläche genutzt werden. Daher eignet sich DynAmbient als Grundlage für die Implementation nahtloser Interaktionstechniken, weil das, in Abschnitt 2.2.1 erläuterte, Ziel *Continuity* als Richtlinie bei der Programmentwicklung genutzt werden kann.

Zum besseren Verständnis, werden die genannten Interaktionstechniken in den folgenden Abschnitten 3.2.2 bis 3.2.2 einzeln erläutert.

Datenobjekt skalieren

Wie in Abb. 3.4 a) gezeigt, erfolgt die Skalierung (Vergrößerung bzw. Verkleinerung) eines Datenobjekts mit Hilfe von zwei Berührungspunkten. Das Datenobjekt kann durch Spreizung der Kontaktpunkte vergrössert werden. Eine Verkleinerung erfolgt analog durch eine Bewegung der Kontaktpunkte zueinander. Diese Interaktionstechnik wird auch bei der Powerwall-Applikation von Perceptive Pixel (2007) und dem iPhone von Apple Inc. (2007) verwendet.

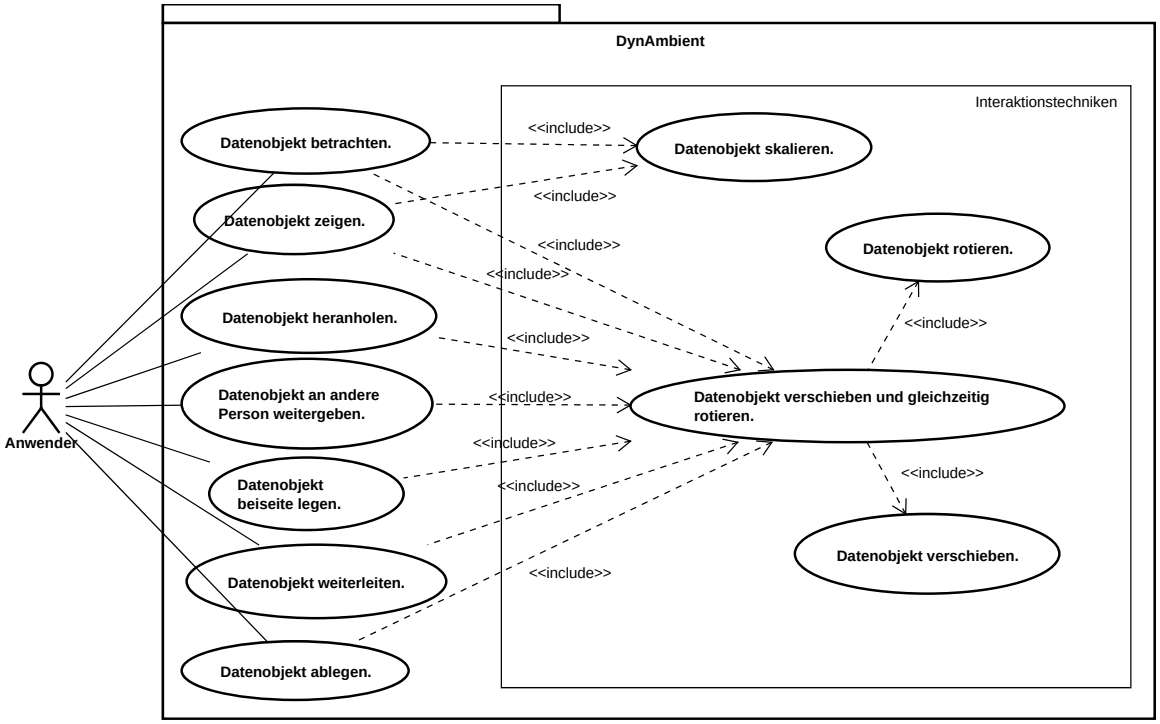


Abb. 3.3: Anwendungsfälle von DynAmbient und zugehörige Interaktionstechniken.

Datenobjekt verschieben

Um ein Datenobjekt zu verschieben, führt der Anwender, wie in Abb. 3.4 b) zu sehen, eine geradlinige Fingerbewegung aus. Die Reaktion des Datenobjekts auf die Bewegung wird wie bei den anderen Interaktionstechniken mit Hilfe einer physikalischen Simulation berechnet. Daher hat der Anwender bei der Wahl des Kontaktpunktes zwei Möglichkeiten um sicher zu stellen, dass das Datenobjekt nur verschoben und nicht gleichzeitig rotiert wird (vgl. Interaktionstechnik 3.2.2):

1. Der Kontaktpunkt befindet sich im Mittelpunkt des Datenobjekts, da bei einer quaderförmigen, homogenen Masseverteilung der Schwerpunkt im geometrischen Zentrum liegt.
2. Der Kontaktpunkt liegt auf der virtuellen Geraden, die vom Zielpunkt der Bewegung und dem Mittelpunkt des Datenobjekts gebildet wird.

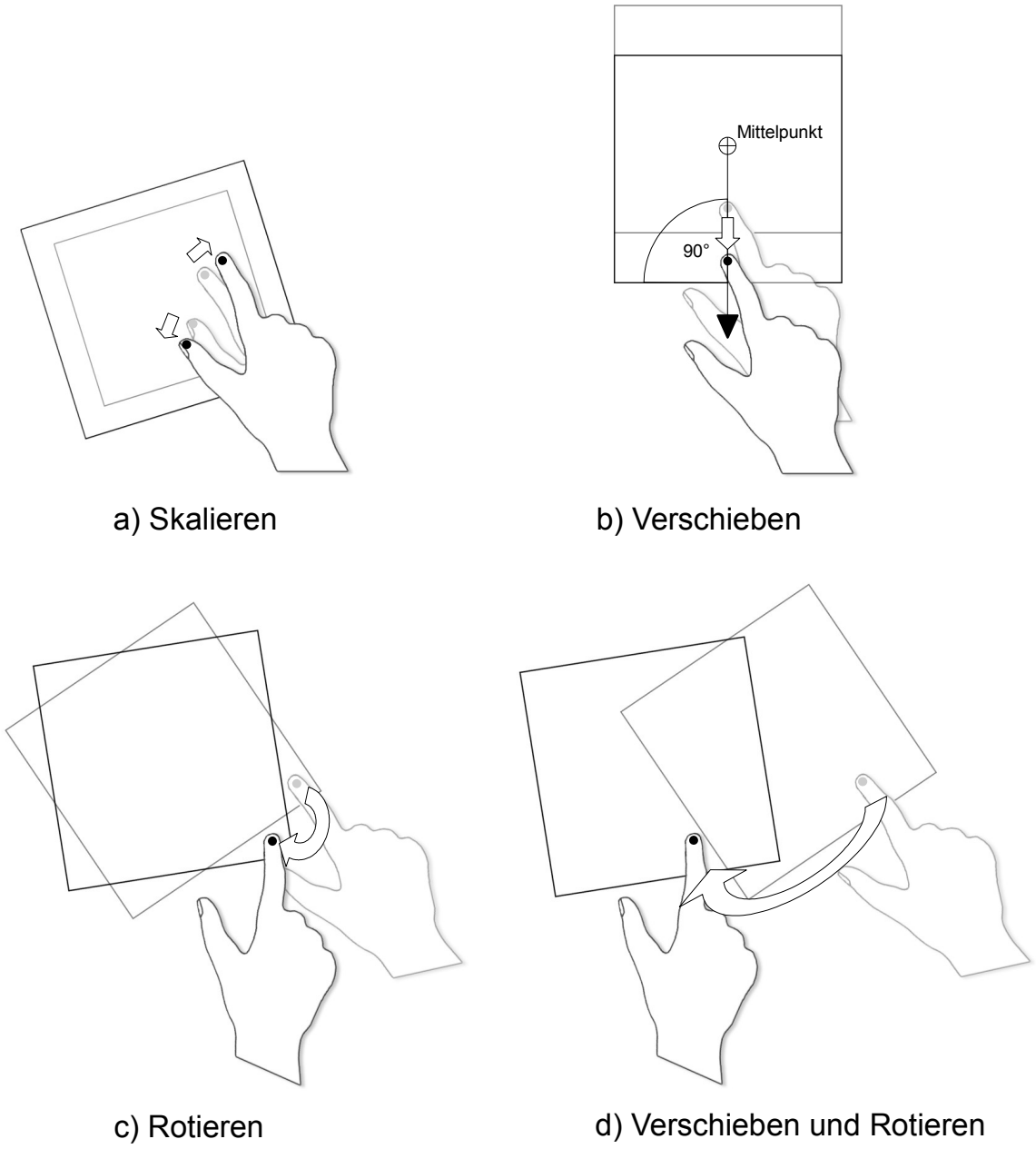


Abb. 3.4: Mögliche Interaktionstechniken bei der Bedienung von DynAmbient. Die Ausgangspositionen der Fingerbewegungen sind in grau eingezeichnet, die Endpositionen in schwarz. Die ausgefüllten Kreise in den Fingerspitzen markieren die Kontaktpunkte der Finger auf der Touchoberfläche.

Datenobjekt rotieren

Wie in Abb. 3.4 c) zu sehen, erfolgt der Ablauf dieser Interaktionstechnik wie beim eben beschriebenen Verschieben eines Fotos. Der Unterschied zwischen beiden Interaktionstechniken liegt in der Bewegung des Fingers: beim Rotieren des Fotos bzw. Videos wird dieser kreisförmig und nicht geradlinig bewegt.

Datenobjekt verschieben und gleichzeitig rotieren

Diese Interaktionstechnik beinhaltet das Verschieben und Rotieren des Datenobjekts: beide Interaktionstechniken werden simultan mit Hilfe einer Kombination aus geradliniger und kreisförmiger Fingerbewegung ausgeführt, wie Abb. 3.4 d) zeigt. Zur erfolgreichen Benutzung dieser Interkationstechnik, darf der Kontaktpunkt weder im Mittelpunkt des Datenobjekts noch auf der virtuellen Geraden liegen, die vom Zielpunkt der Bewegung und dem Mittelpunkt des Objekts gebildet wird.

Die eben beschriebene Interaktionstechnik erlaubt ein effizienteres Arbeiten bei der Benutzung von DynAmbient, da sie es erlaubt, das Verschieben und Rotieren von Datenobjekten in einem Handlungsschritt zu kombinieren. Wie von Buxton u. a. (1985) demonstriert wurde, neigen Anwender dazu, Interaktionstechniken zu parallelisieren, was zu signifikanten Geschwindigkeitsverbesserungen bei der Bedienung führt.

Eine entsprechende Verknüpfung dieser Interaktionstechniken ist bei vielen gegenwärtig verbreiteten Bildbetrachtungprogrammen nicht möglich. In dieser Hinsicht hebt sich die DynAmbient aufgrund der physikbasierten Interaktion positiv ab. Der folgende Abschnitt 3.2.3 befasst sich mit weiteren Vorteilen physikbasierter Interaktionstechniken.

3.2.3 Vorteile physikbasierter Interaktionstechniken

Die Reskalierung, Rotation und Bewegung von Bildern und Programmfenstern erfolgt bei einem Großteil aktuell verfügbarer Computersoftware mit Hilfe von Werkzeugen, wie in Abb. 3.5 gezeigt. Bei Betrachtung der Abbildung fällt auf, dass die Bedienelemente der Werkzeuge die grafischen Objekte „dekorieren“: ein Rahmen, der sensitive Bereiche (z. B. Quadrate an den Ecken) enthält, umgibt das eigentliche Objekt.

Dekorierende Bearbeitungswerkzeuge finden sich auch bei Software-Systemen, die speziell für Tabletop-Systeme mit Touch-Bedienung konzipiert sind, wie dem „DiamondSpin Tabletop Toolkit Project“ (DiamondSpin Projekt, 2006). Hierbei werden die Schaltflächen zum Wechsel zwischen Interaktionsmodi, wie *Skalieren* und *Rotieren*, einzelnen Fensterecken zugeordnet (siehe 3.6).

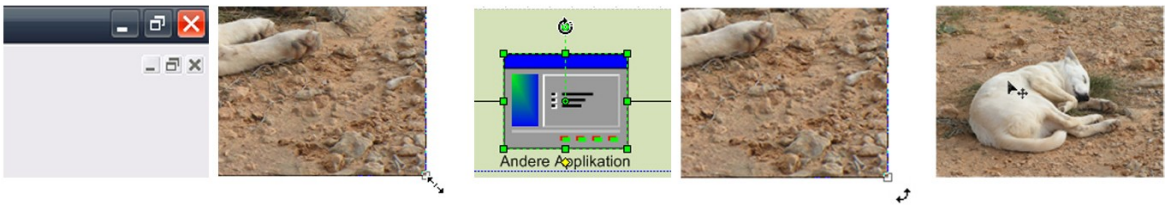


Abb. 3.5: Werkzeuge zur Manipulation von Fenstern und grafischen Objekten.

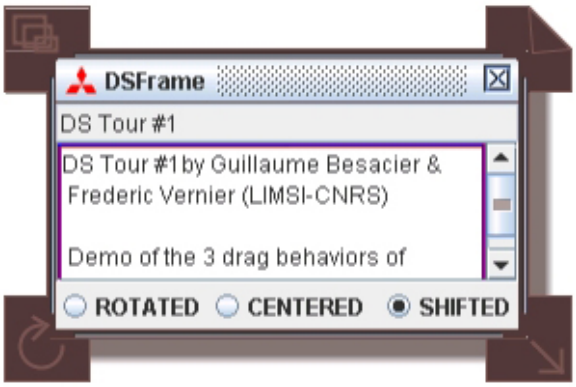


Abb. 3.6: Dekorierende Bedienelemente (braun) um ein Fenster bei DiamondSpin.

Der Einsatz von dekorierenden Bedienelementen bringt eine Reihe von Nachteilen mit sich. Zum einen ist ein Training ungeschulter Anwender nötig, da sich die Funktionalität der Bedienelemente nicht ohne Erklärung erschliesst. Ferner beansprucht die Darstellung der Werkzeuge einen Teil der Anzeigefläche. Dies führt zur Verdeckung anderer grafischer Elemente und kann besonders bei der Bearbeitung digitaler Bilder als störend empfunden werden. Darüber hinaus muss der Anwender zwischen den verfügbaren Interaktionsmodi wechseln: ein grafisches Objekt kann entweder nur gedreht oder nur verschoben werden, während eine Kombination beider Arbeitsvorgänge nicht möglich ist. Der Wechsel zwischen den Interaktionsmodi erhöht den Arbeitsaufwand und zwingt den Anwender zur linearen Abarbeitung einzelner Manipulationsschritte.

Demgegenüber weisen physikbasierte Manipulationsmöglichkeiten die beschriebenen Nachteile nicht auf. Die in Abschnitt 3.2.2 beschriebenen physikbasierten Interaktionstechniken gleichen hinsichtlich ihrer Anwendung Manipulationstechniken aus der Realwelt. Dadurch sind sie leichter zu erlernen und erfordern kaum Training. Ferner ist die Anzeige von dekorierenden Bedienelementen unnötig und es kommt daher nicht zu störenden Verdeckungen anderer grafischer Elemente. Außerdem können mehrere Arbeitsmodi in einem Schritt kombiniert werden, wie die simultane Rotation und Bewegung von Objekten (vgl. Abschnitt 3.2.2), ohne vorhergehenden Wechsel in einen bestimmten Interaktionsmodus nötig ist.

Ein weiterer wichtiger Vorteil physikbasierter Interaktionstechniken zeigt sich besonders deutlich in Kombination mit Touch-Oberflächen: Objekte können direkt „angefasst“ und so bewegt werden, wie es der Anwender beim Greifen nach einem Objekt in der Realwelt erwartet. Dadurch wird es möglich Mitarbeiter am Tisch aufzufordern Fotos und Videos „rüber zu werfen“, aufzufangen oder auf einem Haufen zu stapeln. Somit verbessert der physikbasierte Ansatz die Unmittelbarkeit der Bedienung deutlich, erhöht den Bedienkomfort und unterstützt den Austausch von Objekten, der bei kollaborativer Arbeit fortlaufend notwendig ist.

Das Verschieben von Objekten geschieht bei Desktop-Programmen häufig unter Verwendung der *Drag-and-Drop*-Methode. Damit können Objekte an jeder Stelle des Desktops platziert werden, die mit dem Mauszeiger erreichbar ist. In CCWs erfolgt die Interaktion mit Programmen häufig nicht per Maus sondern mit Hilfe von Touchscreens, wie bereits in Abschnitt 2.1 beschrieben. Dies kann bei großen Displays dazu führen, dass je nach Standort und physischer Konstitution des Anwenders bestimmte Teilbereiche der Anzeigefläche nicht mit dem Finger und daher auch nicht mit Cursor erreichbar sind. Um ein Objekt auf einem nicht erreichbaren Bereich zu platzieren, müsste sich der Anwender bei einem Touchscreen zum entsprechenden Anzeigebereich bewegen. Dies ist aus ergonomischen Gründen nicht praktikabel, da Aufgaben, die ein häufiges Verschieben von Objekten erfordern, für die Anwender einen hohen Bewegungsgaufwand bedeuten würden. Der Einsatz physikbasierter Interaktionstechniken erlaubt durch Anstoßen oder Werfen von Objekten diese auch in Bereichen zu platzieren, die sonst nicht erreichbar wären und verbessert damit den Bedienkomfort bei der Arbeit auf großen Anzeigeflächen.

Vom Anwender häufig benötigte Änderungen an grafischen Objekten, wie Rotieren und Verschieben, werden bei vielen Programmen durch die Auswahl vordefinierter Befehle ausgelöst. Dabei wird das bearbeitete Objekt in der Regel ohne Animation, in den Endzustand überführt. Der Aufruf des Kommandos „Drehung um 90 Grad nach rechts“ führt zu einem gedrehten Bild, wobei die Zwischenzustände der Rotation, wie „Bild um 5 Grad gedreht“, dem Anwender nicht gezeigt werden. Wie in Abschnitt 2.2.3 erläutert, ist die Beobachtung und Nachvollziehbarkeit von Arbeitsvorgängen anderer Mitarbeiter am Tisch bei kollaborativer Arbeit besonders wichtig. Physikbasierte Interaktionstechniken bieten den Vorteil, dass sie für Beobachter besser nachvollziehbar sind, da der optische Fluss bei ihrer Ausführung erhalten bleibt: bei Manipulationen sind alle Zustände zwischen der Ausgangs- und Endgestalt eines Objekts sichtbar. Wird ein Foto gedreht, sieht der Anwender die Rotation.

Natürlich lassen sich entsprechende Animationen auch in Programme mit nichtphysikbasierter Interaktion einbetten. Dabei ist es allerdings nötig die Animationen entweder manuell vor Ausführung des Programms zu erstellen oder mit Hilfe geeigneter Algorithmen während der Ausführung zu berechnen. Beide Ansätze bringen einen erhöhten Entwicklungsaufwand mit sich, der beim Einsatz physikalischer Simulation entfällt, da die Zwischenübergänge bei der Manipulation von Objekten ohnehin berechnet werden.

