Projekt Micarpet
Projektbericht

Tina Soon Hai Ahlbrecht, Elmar Berger, Manfred Biess, Björn Breder, Patrick Breder, Markus Emde, Alexander Janot, Christopher Koschack, Tatiana Kotas, Christian Lauterbach, Manfred Meiss, Franck Ngueuleu, Gunnar Niehuis, Patrick Rodacker, Tobias Scheele, Benjamin Vaudlet und Volker Weinert

6. Januar 2005

Inhaltsverzeichnis

Teil I

Einleitung

Auf den folgenden Seiten ist der Inhalt des Projektes MiCarpet, einem studentischen Projekt des Fachbereiches 3 der Universität Bremen, zusammengetragen; der Inhalt eines zweijährigen Projektes, welches am 1. Oktober 2002 startete und zum 30. September 2004 endete. Innerhalb dieser Zeitspanne wurden viele Ergebnisse erzielt, viel Wissen angeeignet und schliesslich ein finales Produkt geschaffen. Am Ende ist es aber viel mehr als nur der Inhalt, der auf den folgenden knapp 300 Seiten dargestellt werden kann. Viele marginale Errungenschaften und Emotionen können im Folgenden nicht aufgeführt werden. Letztenendes ist das Projekt viel mehr, als dieser Bericht widerspiegeln kann.

Das Projekt MiCarpet war zunächst für eine andere Anzahl Teilnehmender ausgeschrieben, als sich im Laufe der Projektwahl hierzu anmeldeten. Von den knapp 66 MiCa-Interessierten hätten ursprünglich nur knapp 20 Projektteilnehmer übernommen werden können.

Motiviert durch die große Resonanz engagierte sich der Initiator Professor Dr. Friedrich-Wilhelm Bruns, die nötigen Möglichkeiten für die Interessenten zu schaffen. Er verpflichtete weitere Projektbetreuer und schaffte damit die Voraussetzungen zwei weitere Projekte. Die nunmehr drei Projekte sollten fortan als autarke, aber gleichberechtigte Projekte agieren. Die Teilnehmenden teilten sich auf die Teilprojekte MiCa-A, MiCa-B und MiCa-C auf. Später wurden die Titel dieser Projekte zugunsten einer eindeutigeren Erkennung umbenannt. Aus MiCa-C wurde später MiCarpet.

Für alle Teilprojekte bestand somit das gleiche Ziel – mit individueller Definition, was die Umsetzung betrifft. Ganz allgemein gehalten ist das Ziel des Projektes, eine so genannte CAVE mit Mixed-Reality-Teilen zu verwirklichen und die hierzu nötigen Komponenten selbst zu erzeugen oder zu organisieren.

Die Zieldefinition suggeriert ein breites Sepktrum an Komponenten, die in der CAVE zum Einsatz kommen. Sowohl mechanische und handwerkliche Aspekte, die zum Bau eines geschlossenen Raumes mit Projektionsflächen benötigt werden, als auch solche, die zur Softwareentwicklung benötigt werden, kommen zum Einsatz. Grafische Modellierung, Mechatronik und Interfacetechnik seien als weitere Komponente genannt.

MiCarpet als Titel entstand aus der Idee, eine Virtual-Reality-Umgebung kombiniert mit Interaktionsmöglichkeiten auf sensorischer und aktorischer Ebene und mittels der Spielidee eines fliegenden Teppichs umzusetzen. Um diesen Gedanken zu realisieren, arbeiten die Mitglieder in einer oder mehreren Arbeitsgruppen mit unterschiedlichen inhaltlichen Schwerpunkten zusammen. Ziel dabei ist es, Methodenwissen anzuwenden und in der Praxis einzusetzen, um die Realisierung auf allen Ebenen zu gewährleisten.

Um übersichtlich dabei vorzugehen, wurde ein Drei-Stufen-Konzept entwickelt, welches eine komplette virtuelle Umgebung, in der ein Nutzer unter Zuhilfenahme diverser Ein- und Ausgabegeräte interagiert, realisiert. Die Vorgehensweise und die o.g. Aspekte werden folgend kapitelweise dokumentiert. Hierbei wird in den ersten Kapiteln auf einzelne Gesichtspunkte des Projektes Bezug genommen, dann werden einzelne Komponenten detailliert erklärt und schliesslich wird ein chronologischer Ablauf des Projektes vorgestellt.

Dieser gewählte Aufbau soll ermöglichen, sich über spezielle Segmente oder Teile des finalen Produktes informieren zu können.

Teil II

Projektmanagement

Kapitel 1

Vorgesehene Termine im Projekt

Während der Laufzeit des Projektes finden diverse projektbezogene Veranstaltungen und Treffen statt. Als ordentlicher Projekttag ist der Freitag ausgeschrieben. An diesem Tag findet ein Plenum und eine Vorlesung statt. Des Weiteren wird ein Arbeitsvorhaben innerhalb der Gruppe abgehalten. Darüberhinaus wird das Projekt durch themenbezogene Vorlesungen, Semesteraufgaben und Projektwochenenden begleitet. Die vorgesehenen Termine intendieren den Fortschritt des Projektes und mehren sich im Laufe der vier Semester. Während des Projektes sind insgesamt 43 Plena protokolliert, an die 80 Arbeitsvorhaben abgehalten, 5 Veranstaltungen eigens moderiert und ein halbes Dutzend Projektwochenenden organisiert worden.

Nachfolgend werden die Termine und deren Inhalte der vier Semester dargestellt.

1.1 Plena

Das wöchentliche Plenum am Freitag dient vorrangig der internen Absprache. Zu diesem Termin treffen sich verbindlich alle Teilnehmer des Projektes MiCarpet. Betreuer ist Dr. Dieter Müller. Während das Plenum in der ersten Phase des Projektes als Sammelstelle kreativer Gedanken dient, verändert sich der zweckdienliche Schwerpunkt des Treffens im Laufe der darauffolgenden Semester mehr in Richtung Synchronisation der Teilgruppenarbeiten. Die genauen Inhalte des Plenums des viersemestrigen Projektes sind weiter unten dargestellt. Das Plenum wird gänzlich von der derzeitigen Organisationsgruppe abgehalten. Das vierköpfige Organisationsteam bestimmt selbständig einen Protokollant und einen Moderator. Der Moderator eröffnet das Plenum mit der Vorstellung der Tagesordnungspunkte, die sich im Laufe der Woche seit dem vorangegangenen Plenum gesammelt haben. Vorschläge für mögliche Tagesordnungspunkte dürfen von allen Mitgliedern über eine Veröffentlichung in dem hierzu angelegten *Thread* im eigens aufgesetzten Forum des Projektes gemacht werden. Vorrangig wird das Webinterface für die Veröffentlichung von Vorschlägen genutzt. Die Tagesordnungspunkte werden sequentiell unter Einbezug aller Projektteilnehmer abgearbeitet. Sofern in diesem Rahmen Beschlüsse für das weitere Vorankommen entschieden werden müssen, werden diese mit einem Mehrheitsvotum herbeigeführt. Unter Umständen lösen sich Tagesordnungspunkte im Laufe der Woche bereits auf; diese werden dann im Rahmen des Plenums nicht weiter berücksichtigt. Die Resultate des Plenums werden protokollarisch festgehalten und können so im Nachhinein noch eingesehen werden.

1.2 Arbeitsvorhaben

Als “Bärenanteil“ des freitäglichen *Projekttages* versteht sich das Arbeitsvorhaben. Dieses Segment hat nicht nur am Freitag zeitliche Anteile, sondern begleitet das Studium sporadisch über die gesamte Laufzeit des Projektes hin. Als Treffpunkt für das Arbeitsvorhaben dient vorrangig der Projektraum, der für das Projekt MiCa ausgewiesen wurde. Der Projektraum B1810 befindet sich im GW2 auf dem Universitätsgelände Bremen. Als an das Plenum anschliessende Veranstaltung werden hier zunächt die zuvor gefallenen Entschlüsse umgesetzt und in kleineren Arbeitsgruppen an bestimmten Schwerpunkten weitergearbeitet. Als Teilprojekt MiCarpet steht den Projektteilnehmern ein PC zur Verfügung, der in dem Projektraum bereit steht. Des Weiteren werden auf den knapp 30 m^2 alle Ergebnisse des Projektes erzielt; vom Bau des CAVE-Konstrukts bis hin zur Programmierung der Grafik-Engine. Aufgrund der Raumknappheit werden einige Vorhaben auf externe Facilitäten ausgelagert, so dass sich einige Aufgaben, wie beispielsweise Holzarbeiten nur in einer Werkstatt erledigen lassen o.ä..

1.3 Projektübergreifende, kollektive Nachmittagsveranstaltung Vorlesung

Zum Abschluß des *Projekttages* finden sich die drei über den Tag hinweg autark arbeitenden Teilprojekte zu einer Zusammenkunft im Senatssaal des MZH auf dem Universitätsgelände ein. Diese Veranstaltung bestärkt den Zusammenhalt des Projektes MiCa und soll dem Austausch der Teilprojekte untereinander helfen. Innerhalb dieser Veranstaltung werden themenorientierte Beiträge durch die Tutoren der einzelnen Projekte und dem dozierenden Professor Willi Bruns geleistet.

Die Moderation dieses Termins übernehmen in den letzten zwei Semestern die Studierenden, welche an dem Projekt teilnehmen selbst. Zyklisch wird dieser Termin in den einzelnen Teilgruppen organisiert. Das Projekt MiCarpet moderiert hierbei fünf Termine. Inhalte dieser Termine werden im nachfolgenden Punkt dargestellt. Der Projekttag endet mit dieser Veranstaltung um 17.00 Uhr.

1.3.1 Selbstmoderierte Termine

Die Inhalte der *selbstmoderierten Termine* werden ohne weitere Vorgaben seitens der Dozierenden entwickelt.

Im Rahmen der Abschlussveranstaltung sind folgende Themen durch das Projekt MiCarpet vermittelt worden:

- van Gogh : Felder
- EyeToy®
- SIGGRAPH
- Motion-Platform
- OGRE-Engine

Das Thema **van Gogh** ist in Anlehnung an das Auslaufen der Ausstellung *van Gogh: Felder. Das Mohnfeld und der Künstlerstreit* , welche vom 19. Oktober 2002 bis 26. Januar 2003 in der Bremer Kunsthalle residierte, aufgegriffen worden. Um einen Kontrast zu dem durch bewegte Bilder dominierten Inhalt des Projektes herzustellen, sind hier ganz bewusst künstlerische Abbildungen der Realität als Pendant zu der künstlichen Welt, die durch das Projekt hergestellt wird, gewählt worden. Als weiterer Hintergrund für die Aufbereitung dieser Ausstellung in Kurzformgalt es, der Allgemeinbildung zu gedenken. Inhalt

dieses Termins war ein reduzierte Darstellung der Ausstellung selbst. Einige Werke wurden musikalisch und textlich begleitet vorgestellt.

Abbildung 1.1: Kornfeld, die Mohnblumen, genannt Mohnfeld von van Gogh

Mit Vorstellug des **EyeToy**® von Sony Computer Entertainment Europe wird an die mögliche Verwendung von Tracking Systemen verfolgt. Zu diesem Termin wurde das Prinzip des EyeToys® erklärt und durch praktische Vorführungen mit Hilfe der Leihgabe eines Bekannten eines Teilnehmers unterstrichen. Es wurde dazu eingeladen, diese Anwendung zu testen.

Abbildung 1.2: Das EyeToy® im Einsatz

Aufgrund der kürzlich stattgefundenen **SIGGRAPH** 2003 und der Tatsache, dass ein Teilnehmer - Christian Lauterbach- diese Veranstaltung in San Diego als Voluntär begleitet hatte, war es möglich, die grösste Informatik-Konferenz der Welt in Form eines Erlebnisberichtes hautnah weiter zu geben. Es wurden sowohl grundlegende Informationen zur SIGGRAPH vermittelt, als auch einige Novitäten, welche diese hervorbrachte, in aller Kürze mit Hilfe von Fotografien und Erklärungen dargestellt.

Abbildung 1.3: Vorstellung des Fog-Screens

Mit Fertigstellung der **Motion-Platform** wurden alle Informationen um diese zusammengetragen und das Ergebnis vorgestellt. Hier wurde sowohl auf das endgültige Resultat, als auch auf diverse temporäre Lösungen, welche sich nicht bewährten, eingegangen. Die schliessliche Funktionsweise mittels pneumatisch funktionierender Zylinder wurden an diesem Termin vorgestellt. Als zentrales Element der Realisierung in dem Projekt MiCarpet fungiert die Motion-Platform als Steuerung des Systems. Detailliert wird auf die Plattform weiter unten eingegangen.

Abbildung 1.4: Finale Realisierung der Motion-Platform

In einem letzten Termin, der durch das Projekt MiCarpet in moderierender Weise bestritten wurde, sind die Vorteile der **OGRE-Engine** aufgegriffen worden. Hier wurde erläutert, wie die OGRE-Engine arbeitet und wie diese modifiziert werden musste, um an die Ansprüche des MiCarpet-Systems adaptiert zu werden.

Abbildung 1.5: OGRE-Engine

1.4 Lehrveranstaltungen

Neben dem indendierten Projekttag, welcher im wöchentlichen Turnus stattfindet, wurden die durch Prof. Willi Bruns empfohlenen Begleitveranstaltung des Informatik-Studiums rege besucht. Zu diesen Empfehlungen gehören folgende Veranstaltungen:

- Grafische Datenverarbeitung: 3D Szenen
- Einführungskurs in Aktorik, Sensorik und Mechatronik

Aufgrund einer sehr freien Formulierung der Aufgabenstellung des Projektes, ist es nicht möglich vorherzusagen, welche Domänen das System schliesslich ausmachen werden. Welche Anteile bei der Entwicklung berücksichtigt werden und in wie weit, stellt sich mit der Zeit heraus. Aus diesem Grund, sind nur wenige Lehrveranstaltungen wirklich themenverwandt. Zutreffend sind die beiden oben genannte Veranstaltungen. Je nach Grad der Berücksichtigung eines Themas in der projektbezogenen Umsetzung, wie z.B. die Programmierung der Engine o.ä. wurden noch weitere Veranstaltungen im Einzelnen empfohlen. Hierzu gehört auch die Veranstaltung Spieleprogrammierung, die von dem wissenschaftlichen Mitarbeiter im **[artec??]**, Martin Faust, doziert wurde.
In der Lehrveranstaltung *Grafische Datenverarbeitung: 3D Szenen* wurde durch Professor Frieder Nake die Programmierung in OpenGL erläutert und Grundlagen der dreidimensionalen Grafikverarbeitung vermittelt.
Durch Professor Willi Bruns wurden Grundkenntnisse der Aktorik, Sensorik und Mechatronik in einer begleitenden Lehrveranstaltung im ersten Projektsemester vermittelt.
Als weitere Möglichkeit der Wissensvermittlung dient in dem ersten Projektsemester eine für alle Teilnehmenden verpflichtende Vorlesung, welche zwischen Plenum und Arbeitsvorhaben gehalten wird. Dozent dieser Veranstaltung ist Professor Willi Bruns. Hier sind diverse Themen aufgegriffen worden. Ein breites Spektrum an in Anlehnung an das Thema des Projektes entwickelten Novitäten wurden in ihren Prinzipien und Funktion bereitgestellt; von Schaltskizzen einer SPS, über Motion Tracking bis hin zu bilderkennenden Verfahren.

1.5 Semesteraufgaben

Hervorgehend aus Inhalten der im ersten Semester gehaltenen Vorlseung von Professor Willi Bruns, sind folgende Aufgaben über zwei Semester hinweg, begleitend zu der Entwicklung des CAVE, durch alle Teilnehmer gelöst worden.

- Schaltplan eines Motorrades
- Soundkomponierung mittels AudioMulch©

Aufgrund der Unabdingbarkeit eines Steuermoduls für die Projektrealisierung, wird das Prinzip einer solchen erklärt. Als Aufgabenstellung soll diese Funktionsweise nachvollzogen werden und umgesetzt werden, indem die Steuerung eines Motorrades abstrakt dargestellt wird.

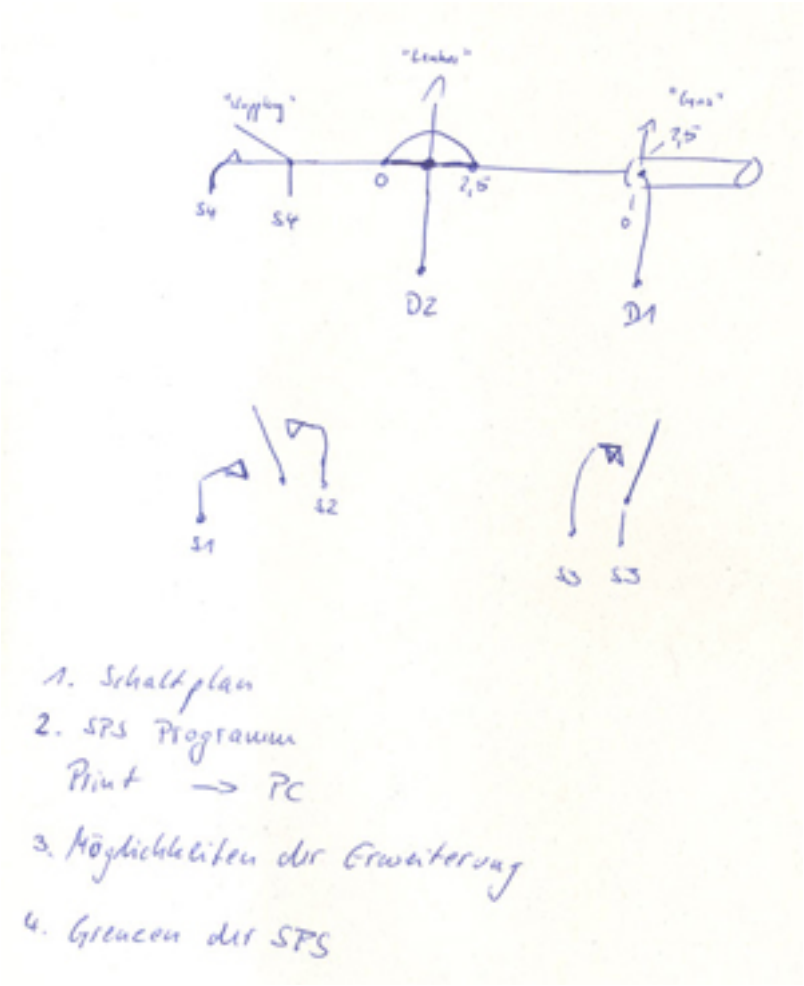


Abbildung 1.6: Skizze einer SPS Schaltung

Als suggestives Mittel zur Schaffung synästhetischer Eindrücke darf die Komponente Akustik in der Umsetzung eines jeden Teilprojektes nicht zu kurz kommen. Um das Augenmerk auf diese Teilkomponente zu rücken, wird hier ein interaktives Musikstudio vorgeführt. Das Release AudioMulch 0.9b12 wird hier als frei herunterladbare Software (http://www.audiomulch.com) genutzt. Jeder Teilnehmer des Projektes ist aufgefordert, einen kurzen *Sound* zu komponieren.