Dem Entwickler werden die Animationen damit „geschenkt“, was zur Verringerung der Entwicklungszeit beiträgt.

3.3 Nicht-funktionale Anforderungen

Nach der eben erfolgten Definition der funktionalen Anforderungen werden in diesem Teil der Arbeit nicht-funktionale Anforderungen an DynAmbient formuliert.

3.3.1 Mehrbenutzerfähigkeit

Wie in Abschnitt 2.2.2 erläutert, erlauben aktuelle Fortschritte bei der Entwicklung von Touch-Oberflächen, die Registrierung und Verfolgung mehrerer Berührungspunkte (Multi-Touch). Damit ist die Entwicklung von Software-Anwendungen möglich, die sich von mehr als einer Person gleichzeitig bedienen lassen bzw. bei deren Bedienung mehrere Finger verwendet werden können.

Da DynAmbient in einem Kontext eingesetzt wird, bei dem mehrere Personen zusammen arbeiten, sollte die simultane Bedienung der Anwendung durch mehrere Anwender möglich sein. Auf Seiten der Hardware kann diese Anforderung mit Hilfe der eben erwähnten Multi-Touch-Fähigkeit erfüllt werden. Softwareseitig müssen ebenso mehrere Beutzereingaben gleichzeitig verarbeitet werden können. Dabei muss sicher gestellt werden, dass DynAmbient beim simultanen Zugriff auf ein Objekt, wie dies z. B. beim gleichzeitigen Berühren eines Fotos durch zwei Personen auftritt, nicht in einen instabilen Zustand gerät, der zur ungewollten Beendigung des Programms (Absturz) führt.

3.3.2 Touch-gerechte Bedienung

Die Benutzung einer Maus erlaubt im Vergleich zu einer Touch-Oberfläche als Eingabegerät eine präzisere Ansteuerung von Bedienelementen, wie von Albinsson und Zhai (2003) nachgewiesen wurde. Wie Benko u. a. (2006) feststellen, enthalten Programme, die für die Bedienung per Maus entwickelt sind, teilweise sehr kleine Bedienelemente.

Wie im vorherigen Absatz erläutert, bietet sich zur Unterstützung der Anforderung *Mehrbenutzerfähigkeit* der Einsatz Multi-Touch fähiger Oberflächen an. Da die Bedienung der DynAmbient in diesem Fall primär über Touch erfolgt, muss die Bedienoberfläche der Anwendung entsprechend gestaltet sein: Bedienelemente müssen groß genug sein um von Anwendern ohne große Mühe angewählt werden zu können.

3.3.3 Performanz und Skalierbarkeit

Bei einem Notfall-Einsatz mit vielen Einsatzkräften, kann es vorkommen, dass sich eine große Anzahl von Fotos und Videos auf der Bedienoberfläche befindet. Da diese Datenobjekte beim Einsatz physibasierter Interaktionstechniken aufgrund ihrer physikalischen Eigenschaften (Abmessung, Gewicht) aufeinander reagieren, führt dies zu einer Vielzahl komplexer Berechnungen.

Da eine Verzögerung bei der Darstellung der Bedienoberfläche oder der Verarbeitung von Nutzereingaben aufgrund des Notfall-Szenarios nicht akzeptabel ist, müssen die eingesetzten Algorithmen bzw. Software-Bibliotheken zur Berechnung der physikalischen Simulation und der Darstellung der Anwenderobefläche performant genug sein um auch bei hoher Belastung eine Mindestreaktionszeit garantieren zu können.

4 Design und Realisierung

Auf Basis der formulierten Anforderungen in vorangegangenen Kapitel 3, wird in diesem Teil der Arbeit eine Software-Architetkur für DynAmbient entwickelt. Zunächst wird jedoch in 4.1 der Anwendungskontext vorgestellt und die grundlegende Funktionalität von DynAmbient in 4.2 erläutert. Nach Diskussion einer geeigneten Software-Architektur für die Anwendung in 4.3, erläutert Abschnitt 4.4 wie die Kommunikation von DynAmbient mit anderen Programmen im CCW erfolgen könnte. Auf Basis dieser Design-Überlegungen, erfolgte die gestalterische und technische Umsetzung von DynAmbient, die in 4.5 dargelegt und begründet wird. Das Ende dieses Kapitels bildet in Abschnitt 4.6 eine Bewertung der realisierten DynAmbient-Anwendung und ein Fazit hinsichtlich der erfüllten Anforderungen aus Kapitel 3.

4.1 Anwendungskontext

Die Systemarchitektur eines CCWs besteht aus einer Vielzahl von Komponenten, wie Computer, Monitore, Speicher-, Eingabe- und Netzwerkgeräten. Als Beispiele für CCW-Systemarchitekturen zeigen Abb. 4.1 die Gerätekonfiguration des iRoom der Universität Stanford und Abb. 4.2 die des Ambient Labors der HAW Hamburg. Die Abbildungen veranschaulichen die räumliche Anordnung der Geräte und im Falle des iRooms die Zuordnung der Anzeigeflächen zu Computersystemen.

Ebenso wie Anzeigeflächen sind Software-Anwendungen, die innerhalb des CCWs genutzt werden, in der Regel bestimmten Computersystemen zugewiesen. Dabei ist es sinnvoll Programme, die über eine grafische Benutzeroberfläche verfügen, aus Performanzgründen auf Computern auszuführen, die direkt mit den Anzeigeflächen verbunden sind. Dagegen lassen sich im Hintergrund laufende Applikationen ohne Bedienoberfläche auf Serversystemen betreiben, die sich über das Netzwerk erreichen lassen.

Prinzipiell können alle Soft- und Hardware-basierten Bestandteile des CCWs als Dienste betrachtet werden. So ist es vorstellbar, dass die Bedienoberfläche eines Programms auf einer beliebigen Anzeigefläche dargestellt und über ein oder mehrere Eingabegeräte im CCW

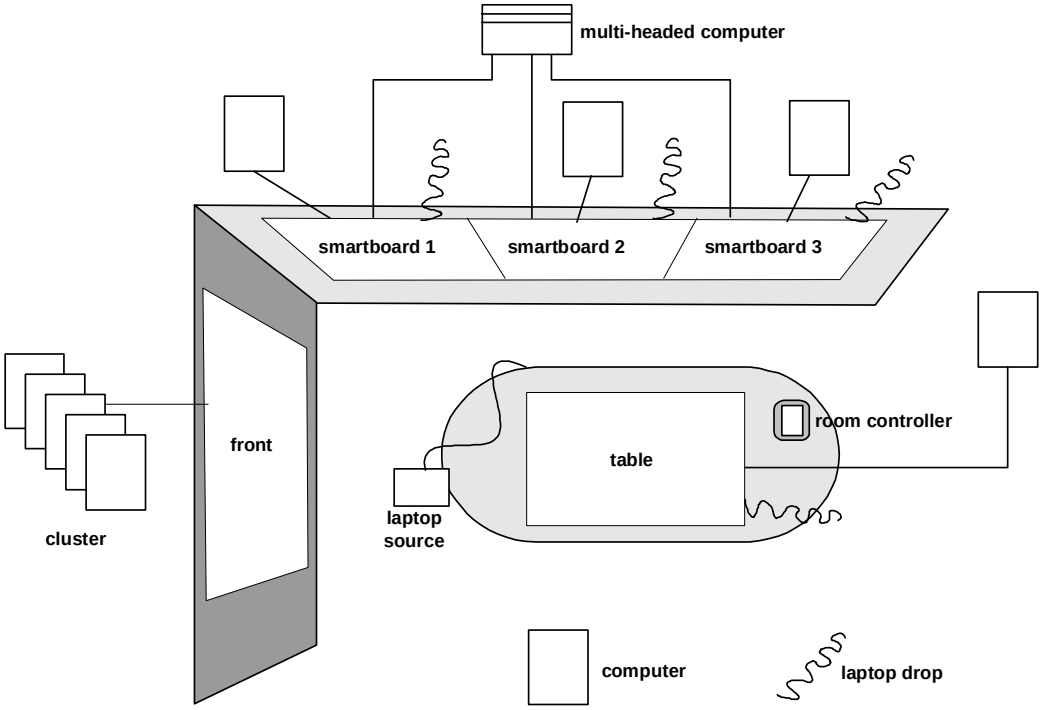


Abb. 4.1: Systemarchitektur des Stanford iRoom (von Johanson und Fox, 2002).

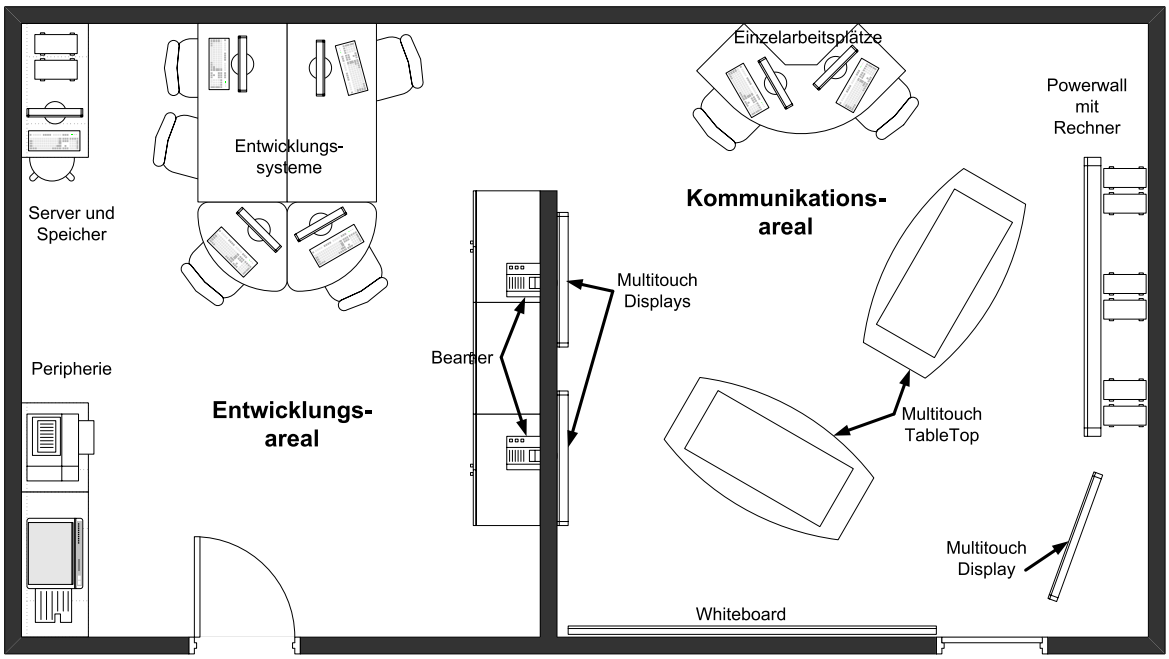


Abb. 4.2: Systemarchitektur des Ambient-Labors der HAW Hamburg (von Fischer, 2007).

bedient wird. In diesem Fall ist die Funktionalität des Programms ein Dienst, der vom Anwender genutzt wird. Eingabegeräte und Anzeigefläche stellen ebenso Dienste dar, die in Kombination mit dem „Programm-Dienst“ eingesetzt werden. Der CCW kann somit als serviceorientierte Umgebung bezeichnet werden. Weitere Überlegungen zu diesem Aspekt folgen in Abschnitt 4.4 dieses Kapitels.

DynAmbient stellt einen Dienst dar, der zur Betrachtung spezieller Datenobjekte, wie Fotos und Videos, konzipiert ist. Die Anwendung ist dabei nur eines von vielen Programmen aus der sich die Anwendungslandschaft des CCWs zusammensetzt. Es ist möglich, dass innerhalb der Anwendungslandschaft mehrere Software-Applikationen existieren, die unterschiedliche Sichten auf gleiche Datenobjekte bieten. Im Rahmen eines Notfall-Szenarios, wie es in Abschnitt 3.1 vorgestellt wurde, könnte z. B. auf einer vertikalen Anzeigefläche die Eingangshistorie aller Fotos und Videos im CCW durch ein spezielles Programm angezeigt werden, während deren Betrachtung auf einem Tabletop mit Hilfe eines anderen Programms wie DynAmbient stattfindet. Zwar unterscheiden sich beide Programme anhand ihrer Bedienoberfläche, greifen aber dennoch auf dieselben Datenobjekte zu.

Die simultane Nutzung von Datenobjekten durch mehrere Anwendungen wirft Fragen hinsichtlich der Konsistenz des Datenbestandes auf. Wie kann ein Programm von Änderungen an Datenobjekten durch andere Programme erfahren? Wie kann ein Programm seinerseits andere Programme über Änderungen informieren, die es selbst vornimmt?

Aus diesen Fragen ergeben sich Anforderungen an das Design von DynAmbient. Die Programmarchitektur der Anwendung muss so gestaltet sein, dass Änderungen, die durch externe Applikationen vorgenommen werden, empfangen und verarbeitet werden können. Ebenso müssen Änderungen an Datenobjekten, die durch Benutzereingaben erfolgen, an externe Applikationen weitergegeben werden. In diesem Kapitel wird eine geeignete Software-Architektur entwickelt, die neben den gestellten Anforderungen die grundlegende Funktionalität von DynAmbient bereitstellt, wie sie ihm folgenden Abschnitt 4.2 einführend beschrieben wird.

4.2 Grundlegende Funktionalität von DynAmbient

4.2.1 Bedienoberfläche und zentrale Programmeigenschaften

Abbildung 4.3 zeigt eine prototypische Bedienoberfläche von DynAmbient: mehrere rechteckige, grafische Objekte sind auf einer statischen horizontalen Fläche verteilt, die eine Holztextur[1] überzieht. Die Objekte überlappen einander teilweise und sind unterschiedlich

[1] In der Computergrafik versteht man unter einer Textur eine Grafik, die auf ein 3-dimensionales Objekt projiziert wird.

ausgerichtet. Einige stehen auf dem Kopf und könnten bei der Ausführung der Applikation auf einem Tabletop (vgl. Abb. 3.1) daher nur von einem bestimmten Standpunkt aus aufrecht betrachtet werden.

Abb. 4.3: Prototyp der DynAmbient-Bedienoberfläche.

Im Vorfeld dieser Masterarbeit fiel die Entscheidung die Modellierung der Objekte auf der horizontalen Fläche, ihre Reaktion auf einwirkende Kräfte und die physikbasierte Steuerung der Anwendung in DynAmbient nicht mit Hilfe selbstentwickelter Algorithmen umzusetzen, sondern durch die Verwendung einer Physik-Engine. Dieser Entschluss fiel aus mehreren Gründen. Zum einen hätte die Selbstentwicklung und die damit verbundene Einarbeitung in Fachmaterie einen erheblichen Mehraufwand bedeutet und wäre daher im Rahmen dieser Arbeit wahrscheinlich nicht möglich gewesen. Die Benutzung einer Physik-Engine ermöglichte stattdessen die Konzentration auf die eigentliche Anwendung. Ferner kann die DynAmbient-Anwenderoberfläche durch den Einsatz einer Physik-Engine flexibel weiter entwickelt werden, da die Anwendung nicht nur „maßgescheiderte“ Algorithmen enthält, sondern in Form einer physikalischen Simulation gestaltet ist.

Alle dargestellten Objekte in Abb. 4.3 sind mit bestimmten physikalischen Eigenschaften versehen. Neben Größe und Gewicht weisen diese, wie auch die darunter liegende horizontale

Fläche, eine spezifische Oberflächenbeschaffenheit auf, die in einer individuellen Haft- und Gleitreibung[2] resultiert.

Die Objekte auf dem Tisch repräsentieren Dateien, die von DynAmbient geladen werden. Der Anwender kann die Objekte bewegen, rotieren und skalieren, wie in Abschnitt 3.2.2 ausführlich beschrieben. Die Reaktion eines Objekts auf die Bewegung angrenzender Objekte und die Eingaben des Anwenders wird mit Hilfe einer so genannten Physik-Engine realitätsnah simuliert.

Die Darstellung der Objekte auf dem Tisch, der horizontalen Fläche und aller weiteren, für den Anwender sichtbaren Elemente, erfolgt mit Hilfe einer Software-Komponente, die es erlaubt 3-dimensionale Szenerien zu visualisieren. Im weiteren Verlauf dieser Arbeit wird diese Komponente als 3D-Engine bezeichnet.

Dem Anwender soll es durch Simulation physikalischer Kräfte möglich sein, physik-basiert mit den Objekten auf der horizontalen Fläche zu interagieren. Dazu wird jedes Datenobjekt, das die DynAmbient-Software lädt, in der Software durch eine physikalische und eine grafische Entität repräsentiert. Die physikalische Entität beinhaltet die physikalischen Eigenschaften des Objekts, während die grafische Entität das äußere Erscheinungsbild des Objekts beschreibt. Die physikalische Entität wird unter Verwendung der Physik-Engine erzeugt und verwaltet. Für die grafische Entität geschieht dies analog mit Hilfe der 3D-Engine.

Bei der Erzeugung der Entitäten in der 3D- und der Physik-Engine, fließen verschiedene Eigenschaften des Datenobjekts ein, wie Abb. 4.4 veranschaulicht. Die Objektabmessungen bestimmen die Größe beider Entitäten. Die grafische Information des Datenobjekts, ist dagegen nur für die grafische Entität relevant. Diese wird in Form einer Textur auf die Vorderseite der 3-dimensionalen Entität gelegt und so dem Anwender präsentiert. Das Aussehen der horizontalen Fläche, die sich in der Abbildung unter der grafischen Entität befindet, wird ebenfalls mit Hilfe einer Textur festgelegt. Weitere physikalische Parameter, wie Gewicht und Oberflächenbeschaffenheit, sind in DynAmbient einheitlich definiert und werden nicht aus den geladenen Datenobjekten extrahiert.

Die 3D- und Physik-Engine bilden zentrale Bestandteile von DynAmbient. Bei der Ausführung der Software wird die Lage und Position aller physikalischen Entitäten in regelmässigen Abständen von der Physik-Engine neu berechnet. Die errechneten Werte werden anschliessend auf die grafischen Entitäten übertragen, die dann in ihrer aktualisierten Position und Lage von der 3D-Engine dargestellt werden. Diesen Ablauf, der sich fortlaufend bei der Ausführung von DynAmbient wiederholt, illustriert Abb. 4.5 in Form eines Aktivitätsdiagramms.

[2] Unter Haftreibung versteht man die Kraft, die dem Verschieben eines ruhenden Körpers entgegenwirkt. Sobald der Körper in Bewegung gesetzt ist, wirkt die Gleitreibung gegen das weitere Verschieben.

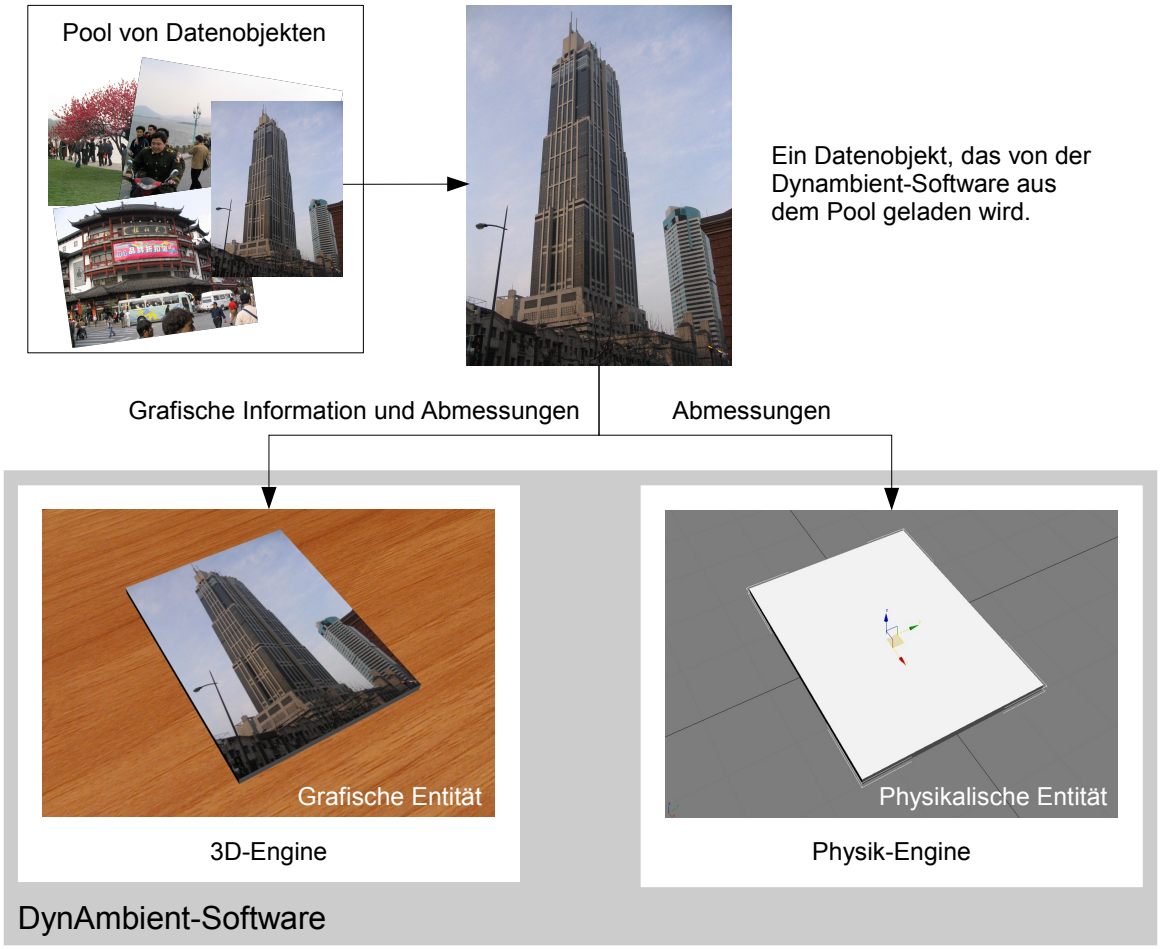


Abb. 4.4: Entitäten, die von DynAmbient beim Laden eines Datenobjekts erzeugt werden.

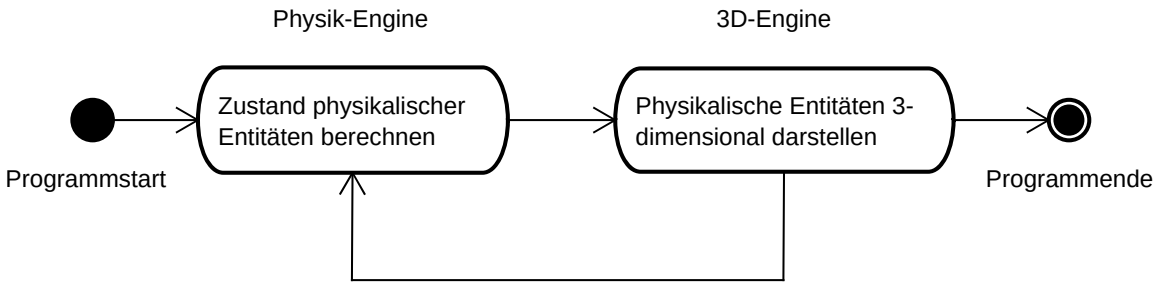


Abb. 4.5: Berechnungsschleife bei der Ausführung von DynAmbient.

4.2.2 Annotation von Datenobjekten

Mit jedem Projekt, das im Rahmen eines CCW bearbeitet wird, ist in der Regel eine Sammlung von Datenobjekten (Dokumente, Bilder, Videos, etc.) verbunden. Diese Objekte bilden das zentrale Arbeitsmaterial, welches den Anwendern mit Hilfe von Programmen und verschiedenartigen Anzeigeflächen visuell zur Verfügung gestellt wird.

Ein Vorteil, der mit der Digitalisierung des Arbeitsmaterials einhergeht, ist die Möglichkeit jedes Datenobjekt mit Metadaten[3] wie Anmerkungen, Projektzugehörigkeit, Versionsnummer, Autor, Bearbeitungsstand etc., zu annotieren. Mit Hilfe der Annotationen kann z. B. gezielt nach einer bestimmten Untermenge von Informationsobjekten gesucht werden. Ferner ist es möglich mit Hilfe geeigneter Computer-Programme automatisiert sicher zu stellen, dass alle Personen, die an einem Projekt beteiligt sind, stets über die aktuellste Version aller relevanten Informationsobjekte verfügen.

Im Falle des Notfall-Szenarios erscheint eine Annotation von Fotos und Videos mit bestimmten Metadaten sinnvoll. Dazu zählen folgende Informationen:

- Ort (Koordinaten) und Zeit der Aufnahme
- Name des Fotografen
- Notizen zum Inhalt des Fotos bzw. Videos
- Schlagwörter unter den das Datenobjekt eingeordnet werden kann (z. B. Feuer, einsturzgefährdet ,verletzte Personen, etc.)
- Einsatzzugehörigkeit
- Informationen zu reskalierten Versionen des Datenobjekts (z. B. Thumbnails), die zur schnellen Vorschau nötig sind

Die Speicherung der aufgezählten Metadaten kann auf verschiedene Weise erfolgen. Eine Möglichkeit ist die Nutzung einer Datenbank, die jedes Datenobjekt mit den zugehörigen Metadaten in einem Tupel speichert. Alternativ könnte die Speicherung der Metadaten auch mit Hilfe von Textdateien z. B. im XML-Format realisiert werden. In diesem Fall würde zu jedem Bild eine entsprechende Datei mit Metadaten exisitieren. Eine weitere interessante Variante besteht in der direkten Einbettung der Metadaten in Datenobjekte. Für Digitalfotos exisitieren zu diesem Zweck unterschiedliche Spezifikationen wie EXIF (JEITA, 2002) und IPTC-NAA (1999). Ähnlich weit verbreitete Standards sind für Video-Daten bisher nicht vorhanden. Es existieren jedoch unterschiedliche Bemühungen zur Etablierung von Beschreibungsformaten für Videos wie in Wactlar und Christel (2002) erläutert wird.

[3] Unter Metadaten versteht man allgemein Daten, die andere Datenobjekte anhand von Informationen beschreiben.

Die Annotation von Fotos und Videos mit Metadaten in DynAmbient versorgt Anwender mit zusätzlichen Informationen zu einzelnen Datenobjekten und ermöglicht eine bessere Sortierung der Inhalte. Diese ursprünglich geplante Funktionalität konnten allerdings aus zeitlichen Gründen nicht implementiert werden und wird deshalb im folgenden Abschnitt 4.3, der sich mit der Software-Architektur von DynAmbient befasst, nicht berücksichtigt.

4.3 Software-Architektur von DynAmbient

DynAmbient erfüllt drei primäre Aufgaben:

1. Darstellung von Datenobjekten (Fotos und Videos) in Form einer grafischen Bedienoberfläche
2. Annahme und Interpretation von Benutzereingaben
3. Verwaltung und Veränderung von Datenobjekten und deren Eigenschaften

DynAmbient muss folglich Funktionsmodule beinhalten, die diese primären Aufgaben übernehmen. Ein Teil der Anwendung ist damit für die Darstellung der Datenobjekte zuständig. Ein anderer akzeptiert und interpretiert Benutzereingaben. Ein dritter verwaltet Datenobjekte und deren Eigenschaften. Die Gesamtapplikation lässt sich damit in Verantwortungsbereiche gliedern, die es ermöglichen, die Programmarchitektur von DynAmbient zu definieren.

Die Modularisierung von Verantwortlichkeiten ist ein wesentliches Ziel bei der Entwicklung von Software. Nach dem Prinzip *separation of concerns* von Dijkstra (1982) wird versucht den Quellcode von Computerprogrammen anhand spezifischer Aspekte und Funktionalitäten aufzugliedern. Dies dient unter anderem der einfacheren Identifikation von Programmfehlern, der Minimierung redundanter Code-Teile und der besseren Verständlichkeit der gesamten Software-Architektur.

Ein Entwurfsmuster der Informatik, das häufig zur Trennung der beschriebenen Verantwortungsbereiche (Darstellung, Interpretation von Benutzereingaben, Verwaltung von Datenobjekten) eingesetzt wird, ist das MVC[4]-Schema (Gamma u. a., 1995). Dieses wird im folgenden zur Gestaltung der Programmarchitektur von DynAmbient verwendet. In den nächsten Abschnitten wird zunächst das MVC-Entwurfsmuster vorgestellt und im Anschluss auf die Architektur von DynAmbient angewandt.

[4] MVC = Model, View, Controller

4.3.1 Varianten des MVC-Entwurfsmusters

Das MVC-Entwurfsmuster setzt sich aus drei Kernkomponenten (Burbeck, 1992) zusammen:

Model Die Model-Komponente verwaltet die Daten der Anwendung und liefert diese größtenteils an die View-Konponente. Ferner nimmt sie aufgrund von Instruktionen, die von der Controller-Komponente gesendet werden, Änderungen an den Anwendungsdaten vor. Im folgenden wird unter Model die Gesamtheit der Anwendungsdaten verstanden.

View Die View-Komponente ist für die grafische Präsentation der Anwendungsdaten zuständig, die von der Model-Komponente verwaltet werden.

Controller Die Controller-Komponente interpretiert Interaktionen des Anwenders, die mit Hilfe von Eingabegeräten wie Maus, Tastatur oder Touchscreen erzeugt werden. Wenn Anwendungsdaten oder deren Darstellung sich aufgrund von Benutzereingaben ändern sollen, informiert der Controller die Model- und View-Komponente.

Die Kommunikationsrichtung zwischen den Komponenten zeigt das linke UML-Diagramm in Abb. 4.6. Dabei ist hervorzuheben, dass die Controller- und View-Komponenten die Model-Komponente nutzen, während diese selbst nicht auf die anderen Komponenten zugreift. Diese Entkoppelung hat den Vorteil, dass die Model-Komponente unabhängig von den beiden anderen Komponenten entwickelt und getestet werden kann.

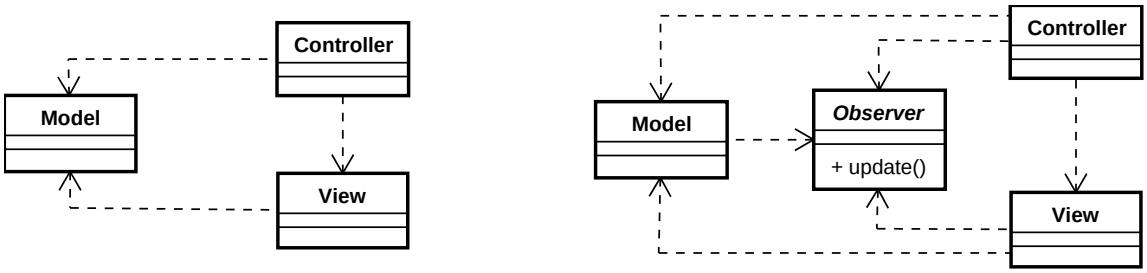


Abb. 4.6: MVC-Klassenstruktur bei passiver (links) und aktiver Model-Komponente mit Observer-Muster.

Von Burbeck (1992) werden zwei Varianten des MVC-Musters beschrieben. Bei der bisher erläuterten Variante mit passiver Model-Komponente, werden Änderungen am Model ausschließlich auf Anweisung der Controller-Komponente vorgenommen. Die Kommunikation zwischen den Komponenten verläuft, wie in Abb. 4.6 dargestellt, einseitig von Controller und View in Richtung Model.

Im Kontext eines CCWs mit verschiedenen Anwendungen (siehe Abschnitt 4.1), die auf identische Datenobjekte zugreifen, können Änderungen am Datenbestand jedoch auch extern

und ohne Einbeziehung der Controller-Komponente erfolgen. Im diesem Fall muss das Model die Controller- und die View-Komponente über entsprechende Änderungen informieren. Die Kommunikation zwischen den Komponenten erfolgt dabei im Gegensatz zur passiven Variante bidirektional. Diese Variation wird als MVC-Muster mit aktiver Model-Komponente bezeichnet.

Wie zu Beginn dieses Abschnitts erläutert, ist die Sicherstellung der Unabhängigkeit des Models von den anderen beiden Komponenten ein Hauptgrund für den Einsatz des MVC-Musters. Bei der Variante mit aktiver Model-Komponente entstehen jedoch ungewollte Abhängigkeiten, wenn die Benachrichtigung von View- und Controller-Komponente direkt durch die Model-Komponente erfolgen würde. Um dies zu vermeiden empfiehlt sich der Einsatz des Observer[5]-Musters (Gamma u. a., 1995).

Beim Observer-Muster implementieren alle Komponenten, die über Zustandsänderungen am Datenbestand der Anwendung informiert werden wollen, das abstrakte Observer-Interface und registrieren sich bei der Model-Komponente. Sobald Änderungen am Datenbestand auftreten, informiert die Model-Komponente alle Komponenten, indem sie über die Liste der registrierten „Beobachter“ iteriert. Der Vorteil bei diesem Ansatz besteht darin, dass die Model-Komponente keine Kenntnis über die registrierten Komponenten oder deren Logik haben muss. Durch den Einsatz des Observer-Musters kann damit die angestrebte Unabhängigkeit zwischen dem Model und der Controller- und View-Komponente erhalten werden. Im rechten UML-Diagramm in Abb. 4.6 ist die Kommunikationsrichtung zwischen den Komponenten bei Verwendung des Observer-Musters dargestellt.

4.3.2 Programmstruktur von DynAmbient

Wie im vorangegangenen Abschnitt 4.3.1 erläutert, eignet sich das MVC-Muster mit aktiver Model-Komponente besonders für Anwendungen bei denen Änderungen am Datenbestand durch externe Anwendungen erfolgen können. Da DynAmbient in einem entsprechenden Kontext ausgeführt werden soll, basiert die Architektur von DynAmbient auf dieser MVC-Variante.

Unter Berücksichtigung der MVC-Funktionsverteilung lassen sich die Bestandteile von DynAmbient wie folgt aufgliedern:

Model Die Anwendungsdaten von DynAmbient stellen Datenobjekte mit bestimmten physikalischen Eigenschaften wie Gewicht, Größe etc. dar. Die Position und Ausrichtung von Fotos und Videos wird zur Darstellung an die View-Komponente übertragen, während die Repositionierung der Bilder aufgrund von Benutzereingaben erfolgt, die von der Controller-Komponente empfangen werden.

[5] zu dt. Beobachter

View Die View-Komponente stellt den virtuellen Tisch und die darauf liegenden Datenobjekte dar. Die aktuelle Lage, Größe und Ausrichtung jedes Fotos oder Videos wird durch Anfrage an die Controller-Komponente bestimmt.

Controller Durch den Einsatz von Eingabegeräten wie Maus, Touchscreen und Tastatur kann der Anwender die Position, Ausrichtung und Größe von Datenobjekten verändern. Die Interpretation der Benutzereingaben und deren Anwendung auf das Model ist Aufgabe der Controller-Komponente.

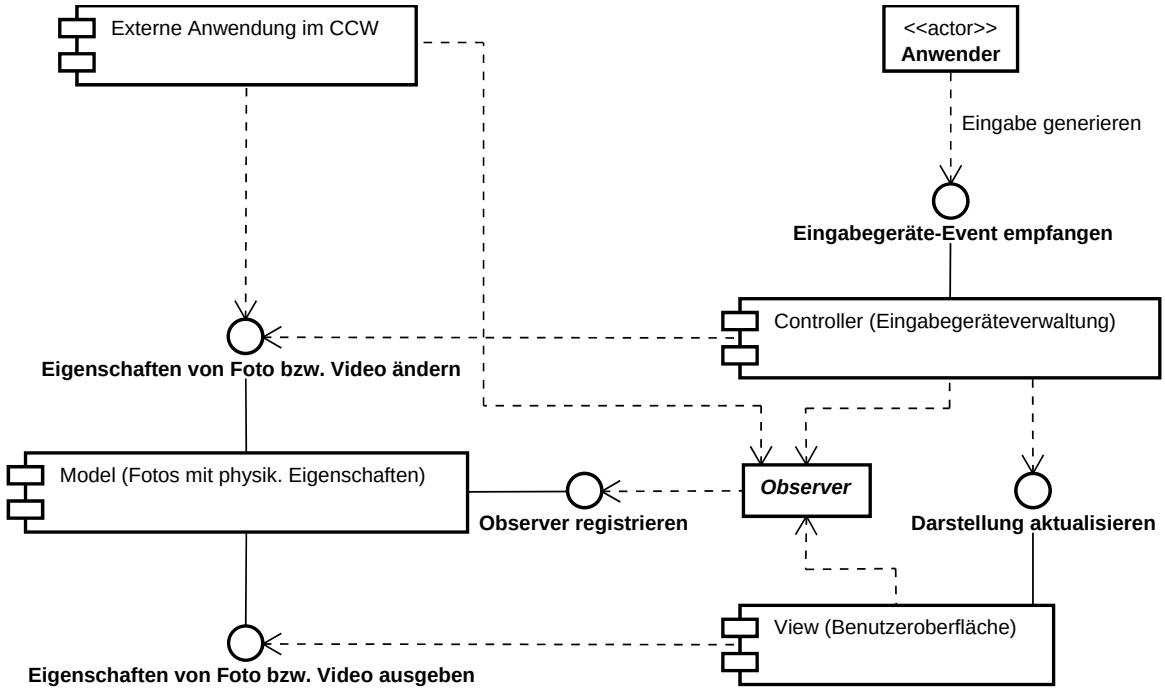


Abb. 4.7: Komponentendiagramm von DynAmbient.