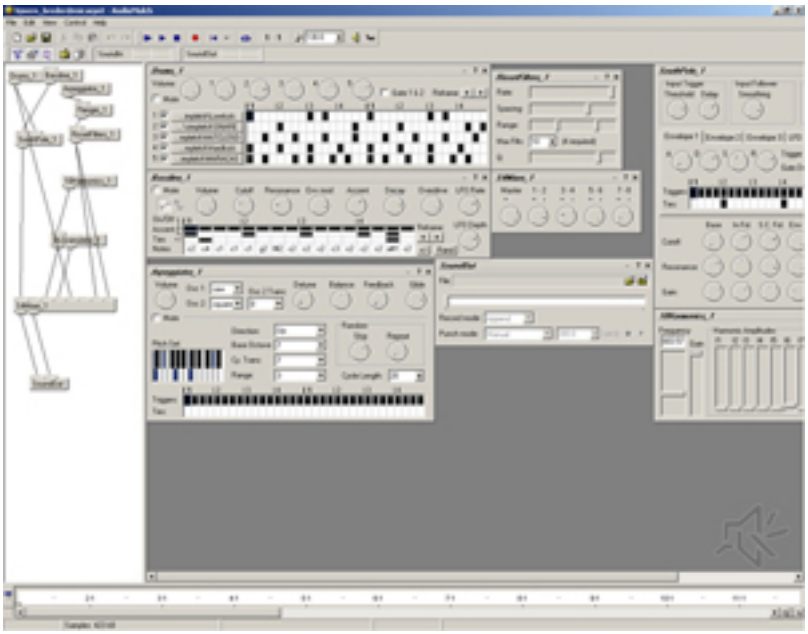


Abbildung 1.7: AudioMulch im Einsatz

1.6 Projektwochenenden

Projektwochenenden sind als intensive Arbeitsphasen im Kollektiv ausgeschrieben. Angedacht ist ein Projektwochenende pro Semester.
Während im ersten Semester noch ein externes Ziel für ein solches vorgesehen wurde, sind die darauffolgenden Projektwochenenden in der Universität Bremen abgehalten worden. Das erste Projektwochenende dient der Zielfindung und der Erschliessung eines Projektnamens. Dieses Wochenende findet im Ökozentrum Verden statt. Als Tagungsstätte bietet das Zentrum einen grossen Seminarraum mit bereitstehenden Schlafmöglichkeiten. Neben der Findung eines Projektnamens und der Ausreifung einer Idee wurde hier auch der Zusammenhalt aller Teilnehmenden gestärkt. Vorträge über das explizite Steuermodul, welches in der Umsetzung des Projektes MiCarpet zum Einsatz kommt, die SIOS und über den derzeitigen Entwicklungsstand der Motion-Platform begleiteten dieses Arbeitsphase. In den darauffolgenden zwei Projektwochenenden wurde als Lokalität die Universität Bremen genutzt. Durch die bestehenden Elemente des CAVE´s und die ausreichende Versorgung mit Internetanbindung und Strom erschien diese Wahl der Räumlichkeiten am geeignetsten. Die Realisierung konnte an diesen Wochenenden beträchtlich vorangebracht werden. In einzelnen Arbeitsgruppen wurde entweder extern, beispielsweise in einer Holzwerkstatt, heimisch in einer kleinen Gruppe, oder vor Ort gearbeitet. Im vierten und letzten Projektsemester wurde die Anzahl der Projektwochenende dem Bedarf entsprechend nach oben gesetzt. Durch das fehlende Zusammenspiel einzelner Komponenten des CAVE´s mussten intensive Testphasen ermöglicht werden. Diese Testphasen erfodern den kompletten Aufbau des gesamten Systems. Dies geht mit einem enormen Platzbedarf einher und fordert daher Freiraum, sowohl zeitlicher, als auch räumli-

cher Natur. Durch das Abhalten der Projektwochenenden konnten Fehleranalysen durch- und Lösungen herbeigeführt werden.

1.7 Abschlusspräsentation (Projekttag 2004)

Der Projekttag fordert den Einsatz aller Teilnehmenden und fördert das Ergebnis einer viersemestrigen Arbeit zu Tage. Der Projekttag versteht sich als Abschlussveranstaltung. Hier wird das Produkt im Einsatz vorgeführt und zum Testen eingeladen. Weiterhin werden Hintergrundinformationen angeboten.
Der Projekttag der Informatik 2004 fand traditionell im MZH der Universität Bremen statt. Alle Projekte, welche im Rahmen des Projektstudiums Informatik im Wintersemester 2002 begannen, stellten ihre Arbeiten auf der ersten Ebene des MZH-Gebäudes vor. Aufgrund des Platzmangels wurde diese Lokalität genutzt, um Hintergrundinformationen zu den Arbeiten des Projektes MiCarpet aufzuzeigen. Hier präsentierte sich das Projekt in Form von Plakaten, einer Bildschirmpräsentation und einer Live-Schaltung zu dem Universitäts-Theater. Das Universitätstheater wurde genutzt, um das System vorzustellen. Die Theaterbühne birgte ausreichend Platz, um die Konstruktion gänzlich aufzubauen und die Wände dieser zu projizieren. Hier konnten interessierte Besucher das System im Einsatz praktisch testen. Eine Kamera übermittelte einen Live-Stream in die erste Ebene des MZH-Gebäudes, welcher am Projektstand beobachtet werden konnte. Um Besucher zu dem Theater zu navigieren, wurde ein Handzettel mit einer Projektbeschreibung und weiteren Details, sowie einer Wegskizze zu dem Theater ausgeteilt. Ausserdem wurden auf dem Campus Wegweiser plaziert. In Form einer viertelstündigen Präsentation war zudem jedes Projekt angehalten seine Ergebnisse auf der Bühne des Senatssaals, ebenfalls im MZH-Gebäude, vorzustellen. Das Projekt MiCarpet setzte dieses in Form panthomimischen Theaters um. In Anlehnung an die Thematik des Spiels wurde eine Einleitung zu diesem im Freibeuter-Genre gespielt. Das Bühnenstück wurde ummantelt von zwei Videos, welche die Projektteilnehmer vorstellten und die Elemente des CAVE´s einzeln betrachteten. Alle Projektteilnehmer haben an diesem Tag zur Identifikation ein T-Shirt mit aufgedrucktem Projektnamen getragen. Das Corporate Design des Projektes wurde an diesem Tag zur Aufmerksamkeitserregung auch am Buffet des Projekttages aufgenommen.

Abbildung 1.8: Das Theater der Universität Bremen am Projekttag der Informatik 2004

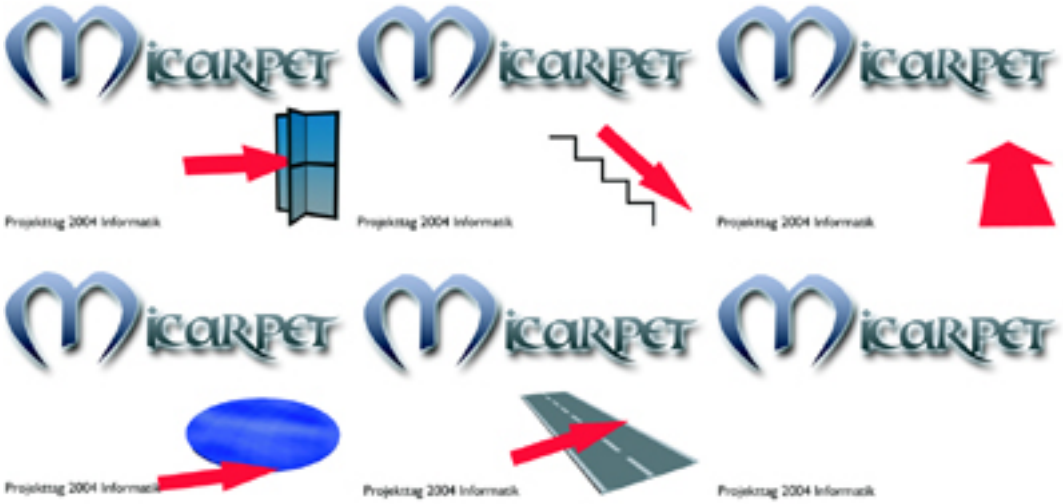


Abbildung 1.9: Die Wegweiser zu dem Theater

Abbildung 1.10: Das Projekt-T-Shirt

Abbildung 1.11: Der Corporate-Kuchen

Kapitel 2

Selbstregulation

Die Begrifflichkeit der Selbstregulation, oder auch Homöostase ist vor allem durch die Systemtheorie geprägt. In dieser die Selbstregulation eines Systems beschrieben durch “die Fähigkeit [...], sich durch Rückkopplung selbst innerhalb gewisser Grenzen in einem stabilen Zustand zu halten.“ [**?**]. Im Sinne dieser Definition versteht sich das Kollektiv, bestehend aus allen Teilnehmenden des Projektes MiCarpet, als das System. Eine Aufgabenstellung an das Projekt und gleichermaßen Freiheit für dieses ist es, selbstregulierend in der Gruppe zu arbeiten. Die oben genannte Selbstregulierung durchzieht mehrere Ebenen und wird auf verschiedene Weisen realisiert.
Zunächst muss im Sinne der Selbstregulierung sichergestellt sein, dass das Projekt weiterläuft und ein Ziel vor Augen hat. Weiterhin muss sichergestellt sein, dass Anforderungen, die an das zu realisierende System gestellt werden, nicht zu komplex werden und somit das Projekt zum Ausufern bringen. Auch müssen alle Ressourcen bei der Umsetzung des Projektes mit einbezogen werden und es muss gesichert sein, dass alle teilnehmenden Studenten alle Domänen, die das Projekt umfasst beherrschen.

2.1 Gruppenweises Arbeiten

Durch gruppenweises Arbeiten werden Expertengruppen gebildet. Diese sind verantwortlich für die Vermittlung des Wissens ihrer Domäne. Mit kurzen Referaten und Statusberichten werden andere Teilgruppen des Projektes aktuell gehalten. Mittels der Kleingruppen werden die einzelnen zu entwickelnden Komponenten des Systems überschaubar gehalten. Den Teilnehmenden ist eine leichte Strukturierung ihrer Inhalte möglich. Jeder Teilnehmende ist gleichsam angehalten in diesen Strukturen mitzuwirken. Jede Gruppe verfügt über einen Deligierten, welcher in so genannten Deligiertentreffen den Stand der Komponente anderen Deligierten mitteilen kann. Die Deligation kann in diesem Rahmen das Zusammenspiel der Komponenten überprüfen und somit vor einer Überstrukturierung des Systems schützen. Als deligierte Person können andere Teilnehmer der Gruppe ermahnt werden, ihr Engagement nicht zu vernachlässigen.

2.2 Motivation

Als Motivation für die Selbstregulation versteht sich die Notengebung. Die Resonanz auf die Arbeit in dem Projekt drückt sich in Form einer Benotung aus und fällt je nach Grad des eigenen Einbringens positiver aus. Auch hier gewinnt die Selbstregulation eine Bedeutung, wenn jeder Teilnehmer des Projektes durch die anderen Teilnehmenden beurteilt wird. Auf Grundlage einer Notenempfehlung durch den Betreuer Dr. Dieter Müller werden individuelle Abweichungen für jeden Teilnehmenden innerhalb der Gruppe besprochen. Die Selbstregulation soll als besondere Herausforderung verstanden werden.

Kapitel 3

Organisation

Die Organisation des Projektes wurde pro Semester auf jeweils vier Personen - das so genannte Organisations-Team - festgelegt. Dieses war dann für jeweils ein Semester für die Leitung des Projekts zuständig. Es übernahm Aufgaben des Projektmanagements, welche wie folgt definiert waren:

- Anleitung bzw. Vermittlung der Gruppen in den einzelnen Bereichen
- Überblick über die Ressourcen
- Verteilung von Aufgaben
- Protokolle der Plena und der Arbeitsvorhaben führen
- Moderation der Plena leiten

3.1 Erstes Semester

Im ersten Semester bestand das Organisations-Team aus Patrick Rodacker, Björn Breder, Christian Lauterbach und Tina Soon Hai Ahlbrecht. Dieses Team hatte für das Semester die Aufgabe, die an jedem Freitag stattfindenden Plena zu leiten und das Projekt voranzutreiben. Dazu wurde zu Beginn des ersten Semesters der Tagesplan festgelegt, der wie folgt aussah:

- 08.30 Uhr bis 10.00 Uhr Projekt Plenum (PP) in Raum SFG 2060
- 10.15 Uhr bis 11.45 Uhr Vorlesung bei W. Bruns in Raum MZH 1400
- 13:00 Uhr bis 16:00 Uhr Arbeitsvorhaben (AV) im Projektraum GW2 1810B
- 16:00 Uhr bis 17:00 Uhr Abschlussveranstaltung des Tages in Raum MZH 1400

Zusätzlich gab es im ersten Semester, dienstags ab 12:00 Uhr in Raum MZH 5210 oder im Projektraum, einen Zusatztermin.

Projektraum: Der Zugang zum Projektraum ist nur mit einem Schlüssel möglich. Dieser befand sich zu diesem Zeitpunkt in einem Schlüsselkasten, der im MZH in der Ebene 0, im Rechnerraumbereich sichtbar angebracht war. In diesem Kasten befand sich sowohl der Türschlüssel, als auch eine Key-Card, mit welcher der Zugang zum GW2 durch die Tiefgarage gesichert war. Um weitere Zugangskarten für

das Gebäude GW2 zu bekommen, wurden 10 Exemplare eines Berechtigungsschreibens von Prof. Dr. W. Bruns unterzeichnet und zur Verfügung gestellt. Jedes Mitglied, welches dieses Schreiben unterzeichnet hatte, bekam eine Karte. Mit Hilfe der Zugangskarten, war es möglich, das Gebäude auch außerhalb der öffentlichen Öffnungszeiten zu betreten.

Rechnersituation: Ausserdem musste ein Beschaffungsplan, über die zur Verfügung stehenden und die noch benötigten Ressourcen, aufgestellt werden. So wurde zunächst festgestellt, wie viele Studenten ein Notebook besitzen. Um weitere Rechner zu erhalten, musste eine gerechtfertigte Begründung an Niels Pollem abgegeben werden, wie viele Rechner, welcher Art für die Realisation des Projekts benötigt werden. Schließlich bekam das Projekt Mica-C eine Ausstattung, die einen neuen Rechner und zwei alte Rechner umfasste. Ein Beamer wurde für die Benutzung vom **[artec??]** zur Verfügung gestellt.

Projektwochenende: Projektwochenenden sollten das Projekt vorantreiben, bei dem die Mitglieder sich zum einen besser kennen lernen sollten, als auch effektiver arbeiten konnten. Am Ende des ersten Semesters fand ein solches gemeinsames Wochenende statt, in dem erste Referate vorgetragen und die Spielidee weiter konzipiert wurde.

Seminare und Referate: Nicht nur Projektwochenenden, sondern auch Seminare sollten zum Fortschreiten des Projektes beitragen. So wurde z.B. an einem Seminar zu dem Thema “Projektmanagement“ teilgenommen, als auch Eigeninitiative ergriffen und spezielles Wissen eines/einer Einzelnen an die anderen durch einen Vortrag oder gemeinsames Arbeiten weiter getragen. Die so behandelten Referate behandelten die folgenden Themen:

- 3D-Engine
- Spiegelung und Projektion
- Aktorik
- Pneumatik
- Sound
- Eingabegeräte

Projektkasse: Ebenso musste über die Einführung und die Pflege einer Projektkasse entschieden werden, um benötigte Materialien vorerst unabhängig von der Universität finanzieren zu können. Als Kassenwart wurde Benjamin Vaudlet berufen, der diese Augabe über die gesamten vier Semester übernahm. Auf Grund seiner viersemestrigen Arbeit wurde er von der Pflicht, Mitglied des Organisations-Teams zu sein, ausgeschlossen. Weiter bevollmächtigt für den Zugriff auf dieses Konto waren Manfred Bieß und Gunnar Niehuis.

Webseite: Es wurde Plattenplatz organisiert, auf dem im ersten Semester ein erster Entwurf einer Webseite eingerichtet wurde, auf der interessante Links zum Thema hinterlassen werden konnten, sowie ein allgemeines Forum zu finden war. Die Webseite war zunächst unter folgender Adresse zu finden: *http://mica-c.dyndns.org*. Daneben wurde eine gemeinsame Gruppe eröffnet, um eine Mailingliste aller Teilnehmer zu realisieren. Näheres zu dem Thema Webseite siehe unter 6.1 auf S.39ff.

Projektnote: Neben der einmal im Semester stattfindenden Rücksprache mit Dieter Müller über die Projekt-Note, wurden die Scheinkriterien bzw. Beurteilungskriterien für die Projektarbeit definiert:

- aktive Mitarbeit
- organisatorische Mitarbeit
- soziales Engagement
- fachliche Aktivität
- Produkt
- Projektbericht

Die Gruppennote wird am Ende des Projekts zuerst über die eigene Einschätzung der Studenten ermittelt und anschließend über die Einschätzung von Dieter Müller. Ebenso verhält es sich mit den jeweiligen Einzelnoten.

Prototyp: In der ersten Phase des Projektes war es das Ziel, einen ersten Prototyp zu entwickeln. Dieser wurde am 22.11.2002 um 14:30 Uhr im SFG vorgestellt. Vorab war es erforderlich, die Gruppe in mehrere kleine Untergruppen von etwa 4-5 Teilnehmern zu unterteilen. Intention dieser Aufteilung war es, detailliertere Ergebnisse einzelner Themengebiete wie z.B. Grafik, Sound, etc. zu erhalten, außerdem waren mehr Hintergrundinformationen erwünscht. Eine zunächst unverbindliche Gruppenaufteilung wurde wie folgt vorgenommen:

- Grafikapp. Besorgen/kennen lernen
- Hardware besorgen/einrichten
- Soundmöglichkeiten erfassen
- CAVE-Rahmen einrichten/entwerfen

Gruppenaufteilung: Nach der Vorstellung des ersten Prototyps wurde eine neue Gruppenaufteilung erstellt:

- Cave
  - Allgemeines
  - Konstruktion
  - Beispiele
  - Projektion
  - Beamer
  - Auflösung, Linsen, Abstände
- Grafik
  - Open GL
  - Wie funktioniert eine 3D-Engine

- Sensorik
  - Sensoren in magnetischen Feldern
  - Optische Sensoren
  - Tracking Systeme
- Aktorik
  - Interaktion in virtuellen Umgebungen
  - Menüführung im 3D-Raum
  - 3D-Mäuse und Eingabegeräte
- Sound
  - Surround Sound
  - Sound in bewegten 3D-Szenen

Projekthandbuch: Gleich zu Beginn des Projektes wurde ein Projekthandbuch (siehe Anhang A erstellt, in dem die Aufgaben der einzelnen Gruppen vorgestellt, die Aufgaben des Organisations-Teams, als auch des Kassenwartes erläutert, die Verantwortlichen bzw. der Ansprechpartner für die Webseite bekannt gegeben, die Zugangs-/Schlüsselordnung festgelegt, Informationen über den Projektraum beschlossen und schließlich Nutzungs- und Verhaltensregelen ausgemacht wurden.