Abbildung 4.7 zeigt das zugehörige Komponentendiagramm von DynAmbient nach dem MVC-Muster mit aktiver Model-Komponente. Wie dargestellt, ist die Model-Komponente in der Lage Änderungen am Zustand eines Fotos oder Videos aufgrund von Benachrichtigungen des Controllers als auch externer Anwendungen durchzuführen. Durch Einsatz des Observer-Muster, werden registrierte Komponenten, darunter auch externe Anwendungen, über Änderungen am Datenbestand informiert.

4.3.3 Funktionsablauf bei Ausführung eines Anwendungsfalls

Der Funktionsablauf bei der Durchführung einer Modifikation am Zustand eines Fotos oder Videos illustriert Abb. 4.8. Dabei wird aus Gründen der Übersichlichkeit die View-

Komponente repräsentativ als Empfänger aktualisierter Daten dargestellt. Alternative Empfänger stellen die Controller-Komponente und externe Programme dar.

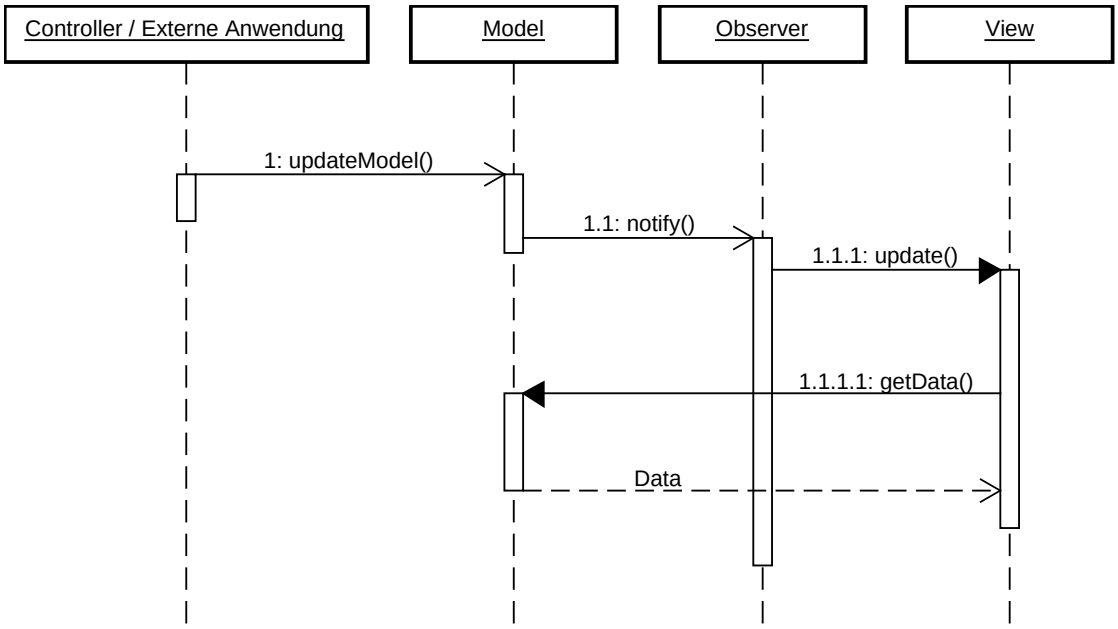


Abb. 4.8: Sequenzdiagramm zum Verhalten von DynAmbient bei Zustandsänderungen des Model.

Der gezeigte Funktionsablauf erfolgt unter anderem bei Ausführung der Anwendungsfälle, die in Abschnitt 3.2.1 vorgestellt werden. Betrachtet man z. B. Anwendungsfall *Datenobjekt heranholen*, so würde die Bewegung eines Fotos oder Videos den geschilderten Funktionsablauf auslösen, weil sich dabei die Position des Datenobjekts und damit der Zustand des Models ändert.

Die Benachrichtigung interner Programmkomponenten über Zustandsänderungen an Datenobjekten, wie Position und Lage, ist für die korrekte Darstellung nötig und hinsichtlich der Ausführungsgeschwindigkeit möglich. Die Übermittlung dieser Informationen über ein Netzwerk zu externen Anwendungen bringt jedoch hohe Belastungen für die Netzwerkinfrastruktur und die kommunizierenden Programme mit sich. Ferner erreichen die Informationen externe Anwendungen nur verspätet und werden ohnehin nicht benötigt, wenn eigene grafische Bedienoberflächen zur Betrachtung von Datenobjekten in die Anwendungen implementiert sind.

Daher ist es sinnvoll nur bestimmte Änderungen am Zustand des Models bzw. der darin enthaltenen Datenobjekte zu übertragen. Im Falle von DynAmbient könnte es sich um Informationen handeln, die selten geändert werden oder eine höhere Relevanz für externe

Anwendungen haben. Dazu zählt die Annotation eines Fotos bzw. Videos oder das Löschen eines Datenobjekts. Die Filterung der Kommunikation von Zustandsänderungen kann z. B. durch die Einführung von Subjekten (Microsoft Corporation, 2007b) gelöst werden, für die sich Komponenten und externe Anwendungen registrieren können.

4.4 Dienstbasierte Anwendungskommunikation

Wie am Anfang dieses Kapitels in 4.1 erwähnt, spielt die Sicherstellung der Datenkonsistenz innerhalb eines CCWs eine wichtige Rolle, wenn mehrere Anwendungen auf gleiche Datenobjekte zugreifen. Dazu diskutiert der vorangegangene Abschnitt 4.3, wie die interne Architektur von DynAmbient mit Hilfe des MVC-Entwurfsmuster gestaltet werden kann, damit Veränderungen an Datenobjekten kommuniziert und empfangen werden können. Bisher wurde allerdings nicht betrachtet, wie die Übertragung der Informationen zwischen Anwendungen erfolgt und der Zustand der Datenobjekte verwaltet werden kann, so dass deren Konsistenz für alle Programme im CCW gesichert ist.

In den letzten Jahren haben serviceorientierte Architekturen (SOA) im Umfeld geschäftlicher Anwendungslandschaften zunehmend an Bedeutung gewonnen. Häufig werden auch eine Reihe von Technologien wie BPEL, ESB oder Web-Services unter dem Begriff SOA subsummiert. Im folgenden wird dieser Begriff jedoch nur als Synonym für einen serviceorientierten Software-Architekturstil verwendet. Eine ausführlichere Beschreibung und Differenzierung des Begriffs findet sich bei Richter u. a. (2005) und Melzer (2007).

Ein Kernaspekt von SOA ist die Kapselung von Programmfunktionaliäten in Form von Diensten. Diese lassen sich miteinander kombinieren um komplexe Abläufe, wie z. B. die Abwicklung eines Bestellvorgangs, zu modellieren (siehe Erl, 2004).

Ein weiteres wichtiges Konzept serviceorientierter Architekturen ist die lose Koppelung von Anwendungen. Unter loser Koppelung versteht man in diesem Zusammenhang die Minimierung von Abhängigkeiten bei der Ausführung kommunizierender Programme. Um dies zu erreichen erfolgt die Kommunikation anhand definierter Schnittstellen, die von Anwendungen veröffentlicht werden.

Das SOA-Konzept eignet sich zur Anwendung auf das beschriebene CCW-Szenario, wobei mehrere Programme miteinander kommunizieren und auf gleiche Datenobjekte zugreifen. Die Kommunikation der Anwendungen untereinander kann dabei lose gekoppelt über definierte Schnittstellen erfolgen. Ferner lässt sich die Verwaltung der Datenobjekte nach dem SOA-Ansatz in Form eines Dienstes realisieren, der folgende Aufgaben übernimmt:

- Registrieren neuer Datenobjekte

- Löschen von Datenobjekten
- Senden des Zustands eines Datenobjekts
- Empfangen von Änderungen am Zustand eines Datenobjekts

Das Verteilungsdiagramm in Abb. 4.9 zeigt, wie der Verwaltungsdienst und DynAmbient innerhalb der Systemarchitektur eines CCWs verteilt werden können. Die Darstellung zeigt mehrere serverseitige Dienste, die von Anwendungen auf Client-Systemen genutzt werden. DynAmbient greift dabei über den Verwaltungsdienst auf Datenobjekte zu. Der abgebildete Indoor-Positionierungsdienst wird in Abschnitt 5.2.1 als Erweiterungsmöglichkeit für die Spezialisierung von DynAmbient erläutert und ist ein weiteres Beispiel für einen serverseitigen Dienst, der von Anwendungen im CCW angesprochen werden könnte.

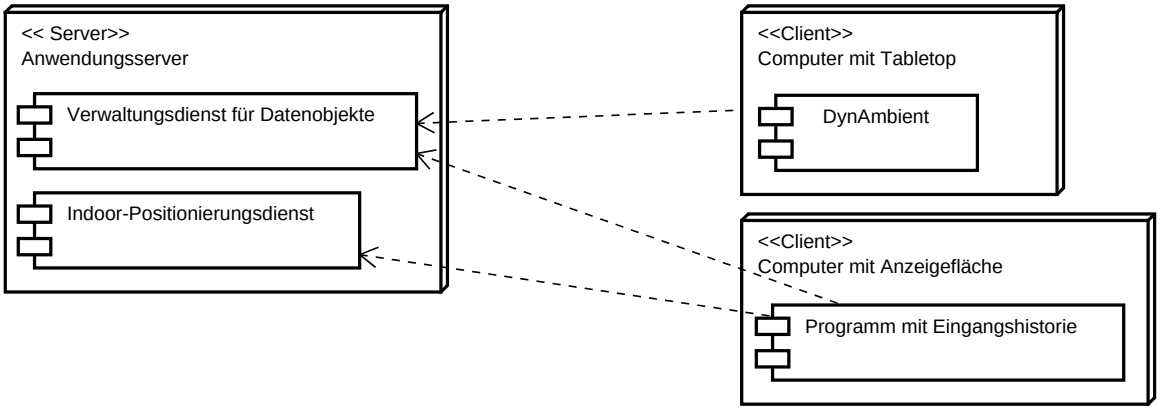


Abb. 4.9: Anwendungs-Verteilungsdiagramm für eine CCW-Systemarchitektur.

Die Verwaltung aller Datenobjekte und deren Eigenschaften durch einen dedizierten Dienst ermöglicht es, die Konsistenz des Datenbestandes im CCW zu gewährleisten. Änderungen, die an Datenobjekten vorgenommen werden, lassen sich mit Hilfe des Dienstes auf einfache Weise propagieren. Dazu registrieren sich Anwendungen für jedes Datenobjekt mit dem sie arbeiten wollen. Sobald Änderungen an einem Datenobjekt durch eine Client-Anwendung erfolgen, wird dies dem Verwaltungsdienst mitgeteilt. Dieser informiert wiederum alle Anwendungen, die sich für das Datenobjekt registriert haben (Push-Strategie). Alternativ ist es auch möglich, dass Anwendungen den Status relevanter Objekte in regelmässigen Abständen vom Verwaltungsdienst abfragen (Pull-Strategie). Zu Umsetzung der Push-Strategie eignet sich z. B. das, in Abschnitt 4.3.1 beschriebene, Observer-Muster.

Die Verwendung eines Verwaltungsdienstes hat ferner den Vorteil, dass die Topologie von Speichergeräten transparent gehalten werden kann. Dies ist möglich, da Anwendungen, die

auf Datenobjekte zugreifen, zunächst den Verwaltungsdienst kontaktieren. Dieser gibt entweder das angeforderte Objekt direkt zurück oder verweist die Anwendungen an den aktuellen Speicherort (z. B. Netzwerkpfad, Datenbankadresse) des Datenobjekts. Aufgrund dieser Entkoppelung ist es möglich, die Funktionalität von Programmen auch bei Änderungen in der physikalischen System-Architektur des CCW zu gewährleisten.

Allgemein hat die Kapselung von Funktionen in Form von Diensten mehrere Vorzüge. Zum einen erlaubt es diese Vorgehensweise Software-Entwicklern definierte Schnittstellen bei der Anwendungsrealisierung zur Verfügung zu stellen. Dies bedeutet, dass Anwendungen unabhängig voneinander entwickelt werden können und die interne Architektur von Dienst-Providern und -Verbrauchern im Hintergrund jederzeit geändert werden kann, solange vereinbarte Schnittstellendefinition unangetastet bleiben.

Ein weiterer Vorzug serviceorientierter Architekturen besteht darin, dass die erneute Implementation von Funktionalitäten, die bereits als Dienste vorliegen, entfällt. Dadurch ist es möglich bei der Entwicklung von Anwendungen Zeit zu sparen. DynAmbient könnte z. B. durch Nutzung des Indoor-Positionierungsdienstes mit relativen geringen Aufwand erweitert werden. Ferner hilft die Nutzung von Diensten Programmierfehler zu vermeiden, die im Zuge einer erneuten Implementation auftreten können.

Abbildung 4.10 zeigt, wie die beschriebene serviceorientierte Anwendungsarchitektur auf eine CCW-Systemarchitektur angewendet werden kann. Die Darstellung stellt eine Kombination der CCW-Systemarchitektur aus Abb. 4.2 und dem Verteilungsdiagramm für mehrere Software-Dienste (vgl. Abb. 4.9) dar.

4.5 Gestalterische und technische Umsetzung

Im folgenden wird die konkrete Umsetzung von DynAmbient anhand der beschriebenen Anforderungen aus Abschnitt 3 beschrieben. Neben der grafischen Gestaltung der Anwendung werden verwendete Software-Komponenten erläutert und erklärt wie die Interaktionstechniken aus Abschnitt 3.2.2 technisch integriert werden konnten.

4.5.1 DynAmbient-Bedienoberfläche

Die Gestaltung der Anwenderoberfläche von DynAmbient erfolgte unter Berücksichtigung des beschriebenen Notfall-Szenarios. Neben der Begutachtung von Fotos und Videos, sollen Anwender in der Lage sein, gesichtete Objekte von der Anwenderoberfläche zu entfernen, abzuspeichern oder an andere Ausgabegeräte wie z. B. Drucker weiterzuleiten. Da DynAmbient auf einem Tabletop ausgeführt wird, ist es wichtig, dass die genannten Aktionen von

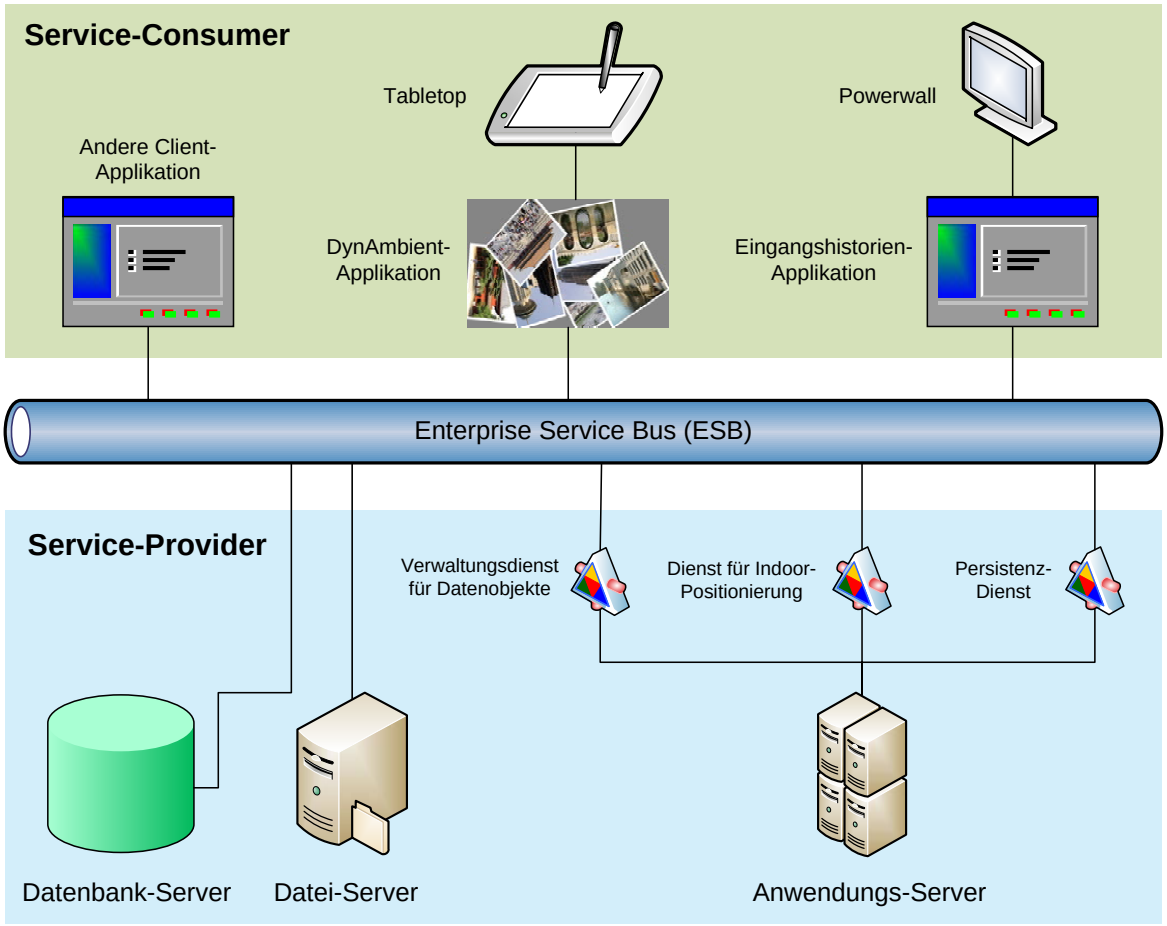


Abb. 4.10: Integration von DynAmbient in eine CCW-Anwendungslandschaft mit serviceorientierter Architektur. Client-Applikationen können bei diesem Entwurf über den ESB Dienste nutzen, die von Server-Anwendungen zur Verfügung gestellt werden. Der Zugriff auf Datenobjekte, die in Datenbank- und Datei-Servern gespeichert sind, ist sowohl für server- als auch clientseitige Applikationen möglich.

allen Positionen um den Tisch ausgeführt werden können. Angesichts des Ausführungskontextes soll die Bedienung der Anwendung eine geringe kognitive Belastung für die Anwender mit sich bringen.

Die Anwenderoberfläche von DynAmbient ist in Abb. 4.11 dargestellt. Anhand der eben beschriebenen Anforderungen wurde eine Arbeitsumgebung entwickelt, in der Objekte auf einer horizontalen Fläche realitätsnah bewegt, rotiert und vergrössert werden können. Ferner lassen sich die Objekte, in vier Öffnungen an den Seiten der Anwenderoberfläche werfen. Durch geeignete Wahl der Öffnung können Datenobjekte im Falle der DynAmbient-Version, die in Abb. 4.11 gezeigt wird, an verschiedene Speicherorte kopiert werden. Ebenso wäre es natürlich möglich die Öffnungen mit Aktionen wie „Löschen“ oder „Drucken“ zu verbinden.

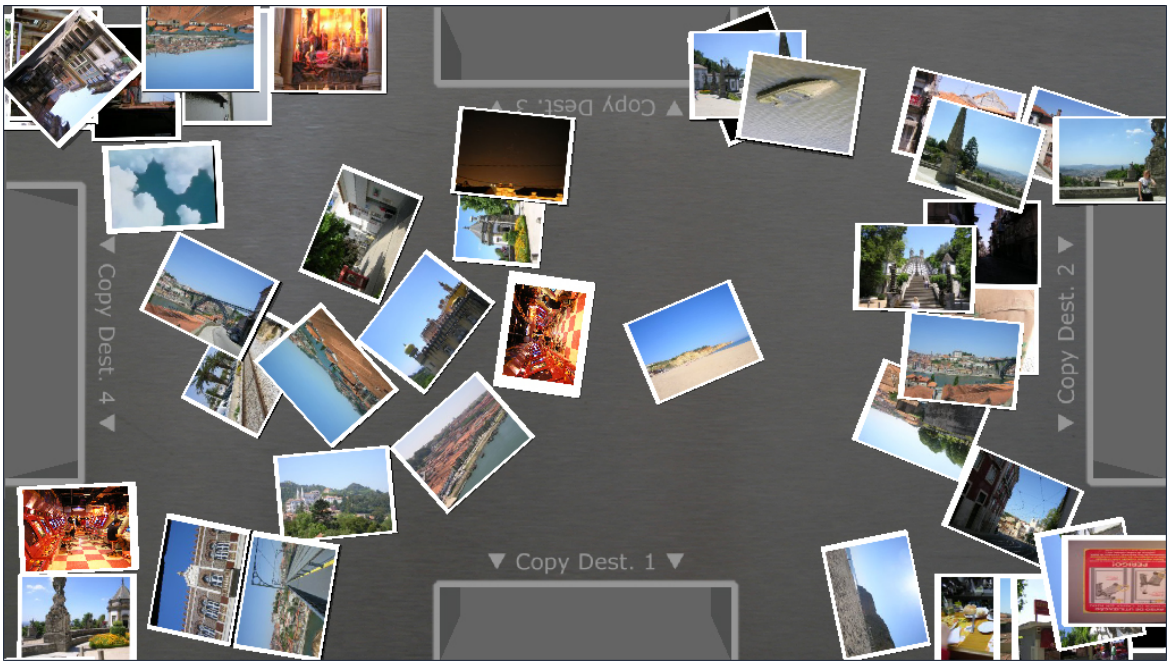


Abb. 4.11: Anwenderoberfläche von DynAmbient.

Die in Abschnitt 3.2.2 beschriebenen Interaktionstechniken „Foto verschieben“, „Foto rotieren“ und „Foto verschieben und gleichzeitig verschieben“ können mit der DynAmbient-Software ausgeführt werden. Lediglich die Interaktionstechnik „Foto skalieren“ wurde nicht wie geplant implementiert, da während der Entwicklung der Anwendung kein Multi-Touch-fähiges Eingabegerät zur Verfügung stand. In der gegenwärtigen Version von DynAmbient wurde ersatzweise ein Funktion integriert, die es erlaubt Objekte heran zu zoomen, wie in Abb. 4.12 illustriert. Bei einer Steuerung der Anwendung mit der Maus, wird die Zoom-Funktion durch einen Doppelklick auf ein Objekt ausgelöst. Bei erneutem Klick wird das Objekt wieder auf seine ursprüngliche Position zurück bewegt.

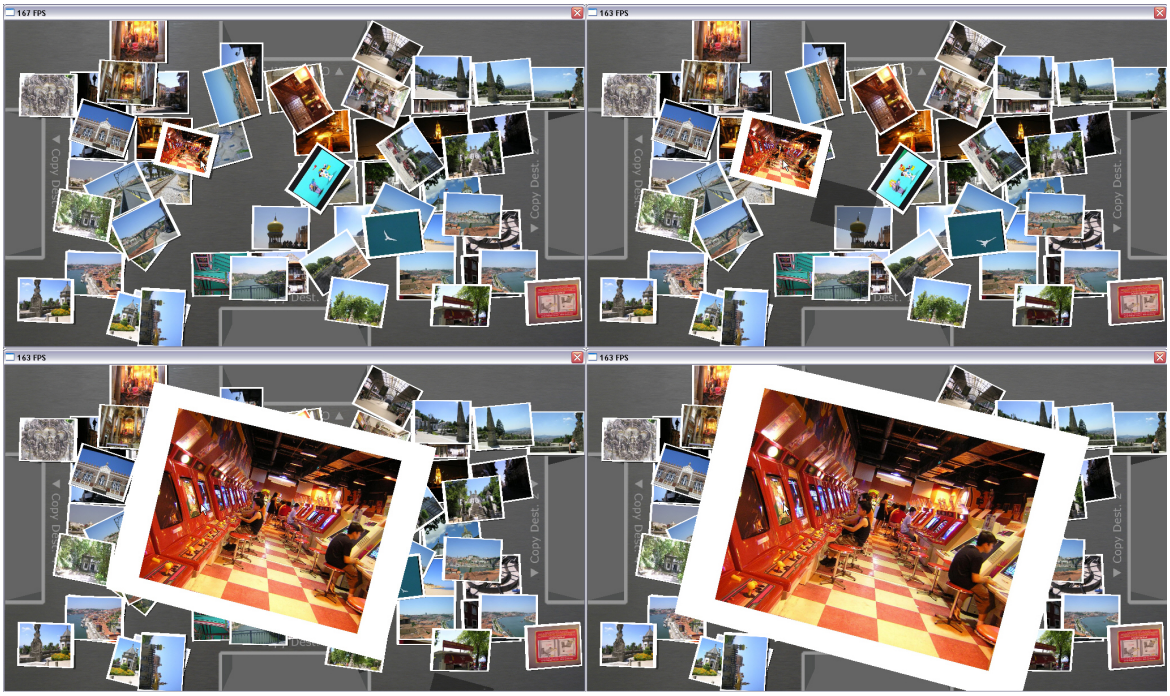


Abb. 4.12: Vergrösserung eines Objekts in DynAmbient. Die Verkleinerung erfolgt grafisch in umgekehrter Reihenfolge.

Weil es durch die physikbasierte Interaktion vorkommen kann, dass Objekte übereinander rutschen, voneinander abprallen oder durch Anstossen mit hoher Geschwindigkeit über die Fläche gleiten, war es notwendig die Arbeitsumgebung 3-dimensional zu gestalten um den Aktionssraum der Objekte einzuschränken. Die Gestalt der Arbeitsumgebung wurde aufgrund diverser Mängel, die bei Anwendungstests festgestellt wurden, mehrmals überarbeitet. Die verschiedenen Entwürfe werden im folgenden Abschnitt vorgestellt.

4.5.2 Iterative Entwicklung der Arbeitsumgebung

Ein wesentliches Merkmal der Methodologie des User-Centered Design (IBM Corporation, 2007) besteht darin, die Ergonomie von Computer-Anwendungen bereits während der Entwicklung in regelmässigen Abständen anhand von Usability-Tests zu überprüfen. Durch die Analyse des Feedbacks von Anwendern ist es möglich Änderungen am Applikations-Design vorzunehmen und so die Qualität der Anwendung kontinuierlich zu verbessern.

Die Entwicklung von DynAmbient erfolgte nach diesem Schema. Zu Beginn dieser Masterarbeit stand lediglich fest, dass eine horizontale Interaktionsfläche mit mehreren Öffnungen die

Bedienoberfläche bilden sollte. Im Laufe der Entstehung des Programms wurde die Ergonomie der Applikation mehrmals überprüft und zweimal überarbeitet. Als Testaufbau diente dabei ein Tabletop-System des Ambient-Labors der HAW-Hamburg, wie in Abb. 3.1 gezeigt. Auf diesem wurde die DynAmbient-Software ausgeführt und von mehreren Studenten getestet. Basierend auf den dabei gesammelten Eindrücken und der durch die Anwender geäußerte Kritik, wurde die Bedienoberfläche angepasst und optimiert.

Ein erster 3-dimensionaler Entwurf der Arbeitsumgebung, zeigt Abb. 4.13. Dabei handelt es sich um eine horizontale Arbeitsfläche, die wie ein Billardtisch von Banden umgeben ist. Diese verhindern, dass Objekte aus dem Sichtfeld des Anwenders geraten, der von oben auf die Fläche blickt. In allen vier Banden befinden sich Öffnungen, in die Objekte von der Tischfläche geschoben werden können. Fällt ein Objekt durch ein Loch wird z. B. das damit verbundene Datenobjekt gelöscht, verschoben oder an ein Ausgabegerät geschickt. Die Öffnungen sind so positioniert, dass sie von Anwendern an allen Seiten des Tisches und Links- als auch Rechtshändern gut erreicht werden können.

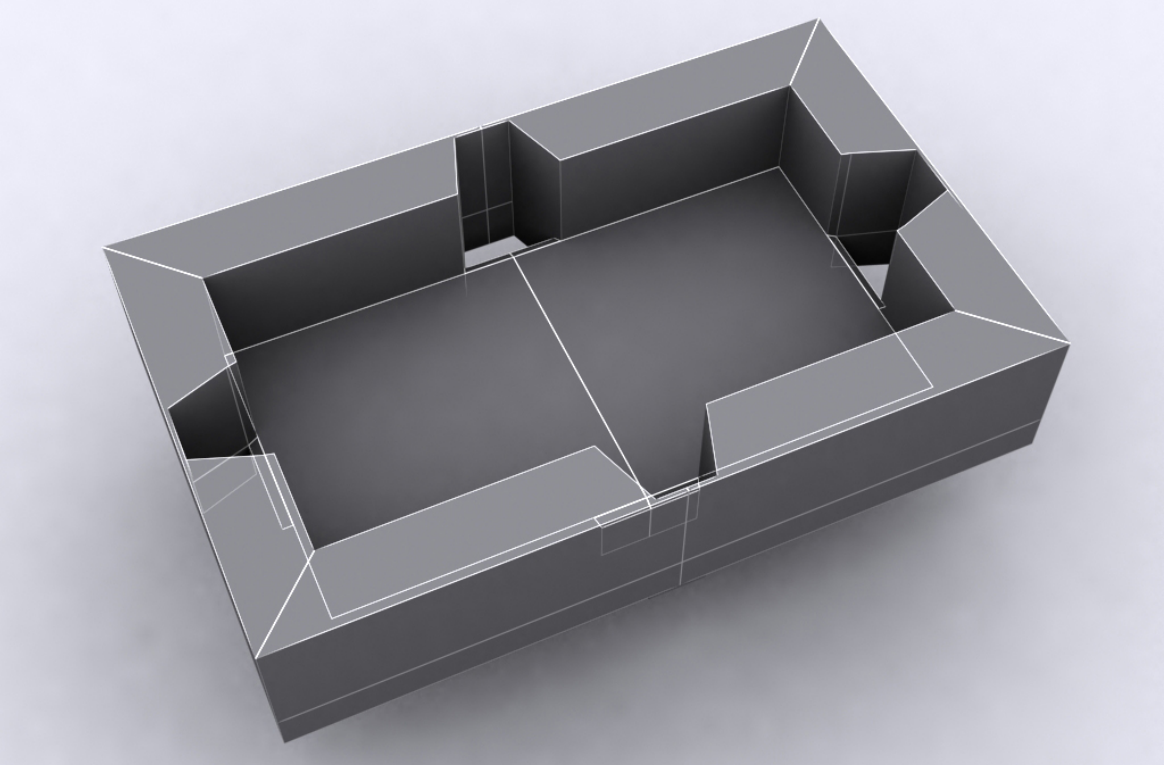


Abb. 4.13: Erster Entwurf der DynAmbient-Arbeitsumgebung.

Bei Tests der Anwendung mit diesem Modell, stellten sich mehrere Nachteile heraus. Zum einen befinden sich die Öffnungen der Arbeitsumgebung in den Banden. Um sie dem Anwender sichtbar zu machen muss daher ein gewisser Anteil der Banden gezeigt werden,

wie in Abb. 4.14 gezeigt. Dies verringert wiederum die Größe der eigentlichen Arbeitsfläche. Ferner war die Größe der Öffnungen zu klein gewählt um die Objekte auf der Oberfläche in ausreichender Größe darstellen zu können.

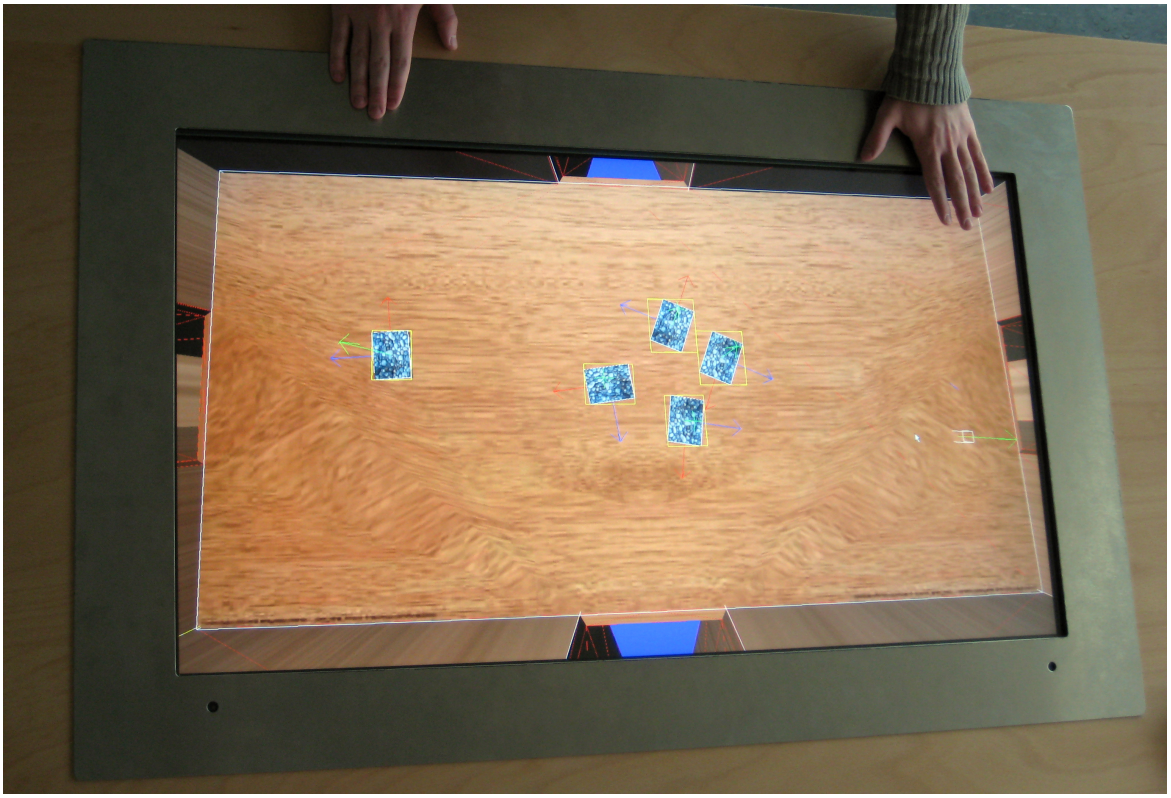


Abb. 4.14: Benutzeroberfläche von DynAmbient auf einem Tabletop mit dem ersten Entwurf der Arbeitsumgebung.

Aufgrund dieser Beobachtungen, wurde die Arbeitsumgebung angepasst. Das Ergebnis veranschaulicht Abb. 4.15. Die Öffnungen an den Seiten wurden bei diesem Modell in die Tischfläche integriert und vergrössert. Bei erneuten Tests konnte jedoch auch dieser Entwurf nicht vollständig überzeugen. Ein Grund dafür bestand darin, dass die horizontale Fläche aus mehreren Quadern zusammengesetzt ist. Dadurch kam es bei der Bewegung von Objekten auf der Fläche vor, dass diese an Berührungsstellen der Quader hängen blieben obwohl dort keine Höhenunterschiede vorlagen. Dieses Problem ist vermutlich auf Rundungsfehler bei der Werte-Berechnung durch die Physik-Engine zurück zu führen.

Eine weiterer Nachteil der zweiten Arbeitsumgebung besteht darin, dass sich hinter den Öffnungen keine Banden befinden. Durchquert ein Objekt eine Öffnung mit hoher Geschwindigkeit, entfernt es sich zunächst ein Stück weit von der Arbeitsfläche und beginnt erst dann zu fallen. Dieses Verhalten wirkt irritierend, da der Anwender kein ausreichendes visuelles Feedback erhält, was mit dem Objekt passiert.

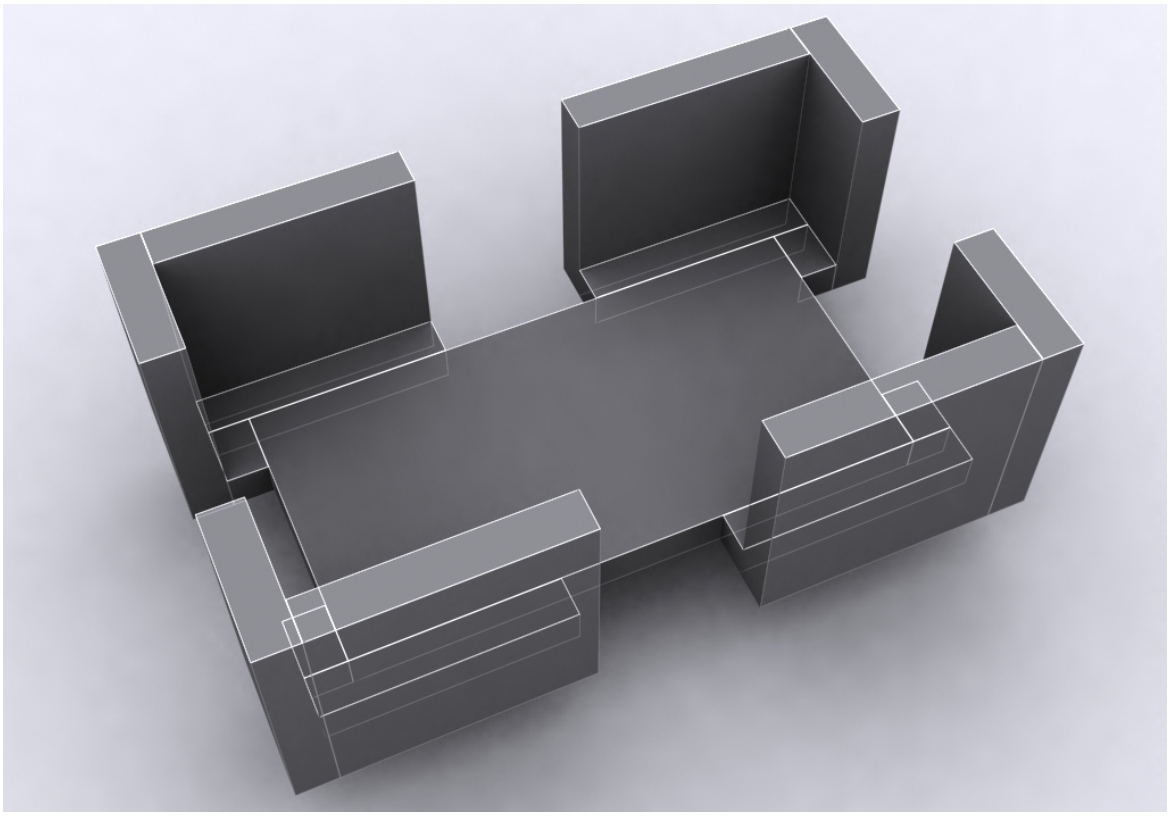


Abb. 4.15: Zweiter Entwurf der DynAmbient-Arbeitsumgebung.