3.2 Zweites Semester

Das Organisationsteam (Orga-Team) des zweiten Projektsemesters (Sommer Semester 2003) bestand aus folgenden Personen, die während eines gemeinsamen Projekt-Wochenendes in Verden ausgelost wurden: Gunnar Niehuis, Christopher Koschak, Tobias Scheele und Volker Weinert.

Nachdem die erste Orga-Gruppe hervorragend in das Projekt eingeführt hat, galt es im zweiten Semester die gewonnenen Erkenntnisse und Ergebnisse zu verarbeiten und umzusetzen. Organisatorisch gab es dabei soviel gar nicht zu verwalten.

Punkte, die in den Plena des zweiten Semesters ausführlicher diskutiert wurden:

Raumproblematik: Der uns zur Verfügung stehende Raum im GW2 konnte freitags unmöglich von allen Personen gleichzeitig genutzt werden, da er kein Fenster hatte und viel zu klein war. Einige Teilgruppen trafen sich dann in anderen Räumen, z.B. dem Studierraum im GW2, an anderen Tagen oder privat. Dadurch war die Anwesenheit einiger nicht mehr transparent, was zu einigem Unmut führte. Leider haben wir in den vier Semestern keinen neuen Raum bekommen.

Pünktlichkeit: Schon im vergangenen Semester war das Thema Pünktlichkeit auf jeder Tagesordnung. Leider verkam dieser Punkt im zweiten Semester zum „Running Gag“, was an der Tatsache an sich nur wenig änderte: Das Plenum wurde jedesmal durch Verspätungen von diversen Personen gestört, obwohl der Beginn meist deutlich nach hinten gelegt wurde.

Namensgebung: Am Ende des ersten Semesters wurde als Projektname „Orient Express“ von den Teilnehmern des Projektwochenendes gewählt. In den folgenden Plena rückte der Name wieder in den Mittelpunkt der Diskussion, weil man sich weder sicher war, ob er rechtlich geschützt ist, noch ob es der Spielidee - Geschicklichkeitsparkours mit einem fliegendem Teppich - in irgendeiner Weise nahe kam. Im Zuge neuer Überlegungen kam es dann zum endgültigen Namen „MiCarpet“. Dieser Name gefiel allen auf Anhieb und wurde ohne große Abstimmung von der Gruppe angenommen.

Logo: Parallel zur Namensfindung wurden von verschiedenen Mitgliedern Logovorschläge gemacht. Die ersten Versionen erinnerten dabei stark an Startbildschirme von Spiele- Klassikern. Für unser Corporate Design wurde allerdings ein druckfähiges Exemplar gebraucht. Die bis zuletzt gültige Fassung wurde aus zwei Vorschlägen kombiniert.

Grafikengine: Die Entscheidung für oder gegen die Madness-Engine wollten die an der Grafik- Implementierung beteiligen Personen nicht alleine fällen. Sie hatten sich bereits wegen ihrer Grenzen gegen die Madness-Engine und für die freie OGRE-Grafik-Engine entschieden. Die Entscheidung der Grafikgruppe wurde von der Gruppe getragen.

Webseite: Im Laufe des zweiten Semesters wurde der Internetauftritt überarbeitet und etwas professioneller gestaltet (siehe auch 6.2 auf S.41), und das Forum etwas besser in den Auftritt integriert. In den Plena wurden Vorschläge von der Gruppe abgesegnet.

Sponsorenmappe: Zur Aquise sollte eine Sponsorenmappe (siehe Punkt 7.1 auf S.43) angefertigt werden, in der sich das Projekt vorstellt. Vorschläge und Inhalte wurden diskutiert. Abgerundet wurde die Mappe mit Steckbriefen der Einzelpersonen, was wiederum zu Diskussionen führte.

Hardware: Erweiterung der Hardware um neue Grafikkarten hat die Gruppe eine Weile beschäftigt, da die Mittel nicht aus eigener Tasche sondern vom Fachbereich zur Verfügung gestellt werden sollten.

Regelmässiges: In regelmässigen Abständen haben die Teilgruppen über Ihr Vorankommen und Problemen dabei berichtet.

3.3 Drittes Semester

Im dritten Semester waren die meisten Teilgruppen mit der vorläufigen Umsetzung eigener Komponenten fertig. Letztere wurden in unregelmäßigen Abständen zusammen gebracht, um sie aufeinander anzupassen. Organisation der Cave-Tests war die bedeutendste Aufgabe des Orga-Teams in diesem Semester. Die Umordnungen der Teilgruppen, die sich durch Arbeitsaufwand ergaben, wurden in den Plena ausdiskutiert. Die Planung von Projektwochenenden brachte nicht so viele Umstände, da einige MiCarpet-Komponente relativ transportunfreudig geworden sind. Gearbeitet wurde an diesen Tagen an der Uni. So belief der Aufwand für uns betreffend Projektwochenendplanung nur in Organisation der Räumlichkeit und Festlegung der Projekttagtermine. Die Organisation der Plena, der Nachmittagsveranstaltungen, der benötigten Arbeitsmaterialien und vor allem die der außernormalen Projekttreffen blieb natürlich auch in diesem Semester in der Obhut des Orga-Teams.

3.4 Viertes Semester

Das Organisationsteam (Orga-Team) des vierten Projektsemesters (SoSe 2004) umfasste die Projektmitglieder Elmar Berger, Patrick Breder, Markus Emde und Franck Ngueuleu. Im letzten Semester galt es hauptsächlich, das erfolgreiche Fertigstellen aller Projektkomponenten voranzutreiben und alle Teile, aufeinander abgestimmt, zusammenzufügen. Als Endtermin stand dabei der Projekttag am 09. Juli 2004 im Vordergrund. Zudem musste dieser Projektbericht zum Abschluss gebracht werden.
Im Folgenden sollen die Eckpunkte dargelegt werden, die es zu organisieren galt.

3.4.1 Plena

Im Rahmen der Plena wurden im vierten Semester diverse Punkte, die in der Gruppe thematisiert werden sollten, angesprochen. Auf diese Weise sollte eine pünktliche Fertigstellung des Gesamtprodukts sichergestellt werden.

Pünktlichkeit Das in den letzten Semestern bereits diskutierte Problem der verspäteten Ankunft von Projektmitgliedern zu den Plena wurde auch im vierten Semester angesprochen. Um diesem Thema Nachdruck zu verleihen und es nicht dauerhaft zur Diskussion stellen zu müssen, wurden ab dem Plenum des 23. April die Anwesenheitszeiten der einzelnen Mitglieder protokolliert. Dies sollte der Motivation auch derer dienen, die ständig pünktlich erschienen und auch den gesamten Freitag durchgearbeitet haben. Von Seiten dieser wurde vermehrt Unmut bzgl. der Arbeitsmoral einiger Projektmitglieder an das Orgateam herangetragen.
Da jedoch die Protokollierung der Anwesenheitszeiten das Gruppengefüge zu sehr störte und eine Notengebung auf Grundlage der Anwesenheit vermutet wurde, stimmten die Anwesenden des Plenums vom 28. Mai gegen eine weitere Protokollierung.

Motivation Nicht nur über den Weg der Protokollierung der Anwesenheit, sondern auch durch zahlreiche Appelle der jeweiligen Moderatoren der Plena, wurde versucht zu motivieren. Mitgeteilt wurde, dass eine erhöhte Leistungsbereitschaft in der Gruppe entstehen sollte, um einen guten Abschluss zum Projekttag vorweisen zu können.

Gruppenzusammenhalt Im Rahmen der Motivationssteigerung sollte auch der Gruppenzusammenhalt gestärkt werden. Zu diesem Zweck war zum einen ein Projektwochenende außerhalb der Uni angedacht, zum anderen aber auch Projektwochenenden im Projektraum. Erstere Idee wurde jedoch verworfen, da die Platform-Gruppe eine Druckluftversorgung für ihre Tätigkeiten brauchte. Im Rahmen des gemeinsamen Arbeitens sollten Probleme anderer Projektmitglieder bei der Arbeit geklärt und diskutiert werden können.

Knappe Zeit Da vom Orga-Team die Zeit vom ersten Plenum bis zum Projekttag als sehr knapp erachtet wurde, um ein insgesamt lauffähiges System zu erstellen, sollte jede Teilgruppe einen individuellen Zeitplan erstellen. Darin sollten einzelne Meilensteine festgelegt werden, um auch die Arbeit der einzelnen Gruppen besser aufeinander abstimmen zu können.
Zudem wurden Termine festgelegt, an dem das Gesamtsystem oder einzelne Teilkomponenten gemeinsam getestet werden sollten.

Finanzen Auf Grund der Tatsache, dass 18 Monate in die Projektkasse eingezahlt wurde, wurde von einer ausreichenden Kapitaldecke ausgegangen. Auf Nachfrage stellte sich jedoch heraus, dass nur we-

nige Projektmitglieder den anfangs vereinbarten Zahlungen stets Folge geleistet hatten (siehe Anhang: Projekthandbuch). Diese Tatsache wurde im Plenum diskutiert und anschließend darüber abgestimmt, ob die Zahlungen nachgereicht oder vollständig ausgesetzt werden sollten.

Nachmittagstermine In diesem Semester hatte das Projekt zwei Nachmittagstermine zu gestalten, die gemeinsam mit den anderen Projekten („Micasa“ und „Micado“) begangen wurden. Für den ersten Nachmittag wurde die erste Gaming-Schnittstelle für die PlayStation 2 vorgestellt, bei der die Steuerung über Körperbewegungen erfolgt. Zum zweiten Nachmittag wurden die Komponenten und die Funktionalität der Motion Platform „Magic Carpet“ des Projekts vorgestellt. Eigentlich geplant war ein gemeinsames Grillen am Unisee, das jedoch auf Grund der fehlenden Teilnahme vieler Mitglieder der anderen beiden Projekte abgesagt wurde. Hierfür trat der oben genannte Alternativplan in Kraft.

3.4.2 Projektwochenenden

Am Anfang des Semesters in der vorlesungsfreien Zeit und vor dem Projekttag am 09. Juli 2004 wurden verstärkt Projektwochenenden organisiert, um das oben bereits erwähnte gemeinsame Arbeiten voranzutreiben. Gerade vor dem Projekttag stand das gemeinsame Testen und die damit verbundene Abstimmung des Gesamtsystems im Vordergrund. Durch das gemeinsame Testen konnten Systemfehler gefunden und behoben werden.

3.4.3 Projekttag

Für den reibungslosen Ablauf des Projekttags waren diverse organisatorische Hürden zu nehmen.

Außenminister Bereits früh im Laufe des Projekts wurde ein Abgesandter eines jeden Projekts zu einem Treffen geschickt, auf dem organisatorische Dinge diskutiert und Absprachen zwischen den einzelnen Projekten bezüglich des Projekttags getroffen wurden.

Räumlichkeiten Um auf dem Projekttag eine Räumlichkeit zur Präsentation des Gesamtsystems des Projekts „MiCarpet“ zur Verfügung zu haben, musste im Raumbüro eine entsprechende Reservierung vorgenommen werden. An diesen Raum wurden spezielle Anforderungen bezüglich Größe, Lage und Infrastruktur gestellt. Nachdem die Wahl auf das Uni-Theater gefallen war, musste dort ein Netzwerkanschluss gelegt werden, da eine ständige Live-Übertragung aus dem Uni-Theater heraus hin zum Stand in der Ebene 1 des MZH bestehen sollte.

Präsentationen Da eine ansprechende Bühnenshow und informative Präsenationen den Projekttag abrunden sollten, musste ein entsprechendes „Storyboard“ für die Show und ein Gesamtkonzept zur Außenwirkung der zu übermittelnden Informationen entwickelt werden. Daher wurde von den Projektmitgliedern in Heimarbeit überlegt, welche Form einer Bühnenshow die richtige sei, um Sympathien im Publikum zu erwerben. Da die Wahl auf eine Mischung aus Theaterstück und Videosequenzen gefallen war, mussten zum einen der Videodreh, zum anderen die Erstellung und Einprobung eines Theaterstücks organisiert werden.

PR-Material Um durch geeignete Medien die Informationen wirksam an das Publikum vermitteln zu können und Besucher in das Uni-Theater zu schleusen, wurde das PR-Team beaufragt. Geeignetes PR-Material waren Flyer, Poster, Banner, Hinweisschilder, aber auch T-Shirts und Bildschirm-Präsentationen.

3.4.4 Berlinfahrt

Nach der Einladung des projektbetreuenden Professors Friedrich W. Bruns galt es, die Fahrt in einem geeigneten Verkehrsmittel und die in Berlin zu verbringende gemeinsame Zeit zu organisieren.

3.4.5 Projektbericht

Um einen termingerechten Abschluss des Projekts auch nach dem Projekttag zu gewährleisten, wurde bereits früh der Projektbericht und seine Inhalte thematisiert. Zum einen wurden Konventionen vorgeschlagen, die seitens des Projekts abgelehnt wurden, zum anderen wurde eine Gliederung erstellt, die die Grundlage dieses Projektberichts darstellte, und durch mehrfache Diskussion in Plena abgerundet wurde.

Kapitel 4

Finanzmanagement

4.1 Finanzierungskonzept

Zur Handhabung der Finanzen wurde im ersten Projektplenum eine Person für den Posten des Kassenwarts gesucht und gefunden: Benjamin Vaudlet erklärte sich dazu bereit. Die primäre Aufgabe des Kassenwarts ist die Buchhaltung, welche das Überwachen der Ein- und Ausgänge des Projektkontos einschließt. Hierzu zählt auch das Auffordern der monatlichen Einzahlungen einzelner Teilnehmer auf das Projekt-Konto, sowie das Einfordern der universitären Vergütung, auf die im weiteren Verlauf noch eingegangen wird.

Die Aufspaltung des ursprünglich geplanten MiCa-Projektes, in drei verschiedene, autarke Teilprojekte, brachte konsequenterweise das Problem mit sich, dass die finanziellen Mittel, die à priori für ein einzelnes Projekt vorgesehen waren, auf eben diese drei Projekte umverteilt werden mußten. Die finanzielle Unterstützung von Seiten der Universität Bremen (4.4) war deshalb zunächst so geplant, dass für jedes Projektmitglied fünf Euro pro laufendes Semester zur Verfügung gestellt werden sollten, was in unserem Fall einem Gesamtbetrag von 340 Euro für zwei Jahre entsprochen hätte. Uns war schnell klar, dass es wohl sehr schwierig werden würde, mit diesem Betrag einen erfolgreichen Projektabschluß zu erreichen, auch was unsere eigenen Vorstellungen und Ansprüche an das Projekt betraf. Um an weitere Ressourcen zu gelangen, wurde die Idee der externen Förderung durch Spenden (4.2) geboren sowie der Beschluß gefaßt, neben einer einmaligen Einzahlung von zehn Euro im zweiten Monat (November 2002), fortlaufend für jeden weiteren Monat der Projektlaufzeit, eine Einzahlung in Höhe von fünf Euro durch jedes Mitglied des Projektes vorzunehmen.

Zur bequemeren Handhabung dieser Einzahlungen und den bevorstehenden Ausgaben wurde zu Beginn der Projektlaufzeit das bereits kurz erwähnte Projektkonto bei der Kreissparkasse Syke eröffnet. Auf das konkrete Vorgehen der Zahlungsweisen (4.3) sowie auf die Rückerstattungen (4.5) wird weiter unten genauer eingegangen.

4.2 Spenden

Durch die Restriktionen bezüglich des festgesetzten Budgets bestand von Anfang an das Interesse, die Ausgaben so gering wie möglich zu halten und über potentielle Anschaffungen erst einmal zu diskutieren, und so etwaige preiswerte Alternativprodukte zu erschließen. Des Weiteren wurde überlegt, an ausgewählte Unternehmen und Institutionen heran zu treten, die ein Interesse haben könnten, uns zu unterstützen. Die Unterstützung konnte gerne finanzieller (als Sponsor) wie auch materieller Art (als Partner) sein. Präsentationen der Zusammenarbeit zwischen dem Projekt, sowie den Sponsoren und Part-

nern fanden in unterschiedlichen Bereichen statt. So wurde für jedes Unternehmen, welches das Projekt in irgendeiner Form gefördert und unterstützt hat, das jeweilige Unternehmenslogo auf der Projektinternetseite (www.micarpet.de) untergebracht. Außerdem wurde bei bei allen öffentlich-wirksamen Präsentationen des Projektes diese Zusammenarbeit erwähnt, so auch bei der großen Abschlußpräsentation am Projekttag zu der auch alle Sponsoren herzlich eingeladen waren. Um an die Sponsoren heranzutreten, wurde zu gegebenen Anlässen eine eigens hierzu erstellte Sponsoren-Mappe (siehe 7.1) vorgelegt, welche alle für das Projekt relevanten Informationen enthält.

4.3 Zahlungsweisen

Nach der Eröffnung des Projektkontos und der Festlegung der geplanten Einzahlung jedes Studenten von fünf Euro pro Monat, wurde weiterhin festgelegt, dass die Auslagen für projektbezogene Anschaffungen mit einer Quittung belegt werden mußten. Diese gesammelten Quittungen waren zur Verwahrung bei dem Kassenwart abzugeben, welcher daraufhin umgehend die ausgelegten Gelder für Ressourcen auf die Konten der auslegenden Personen zurück überwies. Zur Einzahlung des monatlichen Beitrags waren folgende Angaben über das Projektkonto nötig:

Kontonummer 1160199160
Bankleitzahl 29151700
Institut KSK Syke
Empfänger Mica
Verwendungszweck Beitrag [Monatsname]
Summe 5,00 EUR

Für den Verwendungszweck galt es [Monatsname] durch den jeweiligen Monat, für den der Beitrag galt, zu ersetzen. Der monatliche Beitrag konnte auch durch eine entsprechende Summe, Monate mal Faktor fünf, im Voraus getätigt werden. Dies war dann auf der Überweisung entsprechend zu vermerken. Es wurde sich darauf geeinigt, dass eine pünktliche Einzahlung wünschenswert sei, sicherlich jedoch nicht verpflichtend. Bei finanziellen Engpässen seitens der Projektmitglieder, war es kein Problem, eine Überweisung, bei entsprechender Benachrichtigung des Kassenwarts, ausfallen zu lassen.

Es wurde beschlossen, dass die Auslagen für erforderliche Ressourcen erst nach Rücksprache mit dem Organisationsteam zu tätigen sind und nur durch die Vorlage einer Quittung zurückerstattet werden.

4.4 Universitäre Unterstützung

Relativ früh im ersten Semester des Projektes ergab sich neben den Spenden eine neue Möglichkeit, was die Tilgung der nötigen Ausgaben betraf. So wurde uns über unseren Projektbetreuer von Martina Braun (**[artec??]**) mitgeteilt, dass sich bemüht werden sollte, möglichst eine günstige Lösung zu realisieren, so dass die Ausgaben sich in Grenzen hielten, und bei jedem Einkauf die Quittung aufzubewahren. Diese konnten dann am jeweiligen Ende des Semesters, mit einem von Martina Braun angefertigten Antrag, im artec einreichen. Der Antrag wurde dann der Abrechnungsstelle der Universität vorgelegt, und das Geld wurde genau einen Monat nach der Einreichung auf das Projektkonto zurück überwiesen. Dieses System hatte sich gut bewährt; es wurde versucht, einerseits günstige Produkte zu erwerben, soweit es über die Sponsoren möglich war, andererseits wurde von der Universität am Ende eines jeweiligen Semesters das von unserer Seite ausgelegte Geld zurückerstattet.

4.5 Rückerstattungen

Zur Beantragung der Rückerstattungen über die Universität wurde ein dafür bestimmtes Formular verwendet. Zur Anschauung folgt nun dieses Formular für die Rückerstattungen, welches keiner genaueren Erläuterung bedürfen sollte. Aufgelistet sind dabei alle Ausgaben: das Kaufdatum, sowie das Geschäft, in dem die Waren erworben wurden.

Enrique-Schmidt-Str. 7 (SFG)
28359 Bremen
Tel.: 218-7307
Fax: 218-4449
Datum
XX.XX.XXXX

Universität Bremen
Forschungszentrum Arbeit-Umwelt-Technik
- artec -
Dezernat 3 / 302
Herr Carsten Meier
VWG

Auslagenerstattung

Sehr geehrter Herr Meier,

hiermit bitte ich um die Erstattung privater Auslagen, die ich für artec geleistet habe. Die Ausgaben fielen im Rahmen des studentischen Projektes “MiCarpet“ an.

Bezeichnung	**Datum**	**Betrag(EUR)**
Quittung von XXX für XXX	XX.XX.XXXX	XX,XX
Quittung von XXX für XXX	XX.XX.XXXX	XX,XX
Quittung von XXX für XXX	XX.XX.XXXX	XX,XX
Quittung von XXX für XXX	XX.XX.XXXX	XX,XX
Quittung von XXX für XXX	XX.XX.XXXX	XX,XX
Quittung von XXX für XXX	XX.XX.XXXX	XX,XX
Quittung von XXX für XXX	XX.XX.XXXX	XX,XX
Quittung von XXX für XXX	XX.XX.XXXX	XX,XX
Summe		XXX

Ich bitte Sie, den o. g. Betrag auf das

Konto Nr.: 1160199160
Kontoinhaber: Benjamin Vaudlet
Bank: Kreissparkasse Syke
BLZ: 29151700

zu überweisen.

Mit freundlichen Grüßen

Benjamin Vaudlet

Anlagen
1. Quittungen

Bezüglich der Rückzahlungen monatlich geleisteter Überweisungen seitens der Projektmitglieder zum Projektende, wurde stets durch den Kassenwart darauf geachtet, welche Eingänge von dem Teilnehmenden zu verbuchen waren. Die Rückzahlungen orientieren sich demnach an den geleisteten Anteilen. Der Kassenwart tilgte Auslagen unverzüglich. Das von den Mitgliedern monatlich eingezahlte Geld soll zum Ende des Projektes, nach Eingang der letzten Rückzahlung durch die Universität, (wie oben bereits erwähnt) anteilsmäßig und abzüglich der Kosten die nicht getilgt werden konnten, zurücküberwiesen werden.

Kapitel 5

Hardware-Organisation

Erhaltene Hardware Es wurden zwei PCs der unteren und ein PC der oberen Kategorie inklusive Monitore und Zubehör zur Verfügung gestellt. An den PCs der unteren Kategorie wurden Detailveränderungen vorgenommen. Insgesamt wurde folgendes Material zur Verfügung gestellt:

2x	PC untere Kategorie	Pentium II 233 MHz, 64 MiB Pentium II 400 MHz, 64 MiB
1x	PC oberer Kategorie	Pentium 4 2.0 GHz, 512 MiB, nVidia GeForce 4 Ti4200, 60GiB HDD
1x	Switch/Hub	8Port Switch (100MBit)
2x	SPS	Modul-Bus SIOS (vgl. 14, S.129)

Eigene Spenden Folgende Hardware wurde aus dem Bestand der Projektmitglieder zur Verfügung gestellt:

2x	PC untere Kategorie	AMD K6-2 500 MHz, 192 MiB Pentium 166 MHz
1x	Festplatte	6 GiB für den Projektserver
1x	Switch/Hub	5Port Hub (10MBit)
1x	Grafikkarte	Voodoo 3, 16MiB
1x	Lenkrad	ThrustMaster ForceFeedback Lenkrad
1x	Datenhandschuh	CP5 *2do*

Angeschaffte Hardware Folgende Hardware wurde von uns angeschafft:

5x	Patchkabel	Cat5e 20m
2x		Cat5e 5m
3x		Cat5e 3m
4x		Cat5e 2m
1x	serielles Kabel	RS-232 Verlägerung 5m
1x	Switch/Hub	Reichelt 8Port Switch (100MBit)

Defekte So wurden folgende defekte Komponenten ersetzt:

2x	Festplatte	eine von Beginn an defekt, die andere nach ca. 6 Monaten
1x	CPU Lüfter	defektes Lager, ersetzt
1x	Flachbandkabel	gebrochenen und getauscht
1x	Floppy	getauscht

Organisation Da lediglich ein Projektrechner in der Lage ist, eine relativ aktuelle Grafikleistung zu erbringen und die Mittel dazu fehlten bzw. sich durch Sponsoren keine ergeben haben, wurden die zwei fehlenden Rechner für das verteilte Rendering der Szene stetig im Wechsel durch verschiedene Notebooks der Projektmitglieder kompensiert.

Der Pentium II 233 MHz ist als Projektserver eingerichtet und bietet auf Basis eines aktuellen *Debian Woody* Systems die Dienste File- (Samba) und Webserver (**[Apache??]** + **[MySQL??]**), sowie eine Versionskontrolle (CVS) an.
Die weiteren Projektrechner wurden ebenfalls auf Basis von Debian Woody als Desktopsysteme bereitgestellt, später - bzw. auf dem Pentium 4 von Beginn an - wurde auf allen Rechner zusätzlich *Windows 2000* installiert.
Weitere Anschaffungen bezogen sich im Bereich der Hardware eher auf Kleinteile und Zubehör. Sofern diese nicht aus dem Privatfundus der Projektmitglieder zur Verfügung gestellt werden konnten, wurden die fraglichen Artikel in der Regel bei **[reichelt??]** bestellt.

Kapitel 6

Internetseite

Für die Außendarstellung des Projektes wurde durch das Organisationsteam des ersten Semesters entschieden, eine Internetpräsenz aufzubauen und online zu stellen. Verantwortlich für die erste Version waren Björn Breder, Tina Soon Hai Ahlbrecht und Patrick Rodacker.

Zu erreichen war die Internetseite ab dem 27. Oktober 2002 unter der Adresse:
`http://mica-c.dyndns.org`

6.1 Erste Phase

Die übergeordneten Ziele für die Internetseite waren die Bereitstellung einer Kommunikationsplattform neben dem Medium E-Mail und eine erste Anlauf- und Sammelstelle, um Projektmaterial auch interessierten Außenstehenden zugänglich zu machen.

6.1.1 Technik

Als Hardware diente einer der dem Projekt zur Verfügung stehenden Rechner. Für die Umsetzung der Internetseite wurde die sogennante LAMP-Architektur (Linux - **[Apache??]** - **[MySQL??]** - **[PHP??]**) verwendet. Als Betriebssystem des Servers wurde eine Linuxdistribution (Debian) eingesetzt, welche durch den Apache Webserver **[Apache??]** und MySQL **[MySQL??]** als Datenbankmanagementsystem ergänzt wurde. Für die Kommunikation zwischen dem Web- und Datenbankserver kam die Skriptspache PHP (PHP Hypertext Processor) **[PHP??]** zum Einsatz. Alle Komponenten werden unter der GPL [**?**] (GNU General Public Licence) veröffentlicht und sind kostenlos über das Internet erhältlich.

Bis auf das Forum (**[phpBB??]**) waren die gesamten Seiten eine komplette Eigenentwicklung. Das Layout ist auf der Abbildung 6.1zu sehen.

6.1.2 Struktur und Funktionalitäten

Die Struktur für die erste Version der Seite sah folgendermaßen aus:

- Aktuelles
  - Tagesordnungspunkte (TOPs)

Abbildung 6.1: Erste Version der Internetseite

  - letztes Protokoll
  - Forum
- Gruppe
  - Teilnehmerprofile
  - Orga-Team
  - Kassenwart
  - ProtokollantIn
  - ModeratorIn
  - Divisions
- Projekt
  - Beschreibung
  - Linksammlung
  - MiCa-Lexikon
  - Literatur
- Termine
  - Meilensteine
  - Protokolle
- Home

Neben dem Anzeigen von Informationen beinhaltete die Internetseite noch weitere Funktionalitäten für die Mitglieder, unter anderem die Informationen über die eigene Person, die Kommunikation im Projekt oder der Hinweis auf externe Informationsquellen:

- Eingabeformular für die persönlichen Daten der Mitglieder.
  Den einzelnen Mitgliedern wurde die Möglichkeit gegeben, bestimmte Angaben (Name, Alter, Telefonnummer, E-Mail-Adresse, etc.) zur Person zu machen und diese auf der Internetseite zu veröffentlichen.

Abbildung 6.2: Die Internetseite nach dem Relaunch

- Eingabeformular für den Vorschlag von neuen Links. Hier konnten Links zu externen Inhalten eingegeben werden, die der Eintragende sinnvoll für die gesamte Gruppe erachtete.
- Eingabeformular für das Wörterbuch. Hier konnten durch die Mitglieder Begriffe und deren Übersetzung eingegeben werden.
- Eingabeformular für die Protokolle.

6.2 Relaunch

Durch das Voranschreiten des Projektes wurden im Laufe der Zeit neue Anforderungen an die Internetseite definiert und ab Mai 2003 umgesetzt.

Das Projekt MiCa-C hatte zu dem Zeitpunkt ein konkretes Ziel definiert und dadurch auch den Projektnamen in *MiCarpet* geändert. Dies führte dazu, dass ein neues Logo, sowie ein komplett neues Layout der Internetseite entworfen wurde (siehe Abbildung 6.2). Zusätzlich wurde die Domain http://www.micarpet.de/ registriert und diente fortan als Plattform für die zweite Version der Internetseite. Nachfolgend werden nur die Neuerungen im Vergleich zu der ersten Version beschrieben.

6.2.1 Strukturänderungen

Da einige Inhalte in der Zukunft nur den Mitgliedern zugänglich gemacht werden sollten, wurde die Struktur in einen internen (zugangsgeschützt durch Benutzernamen und Passwort) und einen externen Bereich aufgeteilt.

Interner Bereich

- Forum
- Download
- Protokolle

Externer Bereich

- Home
- Kontakt
- Fakten
- 3-Stufen-Plan
- Gruppen
- Mitglieder
- Sponsoring
- Lexikon
- Links
- Download
- Impressum
- Disclaimer

6.2.2 Neue Funktionalitäten

Neben den von der ersten Version übernommenen Funktionalitäten sind beim Relaunch weitere hinzugekommen. Des Weiteren ist unter den oben genannten Punkten wesentlich mehr Inhalt online gestellt worden und ermöglicht somit eine umfangreiche Darstellung des Projektes. Zu den Inhalten zählen z.B. die Projektmappe (siehe auch Punkt 7.1 auf S.43) und der 3-Stufen-Plan (siehe auch Punkt 9.7 auf S.57).

- Zugang für den geschützten Bereich via Benutzername und Passwort.
- Einfach zu verwaltender Downloadbereich für die Sammlung von Dokumenten und anderen für die Mitglieder interessanten Dateien.
- Direkter Zugang zu dem Forum (angemeldeter Status) aus dem internen Bereich heraus.
- Erweiterte Protokolleingabe mit flexibleren Eingabemöglichkeiten bzgl. der Anzahl der TOPs und deren Inhalte.
- Übersicht aller Protokolle.

Kapitel 7

Public Relations

Die Darstellung des Projektes nach außen und die Suche nach möglichen Sponsoren waren innerhalb des ersten Jahres die Hauptaufgabe der PR-Gruppe. Zum Ende des Projektes rückten die Erstellung von PR-Material (Flyer, Poster, Wegbeschreibungen, Pressemitteilungen, Präsentationen) in den Vordergrund.

7.1 Sponsoren-/ Projektmappe

Für die Präsentation des Projektes bei möglichen Sponsoren und Interessierten wurde eine Mappe entworfen, in der neben einem einseitigen Faktenblatt auch die Mitglieder mit entsprechenden Profilen (Persönliche Daten, Arbeitsschwerpunkte, zukünftige Arbeitsfelder) aufgeführt wurden.

7.2 Praxisbörse der Uni Bremen am 19.06.2003

Auf der Praxisbörse der Universität Bremen im Juni 2003 war die PR-Gruppe unterwegs, um mit mehreren Firmen den Kontakt aufzunehmen und nach Sponsoren zu suchen. Hierbei dienten die Sponsorenmappen zur Darstellung des Projektes und wurden an mehrere Firmen ausgehändigt. Dies waren unter anderem folgende Unternehmen:

- arcelor Flachstahl bei Stahlwerke Bremen
- Techniker Krankenkasse TK
- Focke & Co Packing Machinery
- Leewood Elastomer Silicone Products
- Daimler Chrysler
- Kraft Foods

Die direkten Kontakte auf der Messe wurden von Seiten der Firmen sehr positiv aufgenommmen, jedoch gab es bis auf zwei negative Rückmeldungen keine weiteren Schritte in Richtung Kooperation, bzw. Unterstützung.

7.3 Weitere Sponsorensuche

Nach der Auswertung der Erfahrungen bei der Suche nach Sponsoren auf der Praxisbörse wurde das weitere Vorgehen in der PR-Gruppe abgestimmt und sollte in Form von telefonischen Kontakten umgesetzt werden. Hierfür wurde ein Ablaufleitfaden entworfen, um auch die anderen Teilnehmer des Projektes in die Akquisetätigkeiten mit einbinden zu können. Der Ablauf war in folgende Unterpunkte aufgeteilt:

1. Unternehmensrecherche
2. Telefongespräche
3. Versand der Projektmappe
4. Auswertung der Rückläufe

Für die Telefongespräche wurde ein Telefonleitfaden entwickelt, der es den Mitgliedern ermöglichen sollte, gezielt nach den für uns wichtigen Informationen zu fragen (Ansprechpartner, Daten für den Versand der Mappe, etc.). Ein mögliches Gespräch sollte demnach wie folgt aufgebaut sein:

1. Ermittlung des entsprechenden Ansprechpartners.
2. Vorstellung der Person und des Projektes.
3. Aufnahme der Daten der Kontaktperson zur Versendung der Projektmappe.
4. Verabschiedung von der Kontaktperson,
5. Festlegung auf einen weiteren Gesprächstermin (telefonisch, per E-Mail, optional auch persönlicher Besuch bei regionalen Unternemen).

Durch das geringe Interesse innerhalb des Projektes, diese Aufgaben zu übernehmen, wurde die organisierte Sponsorensuche in diesem Rahmen kurze Zeit später eingestellt und beschränkte sich fortan auf die privaten Verbindungen, welche durch die einzelnen Teilgruppen hergestellt wurden.

7.4 Sponsoren des Projektes

Folgende Sponsoren haben das Projekt unterstützt und verdienen Dank:

- Tischlermeister Björn Ritschewald (siehe auch Abbildung 7.1 auf S.45)
  Neben der Ausstattung mit Material für den Cave, hatten wir die Möglichkeit die Werkstatt von Björn Ritschewald, sowie weitere Materialien für die Projektwochenenden und den Projekttag zu nutzen.
- Ahlrich Siemens GmbH & Co. KG (siehe auch Abbildung 7.2 auf S.45)
  Die Firma Ahlrich Siemens stellte uns die kompletten Teile für die Motion-Platform zu sehr guten Konditionen zur Verfügung.
- Route Orientteppiche
  Der Teppich für die Motion-Platform wurde von der Firma Route günstig zur Verfügung gestellt.

Abbildung 7.1: Sponsor: Björn Ritschewald - Der Tischlermeister

Abbildung 7.2: Sponsor: Ahlrich Siemens GmbH & Co. KG

7.5 Projekttag

Zur Vorbereitung des Projekttages war die PR-Gruppe sowohl für die projektinternen Materialien, als auch für die projektübergreifende PR-Arbeit zuständig.

7.5.1 Projektinterne Vorbereitung

Für den Projekttag wurden neben verschiedenen Plakaten, Wegweisern und Flyern auch eine Präsentation für die Rechner am Projektstand durch die PR-Gruppe erstellt. Näheres hierzu ist dem Punkt 9.10 (S.59ff.) zu entnehmen.

7.5.2 Projektübergreifende PR-Arbeit

Im Zuge des Projekttages, an dem alle Projekte des entsprechenden Semesters des Hauptstudiums Informatik ihre Ergebnisse aus zwei Jahren Projektarbeit präsentierten, war die PR neben der projektinternen PR-Arbeit auch für die Öffentlichkeitsarbeit für den Projekttag verantwortlich. Daraus ergaben sich folgende Aufgaben:

- Pressemitteilungen
  Neben der Ankündigung auf dem E-Mail-Verteiler des Studienganges Informatik wurden auch zwei Pressemitteilungen auf den Webseiten `http://www.informatik.uni-bremen.de` und `http://www.uni-bremen.de` veröffentlicht. Die Mitteilung enthielt neben dem Zeitplan auch eine Auflistung der Projekte mit einer kurzen Beschreibung und einem Link zu deren Webseiten.

- Plakate
  Die Veranstaltung des Projekttages sollte im MZH stattfinden und entsprechend durch Plakate

Abbildung 7.3: Plakat für die projektübergreifende Ankündigung des Projekttages

auf dem Campus beworben werden. Der Entwurf (siehe Abbildung 7.3) und das Bereitstellen der Vorlage zum Ausdruck wurde ebenfalls durch die PR-Gruppe übernommen.

Teil III

Projektablauf

Kapitel 8

Übersicht der Entwicklungsphasen

Das Projekt wurde über die zwei Jahre durch verschiedene Phasen geprägt, welche aufeinander aufbauten, oder parallel durchlaufen wurden. Im folgenden Abschnitt werden diese charkteristischen Phasen kurz beschrieben.

Prototyp Die Aufgabe innerhalb der ersten 4 Wochen nach Projektbeginn war die Erstellung eines Prototypen. Als System für die Komponente Virtual Reality diente die Grafikengine **[Madness??]** von Martin Faust (**[artec??]**). Dementsprechend wurde ein Großteil der Arbeitszeit auf die Planung des Cave verwendet. Da die Materialen des Prototypen auch für die endgültige Version des Cave verwendet wurden, war diese Phase zwar nach 4 Wochen offiziell beendet, jedoch wurden die Erfahrungen auch in die anderen Phasen hereingetragen. Näheres zu dem Prototypen ist dem Punkt 9.3 zu entnehmen.