Die beschriebenen Nachteile des zweiten Entwurfs, wurden beim dritten Entwurf der Arbeitsumgebung (siehe Abb. 4.16) behoben. Die horizontale Arbeitsfläche besteht bei dieser Ausführung aus einem Stück und hinter den Öffnungen befinden sich Banden, an denen Objekte abprallen können und dann für den Anwender sichtbar fallen. Dadurch ist es für Anwender leichter erkennbar, dass ein Objekt von der Arbeitsfläche entfernt wurde. Um dies noch deutlicher zu unterstreichen wird an der Stelle an der das Objekt verschwindet ein grafischer Effekt eingeblendet, den Abb. 4.17 veranschaulicht.

4.5.3 Verwendete Software-Komponenten

Wie in Roßberger (2007) beschrieben, wurden vor Beginn dieser Masterarbeit zwei Physik-Engines getestet, die aufgrund ihrer Verfügbarkeit als Open-Source-Software gewählt wurden. Dabei handelte es sich um die Open Dynamics Engine (ODE)[6] und Newton[7]. Beide

[6] http://www.ode.org/
[7] http://www.newtondynamics.com/

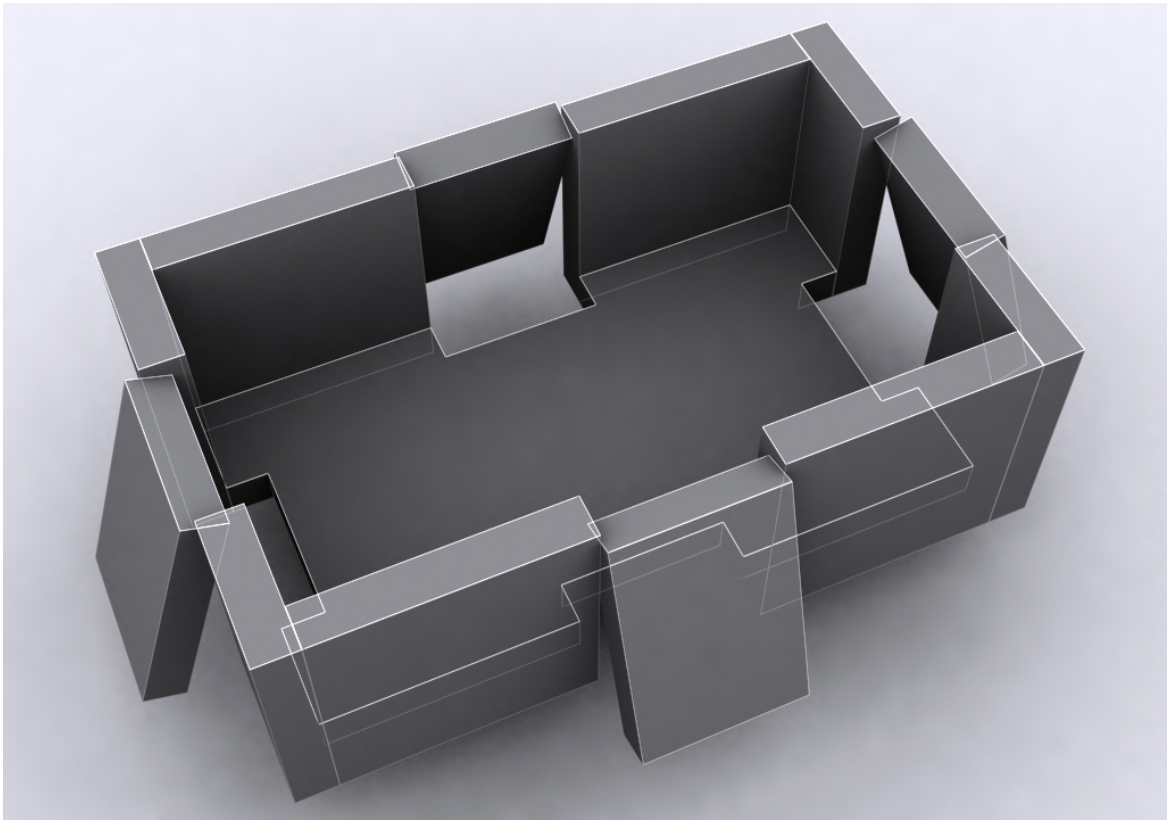


Abb. 4.16: Dritter Entwurf der DynAmbient-Arbeitsumgebung.

Abb. 4.17: Grafischer Effekt, der ausgelöst wird sobald ein Objekt durch eine der Öffnungen in der virtuellen Tischfläche fällt.

Engines erwiesen sich nach eingehenden Tests als nicht stabil genug, unzureichend dokumentiert und verfügten teilweise nicht über benötigte Funktionalitäten.

Aus diesen Gründen fiel die Wahl auf die Physik-Engine Ageia PhysX[8]. Diese Engine wird bei einer Vielzahl professionell entwickelter Spiele eingesetzt, ist sehr stabil, umfangreichend dokumentiert und bietet sehr viele Funktionen. Ferner können über ein spezielles Forum direkt Fragen an die Entwickler der Engine gestellt werden. Die Engine und das zugehörige Software Development Kit (SDK) darf kostenlos und lizenzfrei verwendet werden.

Der Großteil verfügbarer Physik-Engines ist für den Einsatz in Computerspielen konzipiert. Da diese aufgrund bestmöglicher Ausnutzung der Computer-Rechenkapazität überwiegend mit der Programmiersprache C++ entwickelt werden, basieren die meisten Physik-Engines, wie auch Ageia PhysX, ODE und Newton, ebenfalls auf C++. Um die volle Funktionalität von PhysX nutzen und auf einfache Weise integrieren zu können, fiel daher die Entscheidung DynAmbient in C++ zu entwickeln.

Um die Physik-Engine leicht mit einer Grafik-Engine koppeln zu können, sollte diese ebenso in C++ vorliegen. Wegen positiven Erfahrungen bei früheren Projekten, wurde an dieser Stelle die 3D-Grafikengine Irrlicht[9] ausgewählt, die als Open-Source-Software zur Verfügung steht und aufgrund hoher Anwenderfreundlichkeit ein schnelle Entwicklung von Applikationen erlaubt.

Die Wiedergabe von Video-Dateien in DynAmbient erfolgt mit Hilfe der frei verfügbaren Software-Bibliothek libavcodec[10]. Diese wird verwendet, weil sie das Abspielen einer Vielzahl unterschiedlicher Video- und Tonformate erlaubt und dadurch viele verschiedene Dateitypen in die DynAmbient-Software geladen und dort betrachtet werden können.

4.5.4 Physikbasierte Interaktion

Ein primäres Ziel dieser Masterarbeit war die Entwicklung einer Software mit physikbasierter Interaktion. Zu diesem Zweck erfolgt die Beeinflussung der Position und Ausrichtung der Objekte auf der horizontalen Fläche mit Hilfe so genannter Joints (siehe Abb. 4.18). Joints sind Bestandteile des Funktionsumfangs der Physik-Engine. Ein Joint stellt ein Feder-Masse-System dar, das zwei Körper miteinander verbindet und ihn seinem Verhalten einer Feder ähnelt: sobald einer der Körper sich bewegt und die Feder zur maximalen Dehnung bringt, wirkt eine Zugkraft auf den anderen Körper.

Die Bewegung eines Objekts auf der Oberfläche erfolgt nach diesem Prinzip. Dazu existiert in der DynAmbient-Software ein Körper, der mit Hilfe der Physik-Engine erzeugt wird, und

[8] http://www.ageia.com/physx/index.html
[9] http://irrlicht.sourceforge.net/
[10] http://ffmpeg.mplayerhq.hu/index.html

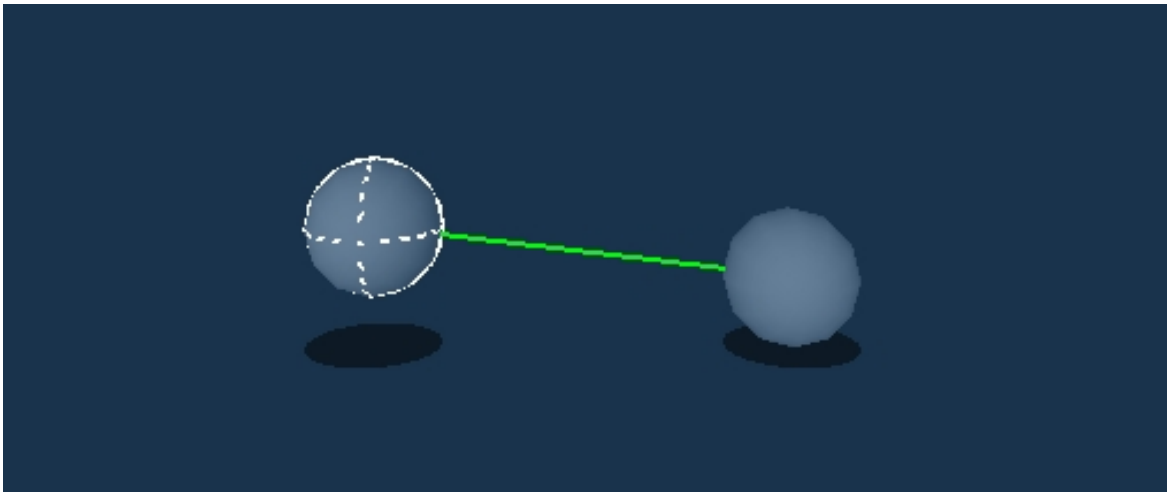
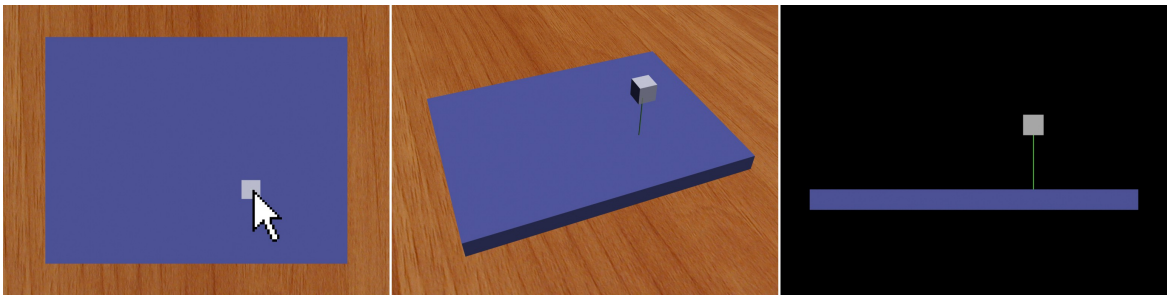


Abb. 4.18: Beispiel für eine Joint-Verbindung (grüne Linie) zwischen zwei Körpern.

für den Anwender nicht sichtbar der Position des Mauszeigers folgt (siehe Abb. 4.19). Durch Druck auf den Touchscreen oder Betätigung der linken Maustaste, wird zwischen dem unsichtbaren Körper und dem darunter liegenden Objekt auf der horizontalen Fläche ein Joint erzeugt. Sobald der Anwender seinen Finger bzw. die Maus bewegt, „folgt“ das Objekt der Position des Mauszeigers und kann so auf der horizontalen Fläche bewegt werden. Der Joint wird zerstört wenn der der Finger vom Touchscreen genommen oder die Maustaste losgelassen wird. Damit ist die Verbindung zwischen den beiden Körpern wieder gelöst.



Anwendersicht Perspektivisch Seitenansicht

Abb. 4.19: Verschiedene Ansichten des Joints, der zwischen dem Mauszeiger-Objekt und dem darunter liegenden Objekt erzeugt wird. Das Maus-Objekt (helles Quadrat unter dem Mauszeiger im linken Bild) und der Joint sind für den Anwender nicht sichtbar.

Die Joint-Paramter in DynAmbient sind so konfiguriert, dass das Verhalten des Joints dem eines Gummibandes ähnelt. Dadurch folgen verbundene Objekte dem Mauszeiger-Objekt mit einer leichten Verzögerung. Diese Konfiguration wurde gewählt, weil sich die in Abschnitt

3.2.2 beschriebenen Interaktionstechniken bei Tests damit am besten ausgeführt werden konnten und am intuitivsten erschienen.

4.5.5 Modellierung der Anwenderoberfläche

Eine Möglichkeit physikalische Objekte in der PhysX-Engine zu erstellen, besteht in der Verwendung bereitgestellter Prozeduren, die es erlauben primitive Körper wie Würfel, Kugeln oder Kapseln zu erzeugen und parametrisieren. Diese Vorgehensweise ist bei der Gestaltung komplexer Szenerien nicht praktikabel, da code-basierte Änderungen erst nach Neukompilierung des Programms visuell überprüft werden können.

Eine komfortablere und deutlich effizientere Vorgehensweise besteht in der Verwendung eines Plugins[11], das es erlaubt 3-dimensionale Objekte und Szenerien mit Hilfe eines 3D-Computergrafikprogramm wie Autodesk 3D Studio Max[12] zu erstellen und anschliessend in Form von XML-Dateien zu exportieren. Diese lassen sich wiederum in die PhysX-Engine importieren und dort direkt verwenden.

Das eben beschriebene zweite Verfahren wurde bei der Entwicklung von DynAmbient genutzt. Die horizontale Fläche, auf der sich die geladenen Datenobjekte befinden, ist Bestandteil eines komplexeren 3-dimensionalen Modells, dessen iterative Entwicklung ausführlich in Abschnitt 4.5.2 beschrieben wird. Abbildung 4.20 zeigt die Entwicklung des Modells in einem 3D-Computergrafikprogramm und dessen Test im Visual Remote Debugger der PhysX-Engine. Der Visual Remote Debugger erlaubt es mit Hilfe einer grafischen Oberfläche den Zustand aller geladenen physikalischen Objekte zu überprüfen. Im Laufe der Entwicklung von DynAmbient wurde diese Anwendung genutzt, um sicher zu stellen, dass die Positions- und Rotationswerte der physikalischen Entitäten korrekt auf die zugehörigen grafischen Entitäten (vgl. Abschnitt 4.2) übertragen werden.

Die grafische Erscheinung der DynAmbient-Arbeitsumgebung wird ebenfalls durch ein 3D-Modell bestimmt. Die Irrlicht-Engine, die DynAmbient zur Visualisierung nutzt ist unter anderem in der Lage Modelle im 3DS-Dateiformat zu importieren. Modelle in diesem Format lassen sich wie die physikalischen Modelle im XML-Format mit Hilfe des 3D-Computergrafikprogramms 3D Studio Max erstellen und exportieren. Dadurch ist es möglich sowohl die grafische als auch die physikalische Entität der DynAmbient-Arbeitsumgebung in Form eines einzigen 3D-Modells in 3D Studio Max zu erstellen und von dort in die beiden benötigten Datenformate für PhysX und Irrlicht zu exportieren.

Da DynAmbient beim Start die physikalische Entität der Arbeitsumgebung aus einer XML-Datei und die grafische Entität aus einer 3DS-Datei lädt, kann die Anwenderoberfläche der

[11] http://sourceforge.net/projects/physxplugin
[12] http://www.autodesk.de/3dsmax

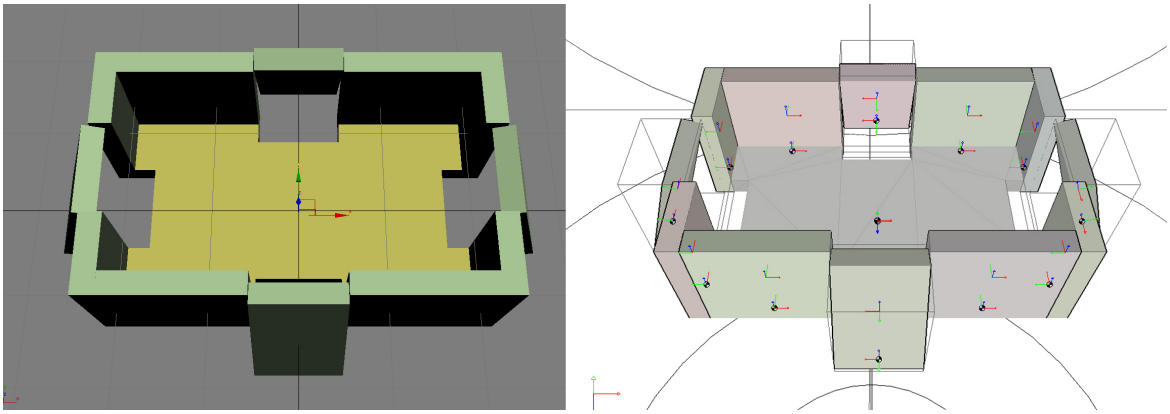


Abb. 4.20: Ansicht des DynAmbient-Szenenmodells in einem 3D-Computergrafikprogramm (links) und im Visual Remote Debugger von Ageia PhysX.

Anwendung ohne Veränderung des Quellcodes angepasst werden. Lediglich das 3D-Modell, auf dem die physikalische und die grafische Repräsentation der Arbeitsumgebung basiert, muss hierfür geändert und neu exportiert werden. Dadurch ist es möglich DynAmbient schnell und bequem für unterschiedliche Szenarien und Einsatzfälle zu konfigurieren.