Wissensaneignung Vom ersten Tag des Projektes an wurde sich auf unterschiedliche Weise thematisch mit den Inhalten auseinandergesetzt. Die von den Dozenten und Betreuern rund um Prof. Dr. W. Bruns angebotenen Optionen umfassten z.B. den Besuch einer Berufsschule für ein Einführungstutorium im Bereich Pneumatik, das Angebot an Büchern und wissenschaftlichen Arbeiten (z.B. Seminarunterlagen [**?**], [**?**]) im Projektraum des **[artec??]**. Die Wissenaneignung ist eine der Phasen, welche permanent von den Teilnehmern permanent aktiv durchlaufen wurde. Am Anfang eher in der Form der Einarbeitung in die Materie Mixed (Virtual) Reality, CAVE und Sensorik und zu einem fortgeschrittenen Zeitpunkt dann zu ganz speziellen Themen, welche durch Aktivitäten in den Teilgruppen zu bearbeiten waren; wie die unterschiedlichen Punkte der Wissensaneignung im Einzelnen aussahen, beschreibt der Punkt 9.6.

Ideen- und Zielfindung Das Ziel des Projektes war durch das Angebot von Prof. Dr. W. Bruns nur in einer groben Richtung vorgegeben, und zwar der des Themas Mixed Reality. Wie die detaillierte Umsetzung innerhalb dieses Themas aussehen konnte, war vollkommen freigestellt. Somit wurde in der Phase der Ideen- und Zielfindung vom Brainstorming bis zu einer Anforderungsdefinition (Am 24. April 2003 wurde die Anforderungsdefinition (siehe Anhang B im Projektplenum verabschiedet und diente somit als schriftliche Ausgangspunkt für die Entwicklung des Gesamtsystems.) versucht eine Umsetzung des Projekts zu finden, mit der sich alle Teilnehmer identifizieren konnten. Der genaue Ablauf dieser Phase ist in Punkt 9.4 aufgeführt.

Entwicklung des Gesamtsystems Nachdem die Ausrichtung des Projektes festgelegt und somit die Hauptarbeit in die Teilgruppen (siehe Punkt 9.5) verlagert wurde, kam es in dieser Phase zu der Entwicklung des Gesamtsystems. Einen Überblick des Systems vermittelt IV. Die einzelnen Komponenten des Systems sind in Kapitel V detailliert beschrieben.

Kapitel 9

Chronologischer Ablauf des Projekts

Im bisherigen Teil wurde sich auf einzelne Komponente des Systems, auf einzelne Bereiche und verschiedene Gesichtspunkte des Projektes konzentriert.
Nachfolgend wird der chronologische Ablauf des Projektes dargestellt, um die Zusammenhänge und einzelnen Entstehungsphasen deutlich zu machen. Die nachfolgenden Kapitel sind in einem chronologischen Stil angeordnet und sollen sich als roter Faden durch das viersemestrige Projekt verstehen. Angefangen bei den Machbarkeitsüberlegungen im ersten Projektsemester, der Definition einer Thematik, über die Instanziierung der Teilgruppen im zweiten Projektsemester, bis hin zu dem abschliessenden Projekttag am Ende des vierten Semesters, werden hier alle Phasen der Entwicklung durchlaufen.

9.1 Machbarkeitsüberlegungen

Mit Überlegungen zu realistischen Umsetzungen in den Grenzen der Thematik beginnt das viersemestrige Projekt MiCa. Dieses Projekt wird zunächst namentlich als Teilprojekt MiCa-C geführt, welches später zu seinem Namen MiCarpet kommt. In Anbetracht der ersten Aufgabenstellung, einen Prototyp eines CAVEs zu entwickeln, wurde überlegt, welche Konstruktion einfach zu handhaben ist. In die Überlegungen flossen bereits verschiedene Berücksichtigungen mit ein. Attribute des CAVEs sollten Portier- und Wiederverwendbarkeit sein. Verscheidenste, nachfolgend dargestellte Konstruktionen wurden hierbei durchdacht. Fragestellungen waren, wie viele Seiten die CAVE bekommen soll und welche Grösse diese haben müssen. In Hinsicht auf die Phase nach der Konstruktion eines Prototyps wird bereits bedacht, Grundelemente des Prototyps weiter zu verwenden. Die Machbarkeitsüberlegungen manifestieren sich schliesslich in individuellen Ausarbeitungen der einzelnen Teilnehmer. Diese werden im Appendix C vorgestellt.

9.2 Definition einer Thematik

Die Thematik des schliesslich zu fertigenden Produktes wurde nach der ersten Phase, dem Bau eines Prototyps, fortgeführt. Nachdem die Grundstimmung mehrheitlich für eine spielerische Umsetzung eruiert wurde, waren alle Teilnehmenden aufgerufen, die Spielethematik auszureifen. Einschränkungen wurden hinsichtlich einer gewaltfreien und nicht anstößigen Atmosphäre gemacht.

9.3 Prototypische Realisierung

Abhängig von der Raumhöhe, in der der Cave vorgestellt werden sollte, besprach sich die Projektgruppe über die Ausmaße des Prototypen. Neben der Angleichung an die Maße des Raumes, musste das Seitenverhältnis der Beamer beachtet werden. Diese hatten ein Seitenverhältnis von 4:3. Es wurde über verschiedene Cave Möglichkeiten gesprochen, wobei angeführt wurde, dass ein Cave in verschiedenen Arten erstellt werden kann, z.B. ohne Decke oder nur mit drei Wänden, mit Spiegeln, etc. Es wurde daran erinnert, dass Prof. Dr. W. Bruns eine modulare Gestaltung vorschlug.

Patrick Breder bezog Kontakt zu dem Tischler Björn Ritschewald, der als Sponsor auftrat. Dieser stellte sich zur Verfügung, Materialien für den Cave bereit zu stellen, diese anzufertigen, als auch zu einem Termin in der Universität zu erscheinen, um bei der weiteren Konstruktion bei der Planung zu helfen. Für den ersten Cave hatte sich die Gruppe entschlossen, einen vierseitigen Cave zu entwickeln, der an drei seitlichen Wänden Projektionswände besitzt. Das Grundgerüst des Caves bestand aus Holz. Vier Holzbalken bildeten die Pfosten für die acht oben und unten quer liegenden Leisten. Die Projektionsflächen wurden durch eine Gummiband-Halterung realisiert. Dazu wurden an den oberen und unteren Leisten viele nebeneinander liegende Nägel angebracht. An den Projektionsflächen wurden an der oberen und unteren Seite genauso viele Löcher eingestanzt und durch diese jeweils ein Gummiband gespannt. Die Gummibänder wurden dann an den Nägeln oben und unten angebracht.

Die drei Projektionswände bestanden aus 2 Bahnen Pergamentpapier, wobei jede Wand jeweils die Maße 200 cm (Höhe) x 220 cm (Breite) hatte. Die vier Eckpfeiler waren jeweils 220 cm hoch und sowohl 4 cm breit, als auch 4 cm tief. Die zwei Fußleisten waren 200 cm hoch, 4 cm breit und 1 cm tief. Die acht Decken-/Bodenleisten waren 208 cm lang, 5 cm breit uns 1 cm tief. Ansonsten benötigten wir durchsichtiges Klebeband, um kleine Risse im Pergamentpapier abkleben zu können.

9.3.1 Hindernisse

Das größte Hindernis in der ersten Entwicklungsphase war die Anschaffung geeigneter Projektionsflächen. Die Schwierigkeit bestand vor allem in der Größe, die für den Cave erforderlich war. Ausgehend von der Idee, die Projektionsflächen aus Papier zu gestalten wurde versucht entsprechend große Papierflächen zu bekommen. Leider konnte kein Papier entsprechender Größe besorgt werden. Deshalb einigte man sich vorerst auf den Kompromiss, jeweils zwei Flächen für eine Wand zu benutzen. Dieses sollte jedoch bis zum Ende des Projekts geändert werden.

9.3.2 Reflexion der Nachteile

Nachteile in der ersten Konstruktion des Caves, lagen vor allem bei den Projektionsflächen. Zum einen mussten jeweils zwei Bahnen für eine Projektionswand benutzt werden, was den Nachteil brachte, dass die Flächen in keinem Fall so angeordnet werden konnten, dass man keinen Übergang sah. Zum anderen die aufwändige Anbringung der Projektionsflächen durch die große Anzahl der Gummibänder. Zwar war die Qualität der Projektion durchaus zufrieden stellend, doch leider gab es ein solches Material nicht in einer erforderlichen Größe. Das Prinzip der Projektionsflächen musste völlig neu überdacht werden, da diese den Strapazen eines häufigen Wiederaufbaus der Anlage nicht auf Dauer überstehen würden.

Des Weiteren wurde der Raum kritisiert, in dem die Präsentation stattfand. Auf Grund der Maße des Raums, konnten nur zwei Flächen angestrahlt werden, was zu einem unbefriedigenden Ergebnis führte.

Die Holzbalken wurden in der Gruppe als zu dick beurteilt. Das führte zu dem Wunsch, den Cave im Endstadium noch leichter und mobiler zu gestalten.

9.3.3 Recycelbare Bestandteile (Vorteile)

Vorteile dieser Konstruktion war das schnelle Auf- und Abbauen, als auch die gute Qualität der Projektionsflächen, die für den weiteren Verlauf des Projekts noch länger von Nutzen waren. Die Holzkonstruktion des Gerüsts erwies sich dabei schon als so ausgereift, dass im späteren Verlauf nur noch kleine Veränderungen vorgenommen werden mussten und dieses erste Gerüst bis zum Ende des Projekts genutzt werden konnte.

Das Projekt sah sich im Vergleich zu den beiden anderen Mica Projekten zu diesem Zeitpunkt schon als recht weit fortgeschritten.

9.4 Ausreifung des Inhalts („Spielideen“)

Nach einem ersten Brainstorming, kamen folgende Ideen zur Erstellung eines Caves zustande:

- Rennsimulation
- Entspannungscave
- Renn-/Flugsimulation mit erweiterter Steuerung
- Motorrad- oder Snowboardsimulation unter Verwendung von Pneumatik
- Bauernhof
- Fliegender Teppich mit Plattform
- Cave mit einer großen Vielfalt an Sensoriken
- Flugsimulation (Cockpit bietet viel Varianz)
- Tauchen (viel Umgebungsvielfalt)
- Realitätsnahe Simulation
- Mechwarrior
- Hoverboat
- Irgendetwas mit Windkraft (Sponsor: Hersteller von Windkraftwerken)
- Viel Bewegung und ohne Schusswaffen
- Planetenwanderung über Stahlseilaufhängungen
- Sportcave für Kampfsport
- Pod-Racer
- Geh-Cave
- Raumfahrtcave
- Virtuelle Datenhaltung
- Adventure Simulation

- Anatomiecave (medizinisch)
- Kunst und Installationen

Nach diesem Brainstorming wurden einige Themenvorschläge für den Cave aufgegriffen und besprochen. Beim Pod-Racer wurde z.B. kritisiert, dass wenig Interaktion und eine Unflexibilität vorzustellen sei. Eine Rennsimulation gebe daher wenig her. Weiter wurde diskutiert wieviele Projektionsflächen benutzt werden sollten. Die Bewegung bei einer Rennsimulation sei lediglich gradlinig. Mehr Interaktion sei z.B. durch Fußmatten erreichbar. Nicht zu übersehen sollte man, dass die Zielfindung sich nicht nur auf den Cave selbst beziehen sollte, sondern auch auf die Anwendung und die anderen Komponenten. Es wurde der Vorschlag gemacht, ein grobes Ziel zu finden, um dann in Kleingruppen daran weiterarbeiten zu können. Es war zu bedenken, dass eine minimale Anwendung sicherlich 1 ½ Jahre zur Programmierung Zeit brauche. Um in dieser Zielfindung nun konkreter weiterzukommen, sollte sich jeder Teilnehmer mit einer Idee zum Cave auseinandersetzen. Dabei sollte dieser eine Skizze seiner Idee mit allen Komponenten anfertigen, mögliche Funktionen daran ermitteln und mögliche Ein- und Ausgabemöglichkeiten angeben, wie z.B. wie man sich in diesem Cave bewegen solle. Das Thema sollte genau beschrieben werden und es sollte gezeigt werden, wie die einzelnen Komponenten umgesetzt werden und welche Geräte was umsetzen könnten.

9.4.1 Präsentation

Aus diesem ersten Brainstorming ergaben sich einige detalliertere Vorstellungen, die inhaltlich im Plenum wie folgt präsentiert wurden:

9.4.2 Zielfindung/-diskussion

Schließlich wurde in einem gemeinsamen Abstimmungsprozess über das Thema des Caves entschieden. Abgestimmt wurde mit je zwei Stimmen pro Teilnehmer in einer anonymen Wahl. Dabei betrug die erste Stimme 3 Punkte und die zweite Stimme 1,5 Punkte. Zur Abstimmung standen letztlich folgende Ideen:

- PiCa (Picasso Cave)
- UE (Universumserkundung)
- MMW (MyMechWarrior)
- FT (Fliegender Teppich)
- Adv. (Adventure Cave)
- UC (Unterwasser Cave)
- RC (Radsport Cave)
- GS (gewaltfreies Spiel)

Die Auszählung ergab folgende Verteilung der Stimmen:

- PiCa (Picasso Cave): 9
- UE (Universumserkundung): 5,5
- MMW (MyMechWarrior): 4,5

- FT (Fliegender Teppich): 22
- Adv. (Adventure Cave): 11,5
- UC (Unterwasser Cave): 4,5
- RC (Radsport Cave): 11,5
- GS (gewaltfreies Spiel): 0

Eine Diskussion zu den Themen FT, RC und Adv. Wurde gestartet, nachdem diese die meisten Stimmen erhielten. Daher wurde ein zweiter Abstimmungsprozess zwischen den drei Themen gestartet. Die Ergebnisse verteilten sich wie folgt:

- FT (Fliegender Teppich): 7
- Adv. (Adventure Cave): 2
- RC (Radsport Cave): 3

Damit war die Entscheidung gefallen, dass das Ziel des Projektes die Entwicklung einer Anwendung mit dem Thema F̈liegender Teppichẅar.

9.5 Instanziierung der Teilgruppen

Im ersten Semester hatten sich folgende Gruppenkonstellationen gefunden:

- Anforderungsdefinition
  - Christian, Christopher, Patrick R.
- Plattform
  - Markus, Elmar, Franck
- Sound
  - Benjamin, Manfred, Gunnar, Tina
- Cave
  - Patrick B., Gunnar, Tina
- Interface/Sensorik
  - Tobias, Alexander, Tatiana, Björn

9.6 Wissensaneignung

Die Wissensaneignung geschah sowohl in Eigeninitiative, als auch durch die Teilnahme an Vorlesungen und Kursen. Folgende Vorträge und Referate wurden präsentiert:

Cave

Allgemeines

Konstruktion

Beispiele

Projektion

Beamer

Auflösung, Linsen, Abstände

Graphik (Christian)

Open GL

Wie funktioniert eine 3D-Engine

Sensorik

Sensoren in m agnetischen Feldern

Optische Sensoren

Tracking Systeme

Aktorik (Franck, Tobi)

Interaktion in virtuellen Umgebungen

Menüführung im 3D-Raum

3D-Mäuse und Eingabegeräte

Sound (Gunnar, Benjamin)

Surround Sound

Sound in bewegten 3D-Szenen

Pneumatik (Tobi, Franck)

Einführung in LaTeX

Projektmanagement Ergebnisse vom Projektmanagement-Seminar: Tatiana und Tina trugen die Ergebnisse des Projektmanagement-Seminars im Freitags-Plenum vor. Zuerst wurde der Begriff “Projekt“ als eine Aufgabe definiert, die Teamarbeit, ein Ziel, ein bestimmtes Budget (monetär wie zeitlich) umfasst, und sich dabei durch eine Einmaligkeit und Einzigartikgeit (das Projekt wird nur einmal ausgeführt und war in dieser Form noch nicht da) sowie eine gewisse Unsicherheit (es ist noch nicht klar, wie vorgegangen wird) auszeichnet. Bei den Zielen eines Projekts sollte insbesondere unterschieden werden zwischen:

Sachziel Was soll geplant und erreicht werden ?

Kostenziel Was darf das Projekt kosten ?

Terminziel Bis wann soll alles erreicht werden ?

Diese Ziele standen in Abhängigkeit voneinander, d.h. eine Änderung an einem wird auch die anderen beeinflussen.

Als nächstes wurden Karteikarten an die Teilnehmer ausgegeben, die dort notieren sollten, was ihre Vorstellungen vom Projektablauf in den nächsten 2 Jahren seien. Die (anonymen) Karten wurden danach vorgelesen. Insbesondere schien es zu grossen Teilen einen Konsens über das Projektziel zu geben. Ein Projekt lässt sich in folgende Phasen einteilen :

1. Planung
2. Arbeitsteilung
3. Ausführung
4. Fortschrittskontrolle, evtl. Korrektur des Plans
5. Abschlussbericht

Insbesondere sollte in der Projektplanung folgendes erledigt werden:

1. Situationsanalyse (Was ist los ?)
2. Zielsetzung (Was soll erreicht werden ?)
3. Konzeptentwurf (Welche Lösungen sind möglich ?)
4. Bewertung (Welche Lösungen sind sinnvoll ?)
5. Entscheidung (Wie ist die Lösung zu realisieren ?)

Weiterhin erläuterten die Referentinnen noch anhand eines Diagrammes den üblichen Verlauf der Motivation sowie Effizienz in einem Projekt: Am Anfang sei die Effizienz niedrig, da noch nicht klar ist, was gemacht wird, aber die Motivation sei hoch. Danach folge eine Phase der Enttäuschung, in der die Motivation stark sinke, aber die Effizienz steige. Nach Überwindung der Enttäuschungsphase folge die Lösungsphase, in der die Motivation wieder steige und auch die Effizienz weiter wachse. Das Projekt schliesse dann mit der Endphase, in der beides nur noch schwach wachse.

Eine Diskussion führte zu dem Schluss, daß dieses Diagramm nur ein Idealbild sei und nicht unbedingt auch auf das Projekt zutreffe, insbesondere wurde empfunden, daß es nicht wirklich eine Enttäuschungsphase gegeben habe.

Projektion/Spiegelung Patrick R. und Björn hielten ein Referat über die Möglichkeiten von Spiegelprojektionen. Insbesondere wurde dargelegt, das ohne Spiegel eine extrem grosse Entfernung von der Projektionsfläche benötigt werden würde, um ein grosses Bild zu projizieren. In einer Diskussion darüber wurde angeregt, später noch einen Prototyp-Spiegel zu bauen, um die Eigenschaften und Probleme zu testen. Ausserdem sollten die verfügbaren Beamer durchgetestet werden, um sich auf die Unterschiede einstellen zu können.

9.6.1 Vorlesungen

GDV, Projektmanagement

9.7 3-Stufen-Plan

Um die lange Laufzeit des Projektes übersichtlicher zu machen, und um Fortschritte und Misserfolge leichter einschätzen zu können, wurden die zwei Jahre in vier Teile unterteilt. Diese endeten mit dem Erreichen eines Meilensteins, bzw. mit der Abschluss-Präsentation.

Der erste Abschnitt endete am 01.10.2003 mit dem ersten Meilenstein. Unsere Planung sah vor, dass es bis zu diesem Zeitpunkt möglich sein sollte, in der Welt über eine einfache Hügellandschaft zu fliegen, ohne dass weitere Effekte Anwendung fanden. Akustisch untermalt werden sollte die Szenerie nur von Hintergrundmusik und einfachen sich nicht ändernden Nebengeräuschen, wie dem Flugwind. Die Plattform sollte mit Sensoren ausgestattet sein, und ein einfaches Manövrieren in der Welt ermöglichen; Force Feedback-Effekte waren hier noch nicht vorgesehen. Die Konstruktion des CAVE sollte sich der Fertigstellung nähern. Die Engine sollte in der Lage sein, Rechner anzusprechen; die Maps sollten geladen werden können. Es sollte bis zu diesem Zeitpunkt eine statische Welt stehen, die weder dynamische Objekte beinhalten sollte, noch dass beispielsweise eine Kollisionserkennung implementiert worden sei.

Am 14.02.2004 sahen die Planungen das Erreichen des zweiten Meilensteins vor. Hier sollte die Force-Feedback-Einbindung sowie die Implementierung einer Kollisionserkennung stattgefunden haben. Die Engine sollte nun netzwerkfähig sein, und über ein Eventsystem verfügen. Die Ereignisse sollten entsprechende Sounds auslösen; d.h. es sollte die Möglichkeit dynamischer Soundquellen geben. Ebenso sollte nun die physikalisch korrekte Simulation des Windes realisiert worden sein. Weiter sollten die Arbeiten am Eingabesystem fertiggestellt worden sein.

Der dritte Meilenstein sollte am 01.05.2004 erreicht werden. Die gesteckten Ziele waren nun eher inhaltlicher Natur; so sollte nun ein gewisses Maß an Spielelogik stehen. Dabei sollte ein Geschicklichkeitsparcours fertig sein, den der Spieler zu absolvieren hatte; favorisiert wurde hier die Idee, mit dem fliegenden Teppich Münzen einzusammeln. Die Objekte, die sich in der Welt befanden, sollten sich nun mittels eines Animationssystems mehr oder wenige realistisch bewegen. Zu diesem Zeitpunkt war auch die Fertigstellung der Soundbibliothek vorgesehen, die in enger Zusammenarbeit mit der Modellierung geschehen sollte, um für die Objekte stimmige und sinnvolle Sounds zu finden. Die Sounds sollten nun im Raum positionierbar sein, um dem Anspruch des Surround-Sounds gerecht werden zu können. Die Sammlung von Sounds für die Bibliothek, sowie die Modellierung von Objekten geschahen parallel zu den anderen anfallenden Arbeiten kontinuierlich über den Zeitraum der zwei Jahre.

Die letzte Phase sollte nun weitestgehend letzten Änderungen, und vor allem Tests vor der Präsentation Raum und Zeit geben. Ebenfalls sollten, sofern es noch Kapazitäten geben sollte, einige zusätzliche Kann-Anforderungen, wie das Einbinden von Partikeleffekten, Tag/Nacht-Zeiten, computergesteuerte Gegner für den Spieler, hinzugefügt werden.

9.8 Autarkes Arbeiten

Aufgrund der hohen Komplexität, welche die Arbeit an einer Mixed RealityUmgebung auszeichnet, wurde schon früh beschlossen, das Projekt in kleinere Teilgruppen zu untergliedern.

Hierzu wurden im Plenum zunächst Überlegungen angestellt, welche Gebiete Bearbeitung verlangten, und welche eher außer Acht gelassen, bzw. nebenher bearbeitet werden konnten. Nach einigen Überlegungen einigten wir uns auf die folgenden Experten gruppen:

- Grafik Engine
- Grafik Modellierung
- Sound
- Plattform
- CAVE
- Public Relations

- Webseite
- Finanzen
- Hardware
- Interface

Die konkreten Arbeitsinhalte entnehme man dem jeweiligen Kapitel in diesem Bericht.

Obwohl einige der Gruppen, wie z.B. die für die Webseite oder die für die Finanzen zuständige Gruppen, nicht direkt mit dem Projektziel verknüpft waren, erschien es uns dennoch wichtig, die Organisation und das Auftreten unseres Projektes nach außen hin durch entsprechende Gruppenbildung zu organisieren. Da die Mitglieder dieser Gruppen ebenfalls in Gruppen mitarbeiteten, die eher dem Kernbereich zugezählt werden können, blieb jedes Mitglied des Projektes stets mit dem Ziel und den Arbeiten daran in engem Kontakt.

Ein Vorteil der Gruppenbildung war, dass die einzelnen Teilbereiche des Projektes unabhängig von den anderen bearbeitet werden konnten. Der Fortschritt der Gruppen wurde durch den zuvor beschriebenen DreiStufen Plan bestimmt. Neben dieser allgemeinen Version erstellte jede Untergruppe einen eigenen DreiStufenPlan mit den eigenen Zielen. Das Erreichen der Ziele, bzw. die Kontrolle des Fortschritts der Teilgruppen wurde durch die Berichte, die von den Teilgruppen regelmäßig in den Plena gegeben wurden, sichergestellt.

9.9 Zwischenpräsentation

Am 24. Oktober 2003 fand die projektübergreifende Zwischenpräsentation der drei MiCa-Projekte statt, auf der der jeweils aktuelle Stand der Projekte gezeigt wurde.

Die Zwischenpräsentation hatte für uns neben dem Zweck der Präsentation nach außen auch noch die Aufgabe, zum ersten Mal das Gesamtsystem komplett aufgebaut zu haben und somit das Zusammenspiel aller Komponenten zu testen. Daher hatte jeder der Teilgruppen den aktuellen Stand ihrer Komponente für die Präsentation vorbereitet sowie insgesamt eine Folien-Präsentation des Gesamtprojektes.

Zu dem Termin war die Konstruktion des eigentlichen CAVE-Rahmens schon abgeschlossen, die Befestigung und das Material der Projektsflächen war aber noch von dem ursprünglichen Prototypen übernommen. Zur Demonstration der Interaktion war die Plattform schon vorhanden und konnte benutzt werden, besaß allerdings zu diesem Zeitpunkt noch keine Aktoren, sondern nur Sensoren zur Steuerung (genaueres im entsprechenden Kapitel). Auf diese Weise konnten Interessierte schon einmal die Steuerung testen. Ebenfalls war ein Prototyp der Geschwindigkeitsregelung über Schalter eingebaut.

Auf der Softwareseite war die Engine in der Lage, mit dem eigenen Exporter erstellte Szenen zu laden und nach den Vorgaben darzustellen. Die Eingaben über das SIOS-Interface konnten zur Steuerung der Engine verwendet werden, außerdem wurden Sounds als Hintergrundmusik abgespielt.

9.10 Projekttag

Am Freitag den 9. Juli 2004 fand der Projekttag des Studiengangs Informatik an der Universität Bremen in der ersten Ebene des MZH statt.

Ziel des Projekttages ist es die Entwicklungen der Projekte, welche ihre Arbeit im aktuellen Semester nach zwei Jahren abschliessen, zu zeigen. Der Projekttag ist ein öffentliches Ereignis, zu welchem alle Interessierten Personen, auch solche, welche nicht an der Universität Bremen arbeiten, eingeladen sind.

Abbildung 9.1: Poster Motionplattform

Im Rahmen dieses Ereignisses stellt jedes Projekt einen Präsentationsstand und präsentiert sich für maximal 30 Minuten auf einer Bühne. Die Präsentationen sollen zwar informieren; wichtig ist aber auch, dass die Zuschauer unterhalten werden.

Der Projekttag begann dieses Jahr um 9.45 Uhr und endete um 17.00 Uhr.

9.10.1 Vorbereitungen

Die Vorbereitungen für den Projekttag fingen Ende April, etwa 2 Monate vor dem 9. Juli an.

Anforderungen

Die erste Aufgabe war einen für unsere Ansprüche passenden Raum zu finden, da das Projekt eine zusätzliche externe Präsentation des CAVEs wollte.

Hierzu fand sich nach kurzem Suchen das Theater der Universität. Es erfüllte alle unsere Ansprüche; eine genügend grosse Fläche und Höhe, ein abgedunkelter Raum, eine Netzwerkverbindung und viele Zuschauerplätze. Um das Erfüllen dieser Anforderungen zu Testen wurden mehrfach Proben zur Durchführbarkeit gemacht.

An diesem Ort sollte am Projekttag gegen Mittag, eine zusätzliche Präsentation stattfinden.

Als nächstes musste die Einrichtung des Standes geplant werden. Wir entschieden uns drei Plakate mit rudimentären Informationen zur Cave, Plattform, Grafik, Interface und Gesamtübersicht zu erstellen.

Zusätzlich sollten zwei Tische mit Flyern und Monitoren aufgestellt werden.

Auf den Monitoren sollte eine projektbeschreibende Präsentation abgespielt werden; die Flyer sollten einen kleinen Einblick in die Thematik geben und den Weg zum Theater aufzeigen.

Abbildung 9.2: Poster: Grafik / CAVE

Das dritte zu planende Element der Präsentation waren die 30 Minuten auf der Bühne. Wir entschieden die von uns geschriebene Spielgeschichte (Verweis Spielgeschichte) anhand eines Theaterstücks (Verweis Bühnenstück, Bühnenaufbau) zu erzählen und die Thematik des Projekts mittels zweier Filmsequenzen zu erläutern. Als erstes sollte ein Film die Personen und die Cave vorstellen (Verweis Film 1). Danach sollte das Theaterstück Anwendung der CAVE erklären und der zweite Film sollte anhand der CAVE erklären, was der Begriff Mixed Reality bedeuted und, welche Anwendung diese Idee bei der der Cave gefunden hat (Verweis Film 2).

Das letzte Element der Bühnenpräsentation sollte eine Live-Videoschaltung zur CAVE im Theater werden. Das Publikum sollte die Person, welche den CAVE bedient dazu anleiten, die Spielaufgabe zu erfüllen. Dazu würde eine zusätzliche Audioschaltung zum CAVE benötigt werden.

Folgende Materialen wurden (zusätzlich zum Material der CAVE) zur Installation benötigt:

- Bühnenmaterial für das Theaterstück
- 4 Netzwerkkabel (Theater, 2x Stand, Bühne)
- 4 Rechner (2x Stand, Theater, Bühne)
- 2 Kameras (Theater, Bühne)
- 1 Mikrophon (Bühne)
- 1 Beamer (Bühne)
- 2 Tische (Stand)
- 4 Stellwände (Stand)

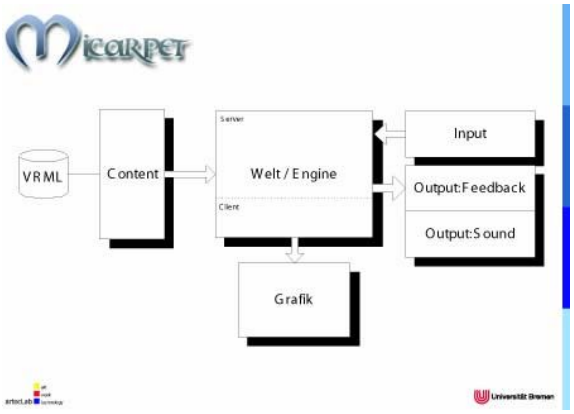


Abbildung 9.3: Poster: Übersicht System

Abbildung 9.4: Handzettel Innenseite

Zeitplan

Als nun alle feststand, welche Orte wie ausgerüstet sein werden wurde ein Personenplan erstellt, welcher die Projekt-Teilnehmer in Schichten zu je 45 Minuten einteilte (Verweis Personenplan). Hierbei wurde darauf geachtet, dass zu jeder Zeit eine Person aus den Gruppen Grafikengine, Motionplattform und Interface im Theater eingeteilt war, um so die Möglichkeit einer schnellen Fehlerbehebung zu ermöglichen. Ausserdem sollte der Stand im MZH dauerhaft mit zwei Personen besetzt sein, um Fragen zu beantworten und ein wachsames Auge auf die Geräte zu haben.

Aufbau

Das Bühnenmaterial für das Theaterstück wurde zum grössten Teil von den Mitgliedern des Projektes gestellt. Die Kostüme wurden von einem Kostümverleih geliehen. Die nächsten drei Punkte der Liste wurden von den Mitgliedern des Projektes gestellt. Die anderen Materialen stellte uns die Universität zur Verfügung. Zusätzlich wurde noch Material zur Dekoration der Theaterbühne bereitgestellt. Es sollte so eine orientalische Atmosphäre entstehen. Ausserdem wurden Polo-Shirts mit dem Micarpetlogo bestellt, um ein einheitliches Auftreten und ein Zugehörigkeitsbestimmung für Andere zu ermöglichen. Des weiteren mussten die Plakate, sowie die Flyer hergestellt werden. Bei dem Druck der Plakate stand uns das BIBA sehr hilfreich zur Seite - vielen Dank - und die Flyer wurden mit den Druckern des Fachbereich 3 in der Ebene 0 hergestellt.

Abbildung 9.5: Handzettel Aussenseite

Um Personen das Auffinden des Theater zu erleichtern stellten wir Wegweiser mit dem Namen des Projektes und der Richtung zum Theater auf.

Mit dem Aufbau der Präsentationsinstallationen wurde am Morgen des Präsentationstages begonnen, da man das Diebstahlrisiko minimieren wollte. Bis zum Beginn der Präsentationen war alles aufgebaut, funktionsfähig und die Flyer und Plakate ausgedruckt und zugeschnitten.

9.10.2 Der Projekttag

Der grobe Zeitplan für den kommenden Tag sah wie folgt aus: Um 10.00 Uhr wurde die erste Ebene des MZH für interessierte Gäste eröffnet. Gleichzeitig wurde auch im Theater Einlass gewährt.

Um 13.00 Uhr sollte die erste kommentierte Bildpräsentation im Theater stattfinden. Um 15.00 Uhr sollte die Bühnenpräsentation im MZH stattfinden.

Gegen 18.00 Uhr sollte der Tag ausklingen und die Installationen abgebaut werden.

Durch die gute Planung und die mehrfachen Tests des Gesamtsystem gab es im Laufe des gesamten Tages keine grossen Pannen und Ausfälle.

Eine zu erwartende Problematik war das fehlende Gewicht zur Steuerung der Plattform bei den weiblichen Gästen. Diese wurde aber oft durch zusätzliche Kreativität behoben, als etwa die Plattform zum Tandem erweitert wurde und von zwei Personen gleichzeitig gesteuert wurde.

Da es als sehr misslich empfunden wurde, dass die Personen am Projektstand keinen Einblick in die Anwendung des Systems bekamen, wurde eine Live-Videoschaltung zum Theater her- und auf dem einem Monitor am Stand dargestellt. Dies bedeutete keinen Verlust an Präsentationsmaterial, da beide Monitore des Standes die selbe Präsentation zeigten. Um 13.00 Uhr fand wie geplant die Präsentation im Theater statt. Diese Präsentation sollte direkt am Objekt, also an dem CAVE, einen Einblick in die Technik und Konstruktion geben (Verweis Präsentation Theater).

Zur Darstellung des Präsentationsmaterials benutzten wir die Vorderseite des CAVEs. Natürlich war der CAVE während der Präsentation nicht im normalem Betrieb. Wie alles an diesem Tag, lief auch diese Präsentation reibungslos.

Einen grösseren Ausfall gab es am Nachmittag, da ein Besucher von der Plattform fiel , sich glücklicherweise aber nicht verletzte. Leider landete er bei seinem Sturz direkt in der Steuerungstechnik. Der so angefallene Schaden beinhaltete allerdings nur die Trennung von Leitungen und nicht das Zerstören von Hardware; nach kurzer Reperatur konnte das Programm fortgesetzt werden.

Um 15.00 Uhr startete unsere Bühnenpräsentation im MZH. Auch diese lief bis zu dem Starten der Live Video-Audio-Schaltung ohne Probleme. Zwar klappte die Videoschaltung ins MZH, aber leider gab es von dort keine Bestätigung, dass sie den Ton aus dem MZH empfangen hätten. Nach kurzem Versuchen brachen ab. Trotzdem war die Präsentation ein voller Erfolg. Sie war mit dem gewollten Witz bei den Zuschauern angekommen und hatte die Thematik des Projektes verständlich erklärt.

Um 18.00 Uhr endete der Projekttag und wurde von allen Mitgliedern als ein voller Erfolg empfunden.

9.10.3 Video

Die Intention einen Projektfilm zu erstellen, lag darin, den Zuschauern am Projekttag neben dem Theaterstück, welches die Spielidee wiedergeben sollte, auch einen Überblick und eine Zusammenfassung über die Arbeit der letzten zwei Jahre im Projekt zu geben.

Das Video wurde darauf in zwei Teile geteilt. Der erste Teil sollte recht kurz gehalten werden und die einzelnen Projektteilnehmer vorstellen. Die Werdensgeschichte des Projekts wie Software und Hardware sollten hier chronologisch gezeigt werden. Erste Einblicke in das Projekt sollten gewährt werden und einen Einstieg in die Spielidee erlauben. Diese sollte dann im Anschluss an den ersten Teil des Videos in Form eines Theaterstücks dargestellt werden.

Im zweiten Teil des Films wurden dann noch einmal explizit die fertigen Teilkomponenten, wie Cave, als Akronym für “Cave automatic virtual environment“, der Ventilator, zur Simulation des Fahrtwindes in Abhängigkeit der Geschwindigkeit, das Schnellspannsystem zum schnellen werkzeuglosen Aufbauen des Caves, der Teppich als Plattform für die Flugsimulation, der Windeinlass zur Simulation des Fahrt-

windes und die Beamer zur Projektion der Flächen, gezeigt. Dann wurde der Übergang von “Real Reality“, was den Benutzer des Caves zeigte, “Mixed Reality“, was den Benutzer des Caves auf dem Teppich sitzend, die Peripherie benutzend und die Welt sehend, darstellte und schließlich “Virtual Reality“, was die Welt präsentierte.

9.11 Medienliste

Folgende Videos wurden erstellt und befinden sich auf der Projekt-DVD:

Videos		
Projekttag: Intro	DVD-Video	5:38min.
Projekttag: Theaterstück	DVD-Video	7:33min.
Projekttag: Abspann	DVD-Video	2:53min.
Plattformvideo 1	DVD-Video	0:46min.
Plattformvideo 2	`/videos/Plattformvideo_2.avi`	0:11min.
Plattformvideo 3	`/videos/Plattformvideo_3.avi`	0:15min.
CAVE im Betrieb	`/videos/micarpet.wmv`	2:44min.

Ebenfalls enthalten ist der Projektbericht als PDF (im Grundverzeichnis) und sämtlichem Bildmaterial bzw. allen Zeichnungen, die im Projektbericht verwendet wurden (in `/grafik/`).