Eine weitere Möglichkeit die Erscheinung von DynAmbient anzupassen, besteht in der Verwendung alternativer Texturen für die Arbeitsumgebung. Die Textur ist eine Bitmap-Grafik, die ebenfalls beim Start der Applikation geladen wird und durch Ersetzen mit einer anderen Grafik mit gleichen Abmessungen ausgetauscht werden kann. Einige Beispiele für unterschiedliche Texturen zeigt Abb. 4.21.

Bei den dargestellten Textur-Varianten wird die Funktion der Öffnungen unter anderem mit Hilfe von Symbolen definiert. Beispielweise weisst die Textur links unten den Öffnungen die Funktionen „Drucken“, „Ausschneiden“, „Kopieren“ und „Löschen“ zu. In der Textur rechts daneben können Fotos und Videos unterschiedlichen Gefahrenarten zugeordnet werden. Die Abänderung der Funktion der Öffnungen in DynAmbient erfordert gegenwärtig noch entsprechende Anpassungen im Quellcode der Anwendung. In einer zukünftigen Version der Software könnte die Funktionalitätsdefinition mit Hilfe einer Konfigurationsdatei geschehen. Dadurch wären Änderungen im Quellcode bei einer Umgestaltung von DynAmbient unnötig, was die Rekonfiguration der Anwendung weiter vereinfachen würde.

4.5.6 Steuerung der Anwendung durch alternative Eingabegeräte

Bei ihrer Benutzung erzeugen Eingabegeräte wie Maus und Tastatur im Betriebssytem individuelle Nachrichten, die als Eingabe-Events bezeichnet werden. Wenn der Anwender in

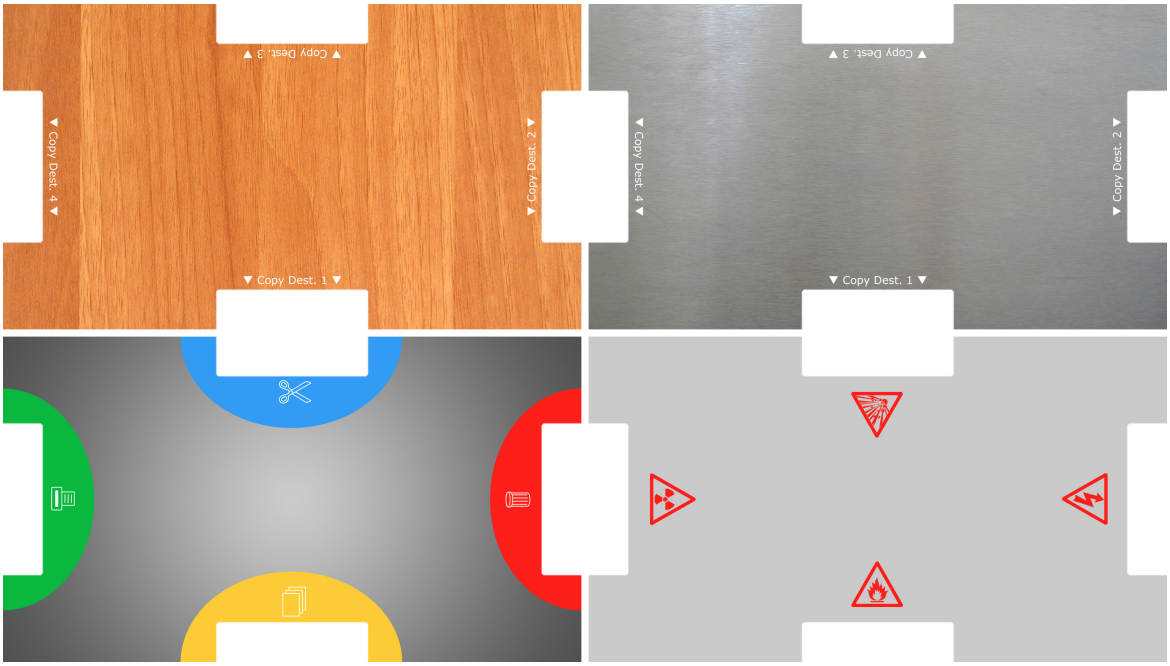


Abb. 4.21: Textur-Alternativen für die DynAmbient-Arbeitsumgebung. Neben dem Austausch des „Untergrundmaterials“ (obere Reihe), kann die Anwendung durch Einsatz anderer Beschriftungen beispielsweise in einem Notfall-Szenario (rechts unten) verwendet werden.

einem Programmfenster z. B. die linke Maustaste drückt, sendet das Betriebssytem ein spezifisches Eingabe-Event an das Programm von dem das Fenster bereit gestellt wird. Soll das Programm auf dieses Eingabe-Event reagieren, so muss es dafür eine spezielle Funktion enthalten in der definiert ist, was beim Empfang des Events geschehen soll.

Ein Großteil aktuell verbreiteter Computer-Anwendungen ist für die Bedienung per Maus entwickelt. Daher enthalten diese Programme Funktionen die Events dieses Gerätetyps verarbeitet können. Damit auch alternative Eingabegeräte wie Touchscreens oder Trackballs zur Bedienung solcher Anwendungen nutzbar sind, generieren diese dieselben Events, die auch von einer Maus erzeugt werden.

Auch die DynAmbient-Applikation enthält Funktionen zur Verarbeitung von Maus-Events. Diese sind in der Controller-Komponente der Anwendung definiert, welche in Abschnitt 4.3 beschrieben wird.

Die Bedienung von DynAmbient erfolgt in erster Linie durch den Mauszeiger und die linke Maustaste. Objekte auf der horizontalen Oberfläche können nach der *Drag-and-Drop*-Methode, die unter den meisten grafischen Betriebssystemen wie Microsoft Windows oder Apple OS X nutzbar ist, verschoben und gedreht werden. Dazu zeigt der Anwender zunächst

mit dem Mauszeiger auf das Objekt mit dem er interagieren will. Durch Druck auf die linke Maustaste kann nun das Objekt „ergriffen“ werden und ist mit Hilfe eines Joints (vgl. 4.5.4) solange an den Mauszeiger „gebunden“ bis die Taste losgelassen wird. Das Vergrössern von Objekten erfolgt wie bereits erwähnt mit Hilfe eines doppelten Klick auf die linke Maustaste. Durch anschließenden einfachen Klick auf die linke Maustaste wird das Objekt wieder verkleinert und dabei an seine Ausgangsposition zurück bewegt.

In der Masterarbeit von Fischer (2007) wird ein multimodales Interaktionssystem entworfen, das sich besonders für die Nutzung in kollaborativen Computerumgebungen eignet. Ein Bestandteil der Arbeit ist die Entwicklung einer Treiber-Software, die es ermöglicht mit Hilfe einer Nintendo Wiimote[13]-Fernbedienung den Mauszeiger unter Microsoft Windows zu steuern und Maustasten-Events zu generieren. Das Wiimote-Eingabegerät ist für die Bedienung von Software-Anwendungen auf vertikalen Anzeigeflächen konzipiert wie in Abb. 4.22 gezeigt.

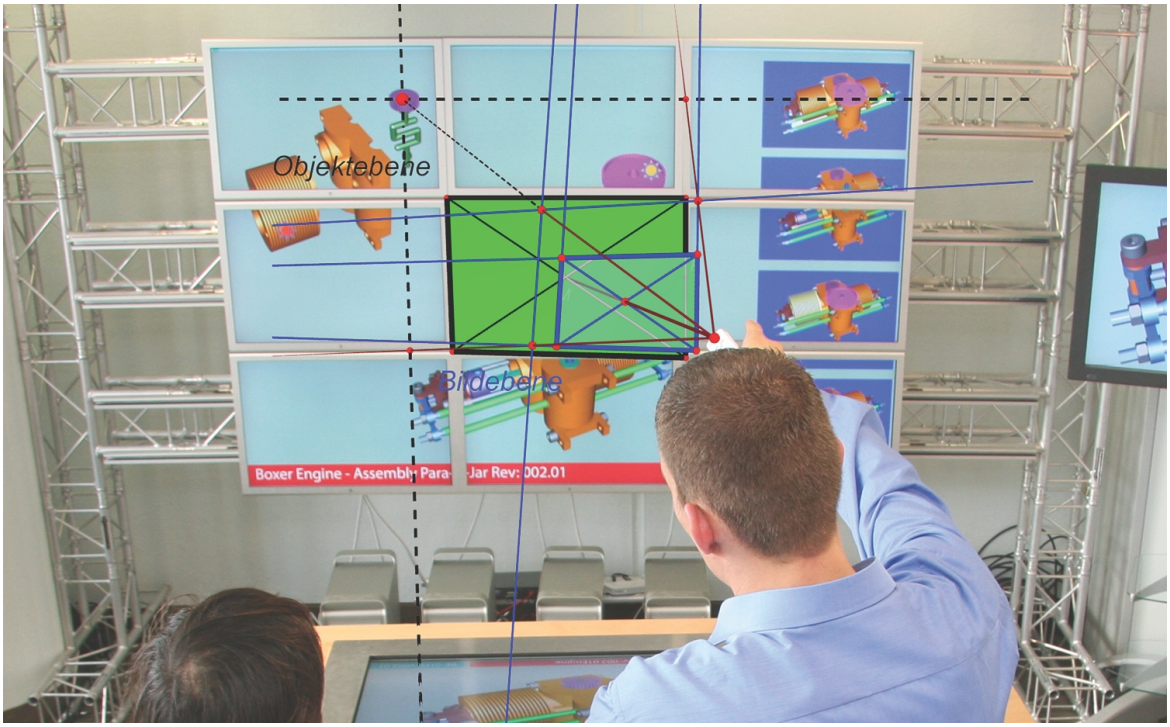


Abb. 4.22: Einsatz der Wiimote-Fernbedienung zur Steuerung einer Anwendung auf einer Powerwall. Die eingezeichneten Linien illustrieren die geometrische Berechnung der Mauszeigerposition anhand der Wiimote-Ausrichtung (von Fischer, 2007).

Da DynAmbient mit jedem Eingabegerät gesteuert werden kann, das die benötigten Maus-Events generiert, ist es ohne Anpassungen der Software möglich, diese in Kombination mit

[13] http://wii.com/

dem Wiimote-Treiber und einer Wiimote-Fernbedienung zu betreiben. Zur Demonstration wurde die DynAmbient-Anwendung mit Hilfe eines Beamers an eine Wand projiziert. Diese war mit vier quadratisch angeordneten LEDs beklebt, welche zur Positionsberechnung des Mauszeigers vom Wiimote-Treiber benötigt werden. Dieser Testaufbau ermöglichte es, DynAmbient vollständig mit Hilfe der Wiimote-Fernbedienung zu steuern, wie in Abb. 4.23 gezeigt.

Abb. 4.23: Steuerung der DynAmbient-Software mit einer WiiMote-Fernbedienung.

Die Benutzung von DynAmbient mit Hilfe der Wiimote-Fernbedienung wurde von mehreren Anwendern als einfach und intuitiv beschrieben. Dabei wurden es nicht als störend empfunden, dass die Textur und physikalische Modellierung der Anwendung für die Nutzung auf einer horizontalen Anzeigefläche angepasst war. Dennoch ist es gut vorstellbar, dass mit einer angepassten Ausführung von DynAmbient für vertikale Anzeigefläche die Benutzerfreundlichkeit noch weiter gesteigert werden kann.

4.6 Bewertung der Umsetzung und Fazit

DynAmbient erfüllt mit Ausnahme der Interaktionstechnik „Objekt vergrössern“ (vgl. 3.2.2) alle in Abschnitt 3.2 formulierten funktionalen Anforderungen. Da bei der Anfertigung dieser Ausarbeitung kein Eingabegerät mit Multi-Touch zur Verfügung stand, wurde die Interaktionstechnik „Objekt vergrössern“ nicht integriert. Aus demselben Grund wurde darauf verzichtet in DynAmbient Funktionen zur gleichzeitigen Verarbeitung mehrerer Eingaben zu integrieren, wie dies bei den nicht-funktionalen Anforderungen in Abschnitt 3.3 gefordert wird. Die Integration der fehlenden Interaktionstechnik und die Verarbeitung mehrerer gleichzeitiger Eingaben ist daher Ziel zukünftiger Arbeiten. Aufgrund der beschriebenen Modularisierung der Anwendung mit Hilfe des MVC-Entwurfsmusters (siehe Abschnitt 4.3.2) kann die Controller-Komponente ohne Änderung anderer Bereiche der Software auf relativ einfache Weise mehrbenutzerfähig gemacht und die fehlende Interaktionstechnik ergänzt werden.

Da die Interaktion in DynAmbient ausschliesslich durch Bewegung der Objekte auf der horizontalen Fläche erfolgt und diese ausreichend groß modelliert sind, konnte die nicht-funktionale Anforderung „Touch-gerechte Bedienung“ erfüllt werden.

Die ausreichende Performanz und Skalierbarkeit von DynAmbient, wie in Abschnitt 3.3 verlangt, wird durch den Einsatz der PhysX-Engine sicher gestellt. Diese ermöglicht bisher als einzige Physik-Engine die Ausführung aller Berechnungen auf einer speziellen Physik-Beschleunigerkarte[14], die den Prozessor erheblich entlastet und somit die Ausführung sehr komplexer Szenen ermöglicht.

Der Einsatz der PhysX-Engine erwies sich im Laufe dieser Arbeit aus weiteren Gründen als gute Entscheidung. Die Engine erwies sich als sehr umfangreich, stabil und erlaubte durch den Einsatz von Joints (vgl. Abschnitt 4.5.4 die problemlose Nachbildung der gewünschten Interaktionstechniken.

Weiterhin kann DynAmbient, wie in Abschnitt 4.5.5 erläutert, durch den Einsatz der PhysX-Engine physikalische Szenerien und 3D-Objekte importieren, die mit Hilfe eines 3D-Grafikprogramms erstellt wurden. Die Nachbildungen verhalten sich dabei nahezu wie ihre reellen Vorbilder: wird an einer Stelle der horizontalen Oberfläche eine Senke modelliert, so werden sich bei der Ausführung von DynAmbient, darin Objekte sammeln, wie dies auch in der Realität geschehen würde. Die Funktionalität der Anwendung entspricht damit dem mentalen Modell des Entwicklers und des Anwenders, die beide aufgrund ihrer Erfahrungen aus der Realität das beschriebene Verhalten erwarten. Der Einsatz eines 3D-Grafikprogramms erleichtert damit die Gestaltung intuitiv bedienbarer Anwendungen anhand realitätsbasierter mentaler Modelle (siehe Abschnitt 2.4) enorm.

[14] http://www.ageia.com/physx/index.html

Ein sehr interessanter Aspekt, der in dieser Arbeit nicht untersucht wurde, ist die Evaluation von DynAmbient hinsichtlich Benutzerfreundlichkeit und der Bedienung der Anwendung durch mehrere Personen. Dabei wäre von zentralem Interesse, wie die Anwenderoberfläche verbessert und effizienter gestaltet werden könnte. Vorstellbar ist in diesem Zusammenhang z. B. die Modellierung von Vertiefungen in denen Objekte gesammelt werden könnten.

Zusammenfassend kann DynAmbient trotz einiger nicht integrierter Funktionalitäten aufgrund physikbasierter Interaktionstechniken als intuitiv bedienbare Software-Anwendung betrachtet werden. Da die Erscheinung und das Verhalten der Anwendung durch 3D-dimensionale Modelle bestimmbar ist (siehe Abschnitt 4.5.5), kann DynAmbient somit einfach an verschiedene Szenarien und Aufgaben angepasst werden. Mögliche Einsatzfelder und Erweiterungsmöglichkeiten für DynAmbient werden im folgenden abschließenden Kapitel 5 betrachtet.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Computer-Programm namens DynAmbient entwickelt, das für die gemeinschaftliche Zusammenarbeit mehrerer Personen in kollaborativen Computerumgebungen konzipiert ist. Die Anwendungsoberfläche des Programms wurde speziell für den Einsatz auf Tabletops gestaltet und zeichnet sich durch Bereitstellung physikbasierter Interaktionstechniken aus, die durch realitätsnahe Simulation der Anwendungsumgebung mit Hilfe einer Physik-Engine integriert wurden.

Das Grundlagenkapitel 2 beschreibt zum einen den Anwendungskontext von DynAmbient. Dabei handelt es um Co-located Collaborative Workspaces (CCWs), die mit Hilfe spezieller Hard- und Software für die Bearbeitung digitaler Dokumente durch mehrere Personen entworfen sind. Zum anderen werden in diesem Kapitel Forschungsarbeiten vorgestellt, die sich mit Bedienkonzepten für CCW-Programme und Ausrichtungstechniken für digitale Objekte auf horizontalen Anzeigeflächen beschäftigen. Einen besonders interessanten Ansatz bilden dabei physikbasierte Interaktionstechniken. Diese erlauben die Erstellung von Computer-Anwendungen anhand realitätsbedingter mentaler Modelle, wie am Ende dieses Kapitels erläutert wird.

Im anschließenden Kapitel 3 werden funktionale und nicht-funktionale Anforderungen beschrieben, denen DynAmbient, als physikbasiertes Computer-Programm für kollaborative Umgebungen, genügen soll. Die Ermittlung der Anforderungen erfolgt dabei anhand eines Beispielszenarios, das Bestandteil des Notfall-Forschungsprojekts der HAW Hamburg ist und sich mit der Entwicklung einer Leitstand-Applikation beschäftigt, die es ermöglicht Datenobjekte wie Fotos und Videos zu sichten und weiterzuleiten.

Anhand der definierten Anforderungen wurde schließlich eine Software-Architektur für DynAmbient entwickelt und in Form eines Programms realisiert. Kapitel 4 dokumentiert dabei getroffene Design-Entscheidungen und erläutert wie DynAmbient durch den Einsatz dienstbasierter Anwendungskommunikation in die Anwendungslandschaft eines CCW integriert werden kann. Ferner wird die gestalterische und technische Umsetzung von DynAmbient vorgestellt und dabei verwendete Technologien beschrieben. Abschliessend diskutiert dieses Kapitel, inwiefern die gestellten Anforderungen umgesetzt werden konnten. Dabei werden

außerdem Vorteile beschrieben, die sich aus dem Einsatz bestimmter Software-Bibliotheken bei der Anwendungsentwicklung ergaben und Möglichkeiten zur Weiterverwendung und Untersuchung von DynAmbient vorgeschlagen. Im folgenden Ausblick wird dieser Aspekt nochmals aufgegriffen. Es werden zudem verschiedene Möglichkeiten der Weiterentwicklung von DynAmbient genannt.

5.2 Ausblick

5.2.1 Spezialisierung der Anwendung

Das vorangegangene Kapitel 4 beschrieb unter anderem das Aussehen der Anwenderoberfläche von DynAmbient. Ferner wurde in diesem Teil der Arbeit erläutert, dass sich dich Anwendung relativ leicht für unterschiedliche Einsatzszenarien umgestalten lässt (vgl. 4.5.5).

Im Hinblick auf das bekannte Notfallszenario bietet es sich an, aufgenommene Fotos und Videos anhand von Geoinformationsdaten (Längen- und Breitengrad) auf einer Karte anzuordnen. Die Positionsbestimmung kann dabei außerhalb von Gebäuden via GPS und im Innenbereich durch Indoor-Positionierungssysteme (Kutak, 2006; Napitupulu, 2006) erfolgen. Eine durch Geoinformationsdaten erweiterte Version von DynAmbient könnte dabei wie in Abb. 5.1 aussehen.

Die entsprechende Umgestaltung von DynAmbient und die Integration von Positionsdaten, die von einem speziellen Dienst im CCW (siehe Abschnitt 4.4) bezogen werden könnten, ist eine interessante Möglichkeit zur Weiterentwicklung der Anwendung.

5.2.2 Unterstützung anderer Eingabegeräte

Wie bereits in Abschnitt 4.2.2 erläutert, ist es mit der gegenwärtigen Version von DynAmbient nicht möglich Datenobjekte zu annotieren. Die dazu nötige Eingabe textueller Beschreibungen könnte durch den Einsatz von Multi-Touchscreens erfolgen, deren Unterstützung ursprünglich ein Ziel dieser Masterarbeit war. Aufgrund fehlender Hardware (siehe Abschnitt 4.5.1), wurde dieses Ziel nicht weiter verfolgt.

Im Rahmen des UbiComp-Projekts der HAW Hamburg, beschäftigen sich gegenwärtig eine Reihe von Forschungsarbeiten, wie von Gehn (2007), mit der Nutzung von Multi-Touchoberflächen zur Steuerung von Computer-Anwendungen für CCWs. Eine verbesserte Unterstützung von Multi-Touchscreens und anderer Eingabegeräte wie der Wii-Fernbedienung (vgl. 4.5.6) stellt eine weitere Entwicklungsrichtung für DynAmbient dar.

Abb. 5.1: Möglicher Aufbau eines Leitstands für Rettungseinsätze. Auf dem Tabletop im Vordergrund sind Fotos anhand ihres Aufnahmeortes mit dem darunter liegenden Gebäudeplan und der großen Karte auf der Powerwall im Hintergrund verknüpft.

5.2.3 Verteilte physikbasierte Arbeitsumgebung

Unter Zuhilfenahme spezieller Software-Bibliotheken (Köckritz, 2007), könnte die 3-dimensionale Arbeitsumgebung von DynAmbient, wie in Abb. 5.2 gezeigt, auf mehrere Anzeigegeräte verteilt werden.

Auf diese Weise könnte ein virtueller kollaborativer Arbeitsraum geschaffen werden der eine physikbasierte Interaktion mit DynAmbient von mehreren Computern und unter Einsatz verschiedener Eingabegeräte erlaubt. Ferner könnte die 3-dimensionale Arbeitsumgebung in der Anwendung auf verschiedenen Anzeigeflächen (Powerwall, Tabletop, etc.) aus unterschiedlichen Blickwinkel dargestellt werden. Schließlich erlaubt eine entsprechende Verteilung von DynAmbient einen entfernten Zugriff auf den virtuellen Arbeitsraum. Damit könnten auch Personen von unterschiedlichen Orten aus gemeinsam an Projekten arbeiten.

Abb. 5.2: Beispiel für eine verteilte 3-dimensionale Anwendung.

5.2.4 Fazit

Wie soeben in den Abschnitten 5.2.1 bis 5.2.3 erläutert, existieren vielfältige Möglichkeiten zur Erweiterung von DynAmbient.

Das vorliegende Programm ist ein Beispiel für eine physikbasierte Anwendung. Für eine zunehmende Verbreitung ähnlicher Applikation spricht zum einen die zunehmende Anzahl an Forschungsarbeiten, die sich mit diesem Thema beschäftigen. Ferner ist ein Großteil der Computer-Spieler inzwischen an die Simulation physikalischer Spieleumgebungen gewohnt und erwartet entsprechende aufwendige Effekte, was wiederum zu einer Verbesserung der Physik-Engines führen wird. Außerdem bietet die zunehmende Verbreitung von Geräten mit Touchscreens wie dem iPhone, dem Microsoft Surface Tisch oder dem HP TouchSmart PC geeignete Anwendungsumgebungen für physikbasierte Applikationen. Somit ist zu erwarten, dass Programme wie DnyAmbient in den nächsten Jahren zu den Standardanwendungen zählen werden.

Literaturverzeichnis

[Agarawala und Balakrishnan 2006] AGARAWALA, Anand ; BALAKRISHNAN, Ravin: Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In: *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA : ACM Press, 2006, S. 1283–1292. – ISBN 1-59593-372-7

[Albinsson und Zhai 2003] ALBINSSON, Pär-Anders ; ZHAI, Shumin: High precision touch screen interaction. In: *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 2003, S. 105–112. – ISBN 1-58113-630-7

[Apple Inc. 2007] APPLE INC.: *iPhone*. Webseite. 2007. – URL http://www.apple.com/iphone/. – Letzter Aufruf am 13. September 2007

[Benko u. a. 2006] BENKO, Hrvoje ; WILSON, Andrew D. ; BAUDISCH, Patrick: Precise selection techniques for multi-touch screens. In: *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA : ACM Press, 2006, S. 1263–1272. – ISBN 1-59593-372-7

[Burbeck 1992] BURBECK, Steve: *Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)*. Webseite. 1992. – URL http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html. – Letzter Aufruf am 14. November 2007

[Buxton u. a. 1985] BUXTON, William ; HILL, Ralph ; ROWLEY, Peter: Issues and techniques in touch-sensitive tablet input. In: *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM Press, 1985, S. 215–224. – ISBN 0-89791-166-0

[Chen u. a. 1988] CHEN, Michael ; MOUNTFORD, S. J. ; SELLEN, Abigail: A study in interactive 3-D rotation using 2-D control devices. In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM Press, 1988, S. 121–129. – ISBN 0-89791-275-6

[Davids 2007] DAVIDS, Arno: *Projekte im Bereich Sensornetze zur Gebäudeüberwachung im Rescue-Umfeld*. Seminararbeit an der Hochschule für Angewandte Wissenschaften Hamburg. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-aw/davids/report.pdf

[DiamondSpin Projekt 2006] DIAMONDSPIN PROJEKT: *DiamondSpin Tabletop Toolkit Project*. Webseite. 2006. – URL http://diamondspin.free.fr/. – Letzter Aufruf am 28. Novermber 2007

[Dietz und Leigh 2001] DIETZ, Paul ; LEIGH, Darren: DiamondTouch: a multi-user touch technology. In: *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM Press, 2001, S. 219–226. – ISBN 1-58113-438-X

[Dijkstra 1982] DIJKSTRA, Edsger W.: On the role of scientific thought. In: *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag, 1982, S. 60–66

[Erl 2004] ERL, Thomas: *Service-Oriented Architecture : A Field Guide to Integrating XML and Web Services*. Prentice Hall PTR, April 2004. – ISBN 0131428985

[Fischer 2007] FISCHER, Christian: *Entwicklung eines multimodalen Interaktionssystems für computergestützte Umgebungen*, Hochschule für Angewandte Wissenschaften (HAW) Hamburg, Masterarbeit, 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/fischer.pdf

[Fox u. a. 2000] FOX, Armando ; JOHANSON, Brad ; HANRAHAN, Pat ; WINOGRAD, Terry: Integrating Information Appliances into an Interactive Workspace. In: *IEEE Computer Graphics and Applications* 20 (2000), Nr. 3, S. 54–65. – ISSN 0272-1716

[Fraunhofer-Gesellschaft 2002] FRAUNHOFER-GESELLSCHAFT: *i-LAND - An Interactive Landscape for Creativity and Innovation*. Webseite. 2002. – URL http://www.ipsi.fraunhofer.de/ambiente/projekte/projekte/i_land.html. – Letzter Aufruf am 17. September 2007

[Gamma u. a. 1995] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995

[Gehn 2007] GEHN, Stefan: *Intuitive Gesten für Multitouch-Displays*. Unveröffentlichte Seminararbeit an der Hochschule für Angewandte Wissenschaften Hamburg. Dezember 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08/bericht.pdf

[Han 2005] HAN, Jefferson Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM Press, 2005, S. 115–118. – ISBN 1-59593-271-2

[Hinck 2007] HINCK, Steffen: *RESCUE: Überblick über Wearable Computing in extremen Situationen*. Seminararbeit an der Hochschule für Angewandte Wissenschaften Hamburg. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-aw/hinck/report.pdf

[Hollatz 2007] HOLLATZ, Dennis: *Einsetzbarkeit von AJAX-basierten Applikationen für kooperatives Arbeiten*. Bachelorarbeit an der Hochschule für Angewandte Wissenschaften Hamburg. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/hollatz.pdf

[IBM Corporation 2007] IBM CORPORATION: *Ease of Use - User-Centered Design*. Webseite. 2007. – URL http://www-03.ibm.com/easy/page/570. – Letzter Aufruf am 8. Januar 2008

[IPTC-NAA 1999] IPTC-NAA: *IPTC - NAA Information Interchange Model Version 4*. Spezifikation der International Press Telecommunications Council (IPTC) und der Newspaper Association of America (NAA). Juli 1999. – URL http://www.iptc.org/std/IIM/4.1/specification/IIMV4.1.pdf. – Letzter Aufruf am 30. Oktober 2007

[Ishii u. a. 1994] ISHII, Hiroshi ; KOBAYASHI, Minoru ; ARITA, Kazuho: Iterative design of seamless collaboration media. In: *Commun. ACM* 37 (1994), Nr. 8, S. 83–97. – ISSN 0001-0782

[Jacko und Sears 2003] JACKO, Julie A. (Hrsg.) ; SEARS, Andrew (Hrsg.): *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*. Mahwah, NJ, USA : Lawrence Erlbaum Associates, Inc., 2003. – ISBN 0-8058-3838-4

[JEITA 2002] JEITA: *Exchangeable image file format for digital still cameras: Exif Version 2.2*. Spezifikation der Japan Electronics and Information Technology Industries Association (JEITA). April 2002. – URL http://www.exif.org/Exif2-2.PDF. – Letzter Aufruf am 25. Oktober 2007

[Johanson und Fox 2002] JOHANSON, Brad ; FOX, Armando: The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In: *WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*. Washington, DC, USA : IEEE Computer Society, 2002, S. 83. – ISBN 0-7695-1647-5

[Köckritz 2007] KÖCKRITZ, Oliver: *Verteilter 3D-Desktop mit Remote-Windows für Collaborative Workspaces*. Seminararbeit an der Hochschule für Angewandte Wissenschaften Hamburg. Januar 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-aw/koeckritz/report.pdf

[Kruger u. a. 2003] KRUGER, Russell ; CARPENDALE, Sheelagh ; SCOTT, Stacey D. ; GREENBERG, Saul: How people use orientation on tables: comprehension, coordination and communication. In: *GROUP '03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*. New York, NY, USA : ACM Press, 2003, S. 369–378. – ISBN 1-58113-693-5

[Kruger u. a. 2005] KRUGER, Russell ; CARPENDALE, Sheelagh ; SCOTT, Stacey D. ; TANG, Anthony: Fluid integration of rotation and translation. In: *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 2005, S. 601–610. – ISBN 1-58113-998-5

[Kutak 2006] KUTAK, Edyta: *Entwicklung eines Location Tracking Systems für die Indoor Navigation*. Seminararbeit an der Hochschule für Angewandte Wissenschaften Hamburg. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/kutak/abstract.pdf

[Liu u. a. 2006] LIU, Jun ; PINELLE, David ; SALLAM, Samer ; SUBRAMANIAN, Sriram ; GUTWIN, Carl: TNT: improved rotation and translation on digital tables. In: *GI '06: Proceedings of the 2006 conference on Graphics interface*. Toronto, Ont., Canada, Canada : Canadian Information Processing Society, 2006, S. 25–32. – ISBN 1-56881-308-2

[Luban 2007] LUBAN, Pascal: *Physics in Games: A New Gameplay Frontier*. Artikel auf Webseite. Dezember 2007. – URL http://www.gamasutra.com/view/feature/2798/physics_in_games_a_new_gameplay_.php. – Letzter Aufruf am 15. Januar 2008

[Magerkurth u. a. 2003] MAGERKURTH, C. ; STENZEL, R. ; PRANTE, T.: *STARS – a ubiquitous computing platform for computer augmented tabletop games*. 2003. – URL citeseer.ist.psu.edu/magerkurth03stars.html

[Melzer 2007] MELZER, Ingo: *Service-orientierte Architekturen mit Web Services. Konzepte - Standards - Praxis*. Spektrum Akademischer Verlag, 2007

[Microsoft Corporation 2007a] MICROSOFT CORPORATION: *Microsoft Surface*. Webseite. 2007. – URL http://www.microsoft.com/surface/. – Letzter Aufruf am 13. September 2007

[Microsoft Corporation 2007b] MICROSOFT CORPORATION: *Model-View-Controller*. Webseite. 2007. – URL http://msdn2.microsoft.com/en-us/library/ms978748.aspx. – Letzter Aufruf am 22. Novemer 2007

[Mitchell 2003] MITCHELL, G. D.: *Orientation on Tabletop Displays*. Burnaby, British Columbia, Canada, Simon Fraser University, M.Sc. Thesis, 2003

[Napitupulu 2006] NAPITUPULU, Jan: *Indoor Map Server in einem Flughafenszenario*. Seminararbeit an der Hochschule für Angewandte Wissenschaften Hamburg. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/napitupulu/abstract.pdf

[Norman 1988] NORMAN, Donald A.: *The Psychology of Everyday Things*. Basic Books, 1988

[Perceptive Pixel 2007] PERCEPTIVE PIXEL: *Perceptive Pixel*. Webseite. 2007. – URL http://www.perceptivepixel.com/. – Letzter Aufruf am 18.Oktober 2007

[Piening 2007] PIENING, Andreas: *Katastrophen-Leitstand: Current work and projects*. Seminararbeit an der Hochschule für Angewandte Wissenschaften Hamburg. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-aw/piening/report.pdf

[Pierce und Nichols 2007] PIERCE, Jeffrey S. ; NICHOLS, Jeffrey: *Personal Information Environments*. Webseite. 2007. – URL http://www.almaden.ibm.com/cs/projects/pie/. – Letzter Aufruf am 16. Januar 2008

[Rekimoto 2002] REKIMOTO, Jun: SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In: *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 2002, S. 113–120. – ISBN 1-58113-453-3

[Rekimoto und Saitoh 1999] REKIMOTO, Jun ; SAITOH, Masanori: Augmented surfaces: a spatially continuous work space for hybrid computing environments. In: *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 1999, S. 378–385. – ISBN 0-201-48559-1

[Richter u. a. 2005] RICHTER, Jan-Peter ; HALLER, Harald ; SCHREY, Peter: Service-orientierte Architektur. In: *Informatik-Spektrum* 28 (2005), Oktober, Nr. 5, S. 413–416. – URL http://www.gi-ev.de/service/informatiklexikon/informatiklexikon-detailansicht/meldung/118/

[Ringel u. a. 2004] RINGEL, Meredith ; RYALL, Kathy ; SHEN, Chia ; FORLINES, Clifton ; VERNIER, Frederic: Release, relocate, reorient, resize: fluid techniques for document sharing on multi-user interactive tables. In: *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM Press, 2004, S. 1441–1444. – ISBN 1-58113-703-6

[Roßberger 2007] ROSSBERGER, Philipp: *Entwicklung einer Anwendung zur physikbasierten Manipulation von Objekten*. Seminararbeit an der Hochschule für Angewandte Wissenschaften Hamburg. Februar 2007

[Sasse 1997] SASSE, Martina A.: *Eliciting and Describing Users' Models of Computer Systems*, University of Birmingham, Dissertation, 1997. – URL http://www.cs.ucl.ac.uk/staff/a.sasse/thesis/Frontpage.html. – Unveröffentlichte Doktorarbeit

[Scott 2005] SCOTT, Stacey D.: *Territoriality in Collaborative Tabletop Workspaces*, University of Calgary, Dissertation, March 2005

[Sears und Shneiderman 1991] SEARS, Andrew ; SHNEIDERMAN, Ben: High Precision Touchscreens: Design Strategies and Comparisons with a Mouse. In: *International Journal of Man-Machine Studies* 34 (1991), Nr. 4, S. 593–613. – URL citeseer.ist.psu.edu/sears91high.html

[Shoemake 1992] SHOEMAKE, Ken: ARCBALL: a user interface for specifying three-dimensional orientation using a mouse. In: *Proceedings of the conference on Graphics interface '92*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1992, S. 151–156. – ISBN 0-9695338-1-0

[Soegaard 2007] SOEGAARD, Mads: *Mental models*. Webseite. 2007. – URL http://www.interaction-design.org/encyclopedia/mental_models.html. – Letzter Aufruf am 11. Januar 2008

[Streitz u. a. 1999] STREITZ, Norbert A. ; GEISSLER, Jörg ; HOLMER, Torsten ; KONOMI, Shin'ichi ; MÜLLER-TOMFELDE, Christian ; REISCHL, Wolfgang ; REXROTH, Petra ; SEITZ, Peter ; STEINMETZ, Ralf: i-LAND: an interactive landscape for creativity and innovation. In: *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 1999, S. 120–127. – ISBN 0-201-48559-1

[Tandler u. a. 2001] TANDLER, Peter ; PRANTE, Thorsten ; MÜLLER-TOMFELDE, Christian ; STREITZ, Norbert ; STEINMETZ, Ralf: Connectables: dynamic coupling of displays for the flexible creation of shared workspaces. In: *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM Press, 2001, S. 11–20. – ISBN 1-58113-438-X

[Tognazzini 1992] TOGNAZZINI, Bruce: *TOG on Interface*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1992. – ISBN 0201608421

[Wactlar und Christel 2002] WACTLAR, Howard ; CHRISTEL, Mike: Digital Video Archives: Managing through Metadata. In: *Building a National Strategy for Digital Preservation:*

Issues in Digital Media Archiving. April 2002, S. 84 – 99. – Commissioned for and sponsored by the National Digital Information Infrastructure and Preservation Program, Library of Congress

[Weiser 1991] WEISER, Mark: The Computer for the Twenty-First Century. In: *Scientific American* 265 (1991), S. 94–104

[Weiser 1996] WEISER, Mark: *Ubiquitous Computing*. Webseite. 1996. – URL http://www.ubiq.com/ubicomp/. – Letzter Aufruf am 13. September 2007

[Winograd 2003] WINOGRAD, Terry: *Stanford Interactive Workspaces Project Overview*. Webseite. 2003. – URL http://iwork.stanford.edu/. – Letzter Aufruf am 17. September 2007

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 21. Januar 2008	
Ort, Datum	Unterschrift