Teil IV

Übersicht des Gesamtsystems

Kapitel 10

Das MiCarpet-System

10.1 Übersicht

Als Einleitung zur Besprechung der Einzelkomponenten im nächsten Teil soll hier eine Übersicht über das gesamte entstandene **[Mixed-Reality-System??]** gegeben werden. Dabei konzentrieren wir uns in diesem Teil vor allem auf die grobe Struktur des Systems, sowie die Verbindung der Komponenten untereinander, ohne auf die jeweiligen Details der Implementierung einzugehen.

Grob gesehen kann man die Komponenten des Systems in Hardware und Software einteilen, mit einer Verbindung der beiden über ein oder mehrere Interfaces. Eine Übersicht der Komponenten ist in Abbildung 10.1 zu sehen:

- die **Engine** ist der zentrale Softwarekern des Systems. Sie übernimmt die Initialisierung und die Steuerung der meisten anderen Softwarekomponenten und verwaltet die Kommunikation unter den Komponenten. Die Engine ist in Abschnitt 12 ab Seite 85 im Detail beschrieben.
- die **Grafik** ist sehr eng verbunden mit der Engine, sie stellt den visuellen (oder rein virtuellen) Teil des Systems dar. Die Grafik-Engine erlaubt die Darstellung der Spielszene auf mehreren parallel arbeitenden Rechnern (siehe 12.4, S.104ff), so dass mit Hilfe mehrerer Projektoren alle Wände der **CAVE** gleichzeitig angestrahlt werden können. Die Grafik wird im Abschnitt 12.1 (S.85ff) näher beschrieben, die CAVE in 11 (S.77ff). Weiterhin wird in Abschnitt 13 ab Seite 113 die Gestaltung der Szenen und Grafikdaten des Systems dargestellt.
- die **Sound**-Komponente ist dafür zuständig, auf entsprechenden Befehl der Engine Soundeffekte und Musik abzuspielen (auch mit entsprechender 3D-Positionierung) und steuert natürlich auf Hardwareseite die 4 **Lautsprecher** an. Genaueres hierzu gibt es in Abschnitt 15 ab Seite 197.
- die **Physik** ist ebenfalls eng mit der Engine gekoppelt, sie simuliert das Verhalten der Objekte in der Szene sowie des Spielers selbst. Daher ändert sie zum einen durch Animation von Objekten die Grafik selbst, zum anderen kann sie aber auch physische Aktionen auslösen (z.B. Feedback bei Kollision). Näheres in Abschnitt 12.2 (S.98ff).
- die **Aktorik** steuert über die „SIOS“ (siehe 14) die **Aktoren** des Systems, sie stellt also eine Schnittstelle zwischen Hardware und Software her. So kann sie z.B. ein physisches Feedback durch Ansteuerung der Motion-Platform oder der Ventilatoren verursachen. Eng damit verbunden ist die **Input**-Komponente, die ebenfalls das SIOS-Interface nutzt, aber Daten von **Sensoren** (z.B. von der Plattform) einliest und diese in Steuerkommandos an die Engine weitergibt. Eine genaue Beschreibung von beiden ist in Kapitel 14 ab Seite 129 zu finden.

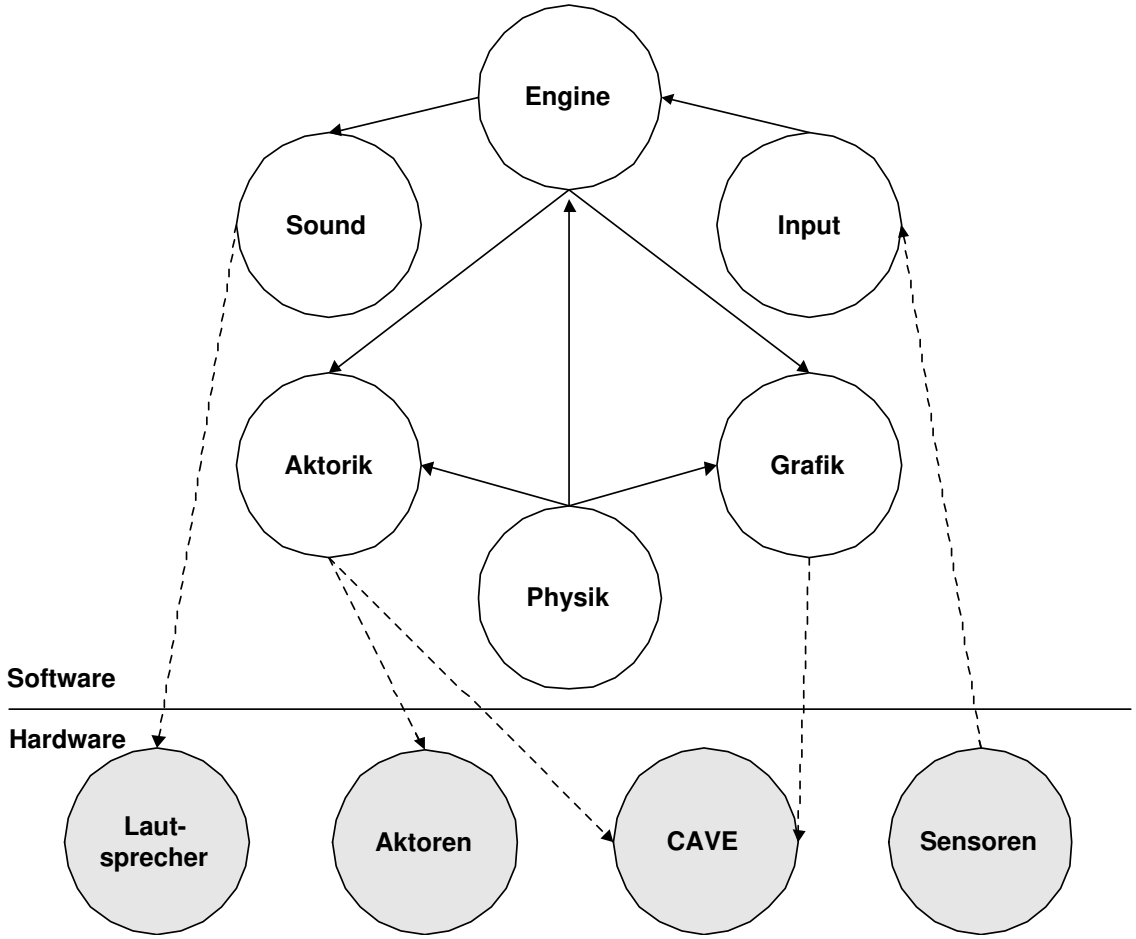


Abbildung 10.1: Komponenten des Systems und der ungefähre Daten- und Kontrollfluss im System

10.2 Hardware

Auf der Hardwareseite besteht das MiCarpet-System (siehe Abbildung 10.2) aus drei PCs, welche über einen **[ethernet??]**-Switch zu einem Netzwerk verbunden sind, sowie angeschlossenen Peripherie-Geräten, Sensoren und Aktoren und natürlich der eigentlichen CAVE.

Die drei PCs sind aufgeteilt in einen zentralen *Master*-PC, der zum einen alle Sensoren und Aktoren kontrolliert und zum anderen auch die Zentrale im Netzwerksystem (siehe 12.4 ab Seite 104) ist. Die restlichen Rechner sind nur *Slaves*, welche die Aufgabe haben, mittels des angeschlossenen Projektors jeweils eine Projektionsfläche der CAVE zu bestrahlen. Die drei PCs sind aufgeteilt in einen zentralen *Master*-PC, der zum einen alle Sensoren und Aktoren kontrolliert und zum anderen auch die Zentrale im Netzwerksystem (siehe 12.4 ab Seite 104) ist. Die restlichen Rechner sind nur *Slaves*, die die Aufgabe haben, mittels des angeschlossenen Projektors jeweils eine Projektionsfläche der CAVE zu bestrahlen.

Die Peripherie am Master sind vier Lautsprecher für das Soundsystem (genauere Besprechung im Kapitel 15, S.197ff), Standard-Eingabegeräte wie Maus und Tastatur und schließlich, als Schnittstellen zur Hardware, zwei SIOS-Interfaces (siehe Abschnitt 14 ab Seite 129). Diese erlauben das Ansteuern der Hardware vom Master aus, als auch das Auslesen der Sensorwerte. Angeschlossen an die SIOS sind die Motion-Platform und das Ventilator-System (Details ab Seite 129, Abschnitt 14).

Der letzte Teil ist schließlich die CAVE selbst, die die Umgebung für die Projektion darstellt und mit der ein Großteil der Sensoren und Aktoren verbunden sind. Die Konstruktion und Genaueres zum Aufbau

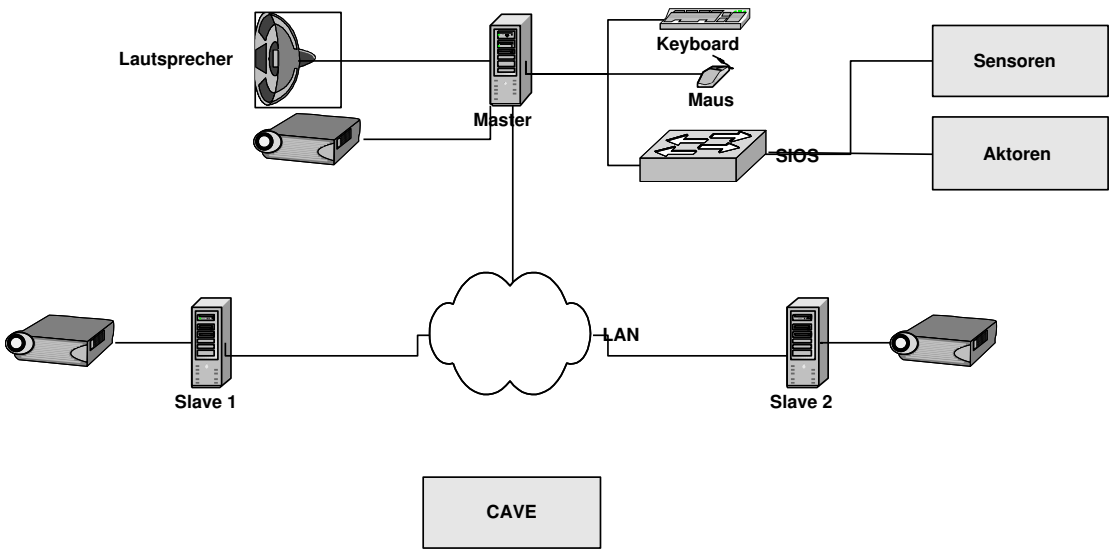


Abbildung 10.2: Hardware-Komponenten des Systems und physische Verbindung

werden im Kapitel 11 ab Seite 77 beschrieben.

10.3 Software

10.3.1 Übersicht

Der Software-Teil des MiCarpet-Systems ist komplett **[objekt-orientiert??]** in **[C++??]** geschrieben und lässt sich konzeptuell in mehrere Komponenten aufteilen: die Kern-**Engine** (Abschnitt 12) inklusive **Event-System** (12.3), **Scripting** (12.5) und **Physik** (12.2), die **Grafik** (12.1) mitsamt **Netzwerk-System** (12.4), das **SoundSystem** in 15 und das **Input-/Outputsystem** als Schnittstelle zu Sensoren und Aktoren in Kapitel 14.

Die Kommunikation unter den einzelnen Komponenten geschieht dabei größtenteils über das Event-System, welches eine sehr flexible Struktur des Systems erlaubt. Dies hat insbesondere auch den Vorteil, dass gerade das SoundSystem und das Input-/Output-System größenteils getrennt vom restlichen Teil der Engine entwickelt werden konnten und so die Gruppen mehr Freiheit in der Programmierung ihrer Softwarekomponenten hatten.

Weiterhin sind die Komponenten so konzipiert, dass die Engine die Komponenten auch nur selektiv laden kann. So wird etwa auf den reinen Slaves nicht das Input-/Outputsystem, das Scripting und der Sound geladen, da die entsprechenden Geräte gar nicht angeschlossen sind bzw. die Kontrolle der Szene komplett vom Master ausgeht.

10.3.2 Klassenstruktur

Eine vereinfachte Darstellung des Systems als UML-Klassendiagramm ist in Abbildung 10.3 auf Seite 72 zu sehen. Die Hauptklasse, die als erstes instanziiert wird, ist `CGame`. Sie initialisiert das Netzwerk (als Master/`CNetMaster` oder Slave/`CNetSlave`) und startet die eigentliche Engine (`CEngine`). Diese kümmert sich um mehrere Dinge: sie startet das Grafiksystem (`CRender`), lädt die Eingabe-/Ausgabe-plugins (`CExtModuleManager`) und startet dann schließlich die gewünschte Szene (`CLevel`).

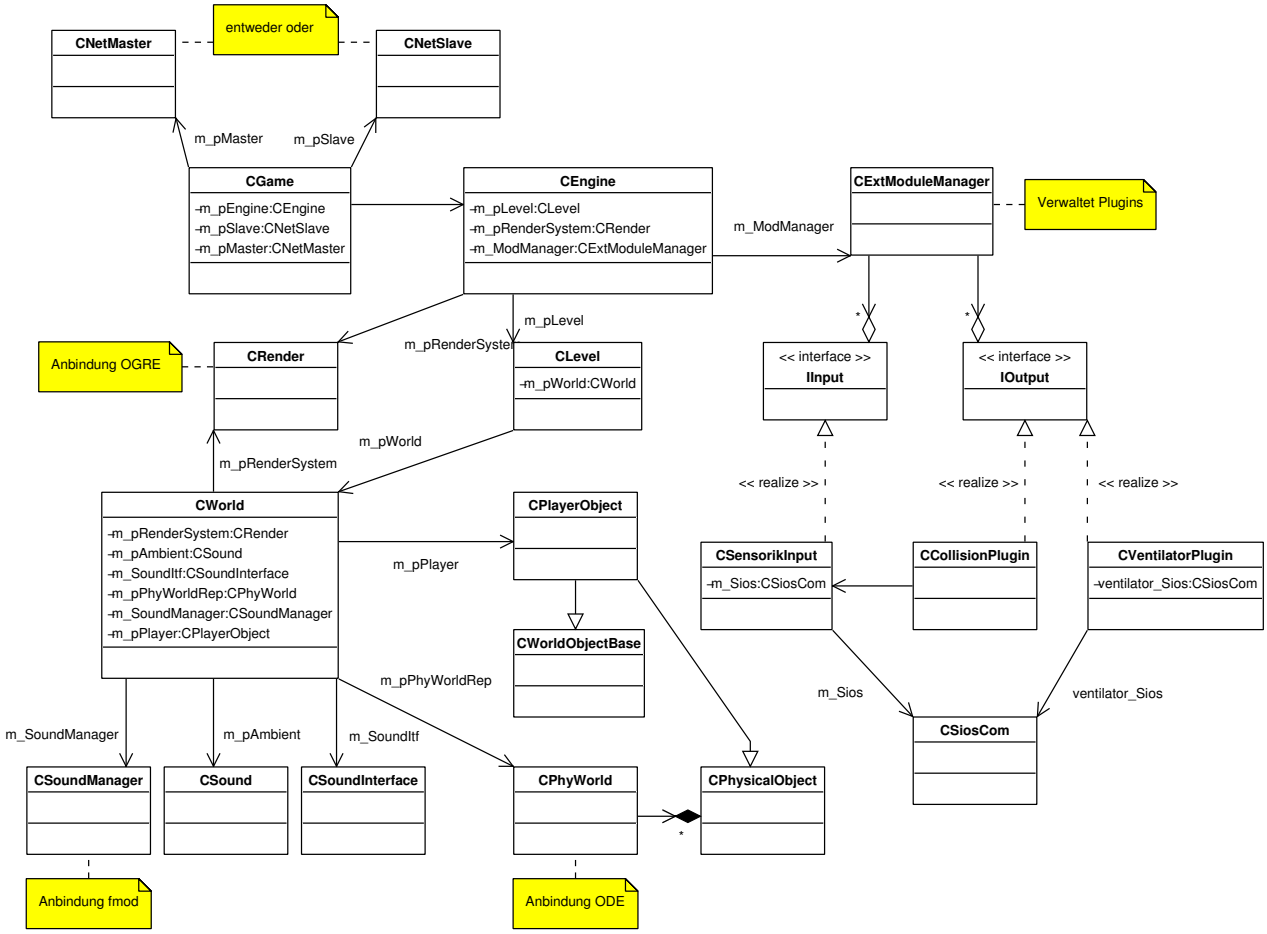


Abbildung 10.3: UML-Klassendiagramm des Systems. Aus Übersichtsgründen werden nicht alle Klassen und Relationen angezeigt.

`CExtModuleManager` ist die Verwaltungsstelle für alle Eingabe- und Ausgabe-**[PlugIn??]**s (etwa Ansteuerung der Plattform), die jeweils von den Interfaces `IInput` und `IOutput` abgeleitet sind. Entsprechend der Einstellungen werden dann die Klassen instanziiert und liefern über Events Daten zurück an die Engine.

`CLevel` steht für eine laufende Instanz einer Szenendatei (`.mcs`) und startet dafür `CWorld`, welche die interne Repräsentation der Szene verwaltet (siehe Abschnitt 12.1.3). Sobald die Szene gewechselt wird, wird die alte Instanz von `Clevel` zerstört und eine neue erstellt. `CWorld` hat insbesondere Anbindungen an das Soundsystem (`CSound`, `CSoundManager`, `CSoundInterface`), die Physiksimulation der Welt (`CPhyWorld`) und das Scripting.

Genauere Erläuterungen zu den meisten dieser Klassen finden sich in den entsprechenden Abschnitten.

10.3.3 Verwendete externe Software-Bibliotheken und Komponenten

Hier soll ein kurzer Überblick über die externen Softwarekomponenten, die unser System benutzt, gegeben werden. Teilweise werden die Komponenten an späterer Stelle noch genauer beschrieben, dann findet sich nur eine sehr kurze Beschreibung mit einem Verweis auf die entsprechende Textstelle.

Boost

Boost[**?**] ist eine freie Open-Source C++ Bibliothek, die sich als Ergänzung zur **[STL??]** von **[C++??]** versteht. Boost stellt dabei eine Vielzahl von Funktionen für einen breiten Bereich von Anwendungen bereit.

Im System werden einige Teile von Boost benutzt, vor allem in der Scripting-Engine (siehe 12.5, S.108).

fmod

Die Bibliothek *fmod*[**?**] ist eine frei verfügbare Bibliothek zur Ausgabe von Soundeffekten und Musik, die auf mehreren Plattformen verfügbar ist. Einzelheiten finden sich in Abschnitt 15 auf Seite 197.

Lua

Lua [**?**, **?**] ist eine Programmiersprache, die sich besonders für Scripting-Aufgaben eignet. Zusammen mit dem **[C++-Binding??]** *luabind* entsteht so eine mächtige Scriptingumgebung, die in unserem System genutzt wird. Näheres findet sich in Abschnitt 12.5 auf Seite 108.

ODE

Die *Open Dynamics Engine* [**?**] ist eine Open-Source-Bibliothek für die Simulation der physikalischen Dynamik von nicht-deformierbaren Körpern. ODE wird von der Grafikengine für die Physik-Simulation und die Kollisionen von Objekten in der Engine genutzt. Mehr findet sich dazu in Abschnitt 12.2 auf Seite 98.

Ogre

Ogre [**?**] ist ein plattformunabhängiges Rendering-Framework für 3D-Grafik und wird vom System als Basis für die Grafikdarstellung verwendet. Näheres siehe unten (12.1.2, S.85).

OpenGL / Direct3D

OpenGL und *Direct3D* sind zwei *Hardware Abstraction Layers*, die es erlauben, 3D-Hardware geräteunabhängig zu programmieren. Während OpenGL ein plattformunabhängiger Standard ist, ist Direct3D nur für Windows verfügbar. Unser System unterstützt, über OGRE (siehe oben), das Rendering über beide dieser Schnittstellen.

P5 SDK

Das *P5 Software Development Kit* ist die Softwareschnittstelle zum Auslesen der Eingabewerte vom P5 **[Datenhandschuh??]** unter Windows und Linux, der experimentell im System verwendet wurde. Eine genauere Beschreibung des Handschuhs findet sich unter Abschnitt 14 auf Seite 129.

TinyXML

TinyXML [**?**] ist eine kleine und schlanke Bibliothek zur Verarbeitung von **[XML??]**-Dateien. Die Bibliothek erlaubt es, XML-Dateien einzulesen (ohne Unterstützung von Document Type Definitions oder XML Stylesheets), zu modifizieren und zu speichern. Unser System benutzt TinyXML an verschiedenen Stellen, insbesondere zum Parsen von Konfigurations- und Szenendateien in der Engine.

Teil V

Teilkomponenten

Kapitel 11

CAVE

11.1 CAVE-Definition

Eine CAVE Automated Virtual Environment (rekursives Akronym, die Bezeichnung geht auf [**?**] zurück) ist ein Raum zur Projektion einer dreidimensionalen Illusionswelt, der virtuellen Realität. Dabei wird die virtuelle Realität in einer in Echtzeit computergenerierten virtuellen Umgebung als Darstellung und als gleichzeitige Wahrnehmung der Wirklichkeit und ihrer physikalischen Eignschaschften bezeichnet. In diesen virtuellen Umgebungen werden beispielsweise unrealistische Welten dargestellt, die zum einen ein Abbild der Realität oder auch ein Abbild einer künstlichen Welten sein können. Des Weiteren ist es möglich Umgebungen der Vergangenheit als auch Umgebungen der Zukunft darzustellen, wie auch in der Realität nicht wahrnehmbare Umgebungen. Das National Center for Supercomputing Applications (**[NCSA??]**) definiert CAVE als "an immersive with spatially engaging environments" [**?**]. Die Bezeichnung "CAVE" rekurriert bewusst auf das **[Hoehlengleichnis??]** in Platons "Republik", das sich mit dem Verhältnis von Wahrnehmung und Erkenntnis sowie Realität und Illusion beschäftigt.

Weitere Merkmale einer CAVE sind die Interaktionen, an denen sich der im CAVE befindende Teilnehmer beteiligt. Zum einen kann er sich frei im Raum bewegen, zum anderen können Objekte manipuliert werden. Des Weiteren werden verschiedene menschliche Sinne angesprochen, wie z.B. der visuelle Sinn, durch die Wahrnehumung der virtuellen Welt, der akustische Sinn, durch das Reagieren auf Sound oder Musik zur Welt oder zu Objekten der Welt und der Tastsinn, der z.B. durch **[forcefeedback??]** angesprochen werden kann.

Komponenten einer CAVE:

Computer: Je nach Anzahl der Projektionsflächen, müssen entsprechende Computer vorhanden sein, um die virtuelle Welt darstellen zu lassen. Dabei ist einer der Rechner der sogenannte Master und hat die Aufgabe, alle anderen Computer (Slaves) zu koordinieren und zu steuern.

Ein Graphik-System: Da die CAVE hauptsächlich aus einem graphischen System besteht, ist es wichtig, den hohen Anforderungen genüge zu leisten. Besonders zu beachten sind dabei die Projektionen: der Abstand der Beamer zu den Wänden, die Auflösung und die richtige Ausrichtung, so dass keine Übergänge von einer Seite zur nächsten zu sehen sind.

Ein Sound-System: Das Sound-System sollte so gut ausgestattet sein, dass der Benutzer das Gefühl bekommt, mit Hilfe von Musik, in die Welt einzutauchen. Dabei ist nicht nur die Musik selbst zu betrachten, sondern auch die richtige und optimale Aufstellung der Sound-Boxen.

Sonstige Hardware: Des Weiteren können Hardware-Systeme unterschiedlichster Art dazu beitragen, der Welt mehr Interaktivität zu verleihen. Dies kann z.B. durch einen **[Datenhandschuh??]**, durch

Joysticks oder sonstige Geräte passieren, die u.a. auch **[forcefeedback??]** ermöglichen

11.1.1 Stand der Technik

Am 16. Mai 2001 eröffnete das **[IAO??]** in Stuttgart eine Einrichtung unter dem Namen “HyPI-6“, eine vollimmersive 6-Seiten-CAVE (siehe auch **[Immersive??]**, die als die derzeit innovativste **[VR??]**-Umgebung weltweit gilt. Die komplette Anlage wurde von SGIs Dienstleistungsorganisation **[PSO??]**(PSO) integriert, welche als Generalunternehmer wirkte. Als HighEnd-Visualisierungssystem ist eine SGI Onyx 3400 mit 6 Pipes installiert - das ist hierzulande die derzeit mächtigste Graphikausstattung dieser neuen Onyx-Modellreihe.

Die **[VR??]**-Installation besteht aus 6 Glaswänden, wobei eine auch als Türe dient, um in den Würfel förmigen Raum zu gelangen. Insgesamt 12 Projektoren des Typs Barco 909 sind für die aufwendige Rückprojektion verbaut, 2 Stück pro CAVE-Fläche für beide Stereokanäle. Erstmals wurde keine Leinwand oder ähnliches verwendet, sondern festes Scheibenmaterial eingesetzt. Die Rückprojektion auf das spezielle, mikrodiffusionsbeschichtete Glas führt zu bislang ungekannter Bildqualität mit hohem Kontrast bei der Modelldarstellung. Gleichzeitig kann der Boden - bisher einmalig - sehr hoch belastet werden und lässt somit auch die Verwendung von Hardware-Mock-Ups in der CAVE zu. Zur Bestimmung der Position des Betrachters relativ zum virtuellen 3D-Objekt wird ein Tracking-System verwendet, das sich im Rucksack des Masters (die Person, die das Tracking und damit die Perspektive vorgibt) befindet. Das hat den Vorteil, dass es keine Stolperfallen durch Kabel mehr gibt. Neu ist auch das Rechner-Hybrid-Konzept, dass sowohl die Ansteuerung eines Grafik-Supercomputer SGI Onyx 3400 als auch eines 12-PC-Multi-Node-Clusters erlaubt.

11.2 Unser Cave

Der Cave ist in mehreren Schritten gefertigt worden. Zuerst wurde ein Prototyp erstellt, an dem dann kontinuierlich weitergearbeitet wurde. Herausgekommen ist ein sehr mobiler Cave, der sich innerhalb kürzester Zeit auf- und abbauen lässt. Die Auswahl und Anpassung der Projektionsflächen war ebenfalls Bestandteil der Arbeit. Schließlich wurde die Entscheidung gefällt, einen weißen Stoff als Fläche zu verwenden. Dieser wurde mittels Klettverschlüssen an dem Holzgestell befestigt und lieferte hervorragende Ergebnisse bei der Projektion.

Zu Beginn des dritten Semesters wurden zunächst einmal die Anforderungen an den zu konstruierenden Cave neu definiert. Die Mobilität sollte bei der nächsten Ausbaustufe etwas weiter in den Vordergrund rücken und einen leichteren Transport der Einzelteile ermöglichen. Ein neues Verbindungssystem der Querverstrebungen sollte einen werkzeuglosen Auf- und Abbau des Caves garantieren. Das Prinzip der Projektionsflächen musste ebenfalls völlig neu überdacht werden, da diese den Strapazen eines häufigen Wiederaufbaus der Anlage auf Dauer nicht überstehen würden.

11.2.1 Materialien

Holzgerüst

Durch Materialeinsparungen konnte schnell ein zufriedenstellendes Ergebnis im Bereich des Gesamtgewichts erzielt und ein leichterer Transport der Einzelteile erreicht werden. Das schnelle, unkomplizierte und werkzeuglose Auf- und Abbauen bereitete deutlich mehr Probleme, so dass nachdem eine geeignetes Verbindungssystem gefunden wurde, die Suche nach einer endgültigen Lösung keineswegs beendet war. Fehlende Erfahrungen und vom Material abhängige Gegebenheiten, ließen einige Versuche ohne

Abbildung 11.1: Unser Cave am Projekttag

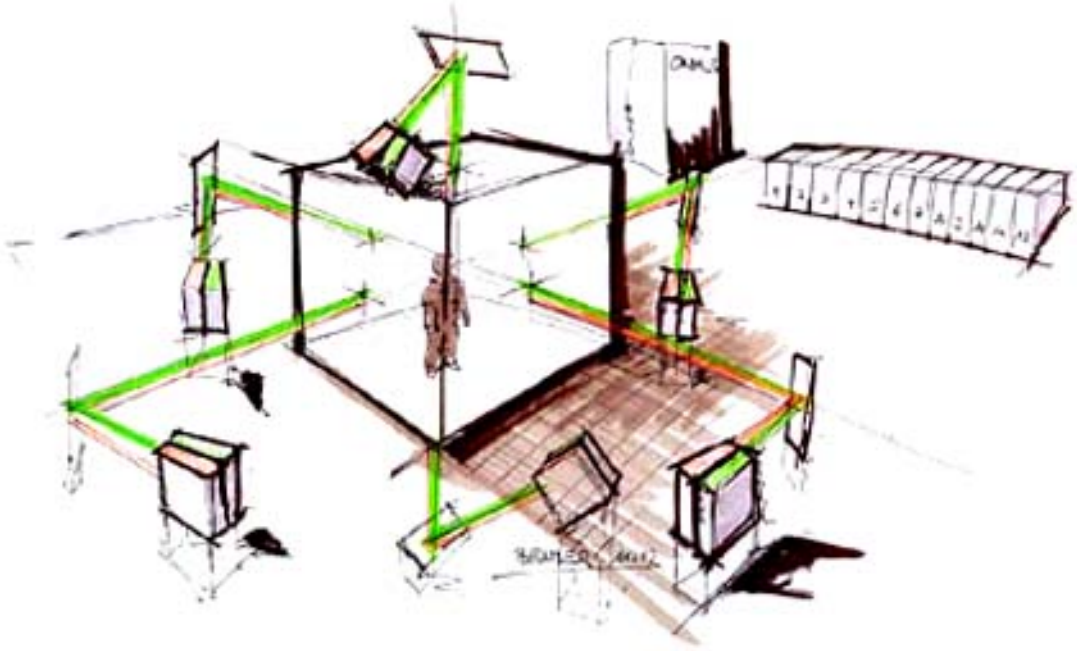


Abbildung 11.2: 6-Seiten-CAVE des FHI

ein zufriedenstellendes Ergebnis verweilen. Eine Kombination aus einem “Kisten-Schnellspannsystem“ und einer Verbindung der Querleisten zu den vier Hauptsäulen des Holzkonstruktes mittels so genannter Zapfen, führte letztendlich zu einem den Anforderungen genügendem Ergebnis.

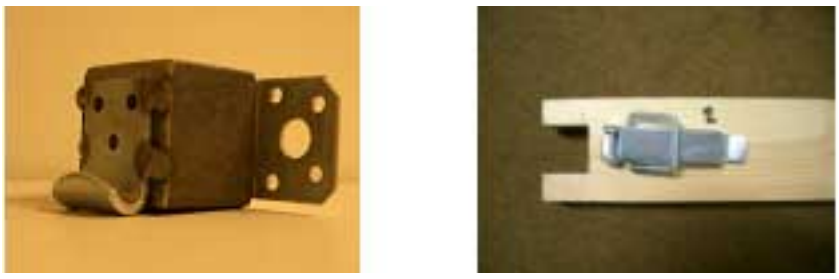


Abbildung 11.3: Schnellspannsystem

Das “Kisten-Schnellspannsystem“ konnte recht unkompliziert über den Sponsor Björn Ritschewald bestellt und zu einem sehr günstigen Preis (sechs geschenkt) erworben werden. Das Gegenstück zu den Holzzapfen bilden speziell konstruierte Metallzapfen, welche eine Führung für die Holzzapfen beinhalten und das Gegenstück der Schnellspannverschlüsse auf sich tragen.

Bis wir zum endgültigen Verbindungssystem gelangen konnten, waren einige Versuche und Überarbeitungen unserer Idee notwendig.

Projektionsflächen

Nach vielen Versuchen und Fehlschlägen wurde eine Lösung für die zukünftigen Projektsflächen gefunden, welche den erforderlichen Ansprüchen genügten und deren Projektionseigenschaften hervorragende Ergebnisse erzielten. Zu Bedenken war, dass sich die Helligkeit der Leinwand nach ihrer Fähigkeit

Abbildung 11.4: Verbindung

richtete, polarisiertes Licht zu bündeln. Je heller die Leinwand war, desto geringer wurde jedoch der Sichtwinkel und es kam zu einer verminderten Bildschärfe.

Es wurde sich deshalb für ein Stoffgemisch namens “Damast-Satin“ entschieden, welches mittels eines Klettverschluss-Bandes an das Gerüst angebracht werden konnten. Auf den ersten Blick schien dieses Verfahren etwas rustikal, bot aber überzeugende Eigenschaften in Hinsicht auf, Mobilität, Robustheit und Spannbarkeit. Nachdem das Verfahren in Folge weniger Fehltritte endlich ausgereift und zu einem ersten Test bereit war, wuchs in uns die Überzeugung, dass sich die Mühen und Strapazen, die uns dieses Verfahren bereitete, absolut gelohnt hatten.

11.3 Skizzen

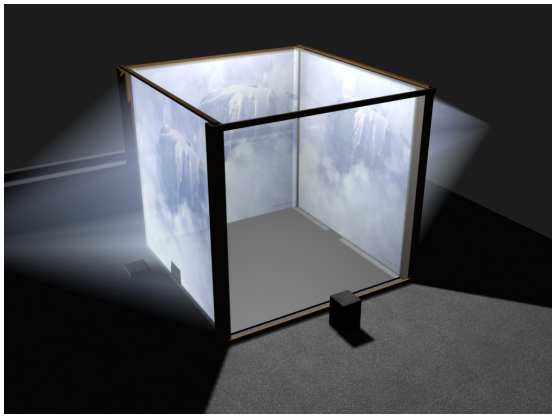


Abbildung 11.5: Seitenansicht (3D Studio Max)

11.4 Portabilität

Eine gute Portabilität konnte durch das Schnellspannsystem gewährleistet werden, da dieses so konstruiert war, dass ein schnelles Auf- und Abbauen ermöglicht wurde. Dies war die Intention unseres Cave-Bauplans. So war der Cave nicht nur portabel, sondern auch mit weniger als 6 Personen aufbaubar. Zusätzlich ergab sich durch die neuen Projektionsflächen des Caves eine einfacherer Handhabung, Aufbewahrung und ein leichteres Anbringen an die Querleisten und an die Hauptsäulen. Im Gegensatz zu den Projektionsflächen des Prototypen (siehe 9.3, S.52), die durch das leicht reißbare Material sehr

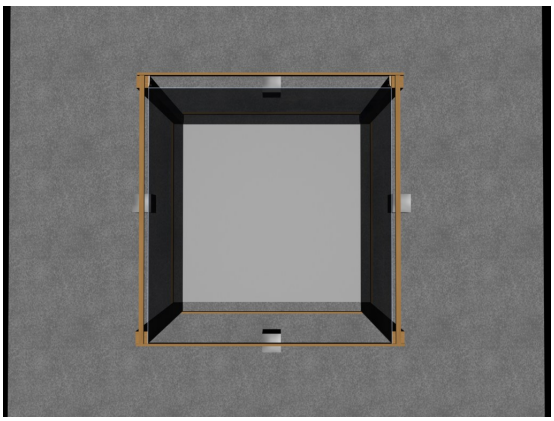


Abbildung 11.6: Draufsicht (3D Studio Max)

gefährdet waren, konnten diese Projektionsflächen aus Damast-Satin zusammengelegt werden. Da der Stoff so elastisch war, dass er sehr stramm gezogen werden konnte, entstanden beim Anbringen an den Cave keine Falten. So war auch dieses Problem gelöst.

11.5 Nachteile

Nachteile der Cave- Konstruktion ergaben sich mit der Zeit durch das wiederholte Aufbauen. Hier führte der Verschleiß bei der Wiederbenutzung des Holzgerüstes - aus sowohl ökonomischem Überlegen als auch aus finanziellen Gründen - zu Beeinträchtigungen. Durch die Verwendung des Holzgerüstes bei dem Prototypen(siehe 9.3, S.52), wurde dieses insofern ein wenig in Mitleidenschaft gezogen, als dass es den Schnellspannern nicht immer den nötigen Halt geben konnte. So waren einige der Schnellspanner mit der Zeit ausgeleiert. Aufgrund des quadratischen Systems des Caves, führte dies jedoch nicht zu einer minderen Stabilität. Daneben hinterließ das ständige Auf- und Abbauen auch beim Stoff der Projektionsflächen seine Spuren. So riss eine Fläche an einer Ecke ein, da der Stoff zu stramm gezogen wurde. Der Schaden war zwar an einer nicht so dramatischen Stelle, doch ließ sich der Anfangszustand natürlich nicht wieder herstellen, so dass eine Naht sichtbar blieb. Mit einer Projektionsfläche wie z.B. Plexiglas wäre dieses Problem gar nicht erst entstanden.

11.6 Projektionsüberlegungen

Unter einer Projektion (lat. proicere: hinauswerfen, hinwerfen) versteht man die Abbildung eines Urbildes auf eine geometrische Projektionsfläche, also durch Strahlen abgebildet. Die Entscheidung viel auf die Projektion von drei Flächen. Die ersten Überlegungen zielten darauf, Projektionen mit Hilfe von Spiegeln zu lösen. Dabei war es die Absicht, Raum einzusparen. Diese Überlegung wurde jedoch schnell wieder verworfen, da die Abstände, die nötig waren, um eine angemessene Größe durch den Spiegel zu erzeugen, nicht so viel Raum einsparten, als dass es sich lohnte, diesen Weg weiterhin einzuschlagen. Also wurde sich für die einfache Projektionsvorschrift der Zentralprojektion entschieden. Wichtig dabei war es, den Beamer mit seinen Strahlen als Lot auf die Projektionsflächen so zu justieren, dass kein Schrägbild geschaffen wurde, was zu Verzerrungen des Bildes führen würde. Ebenso musste das Spiegeln, Drehen und Anpassen der unterschiedlichen Welten bedacht werden, da sich der Benutzer des Caves im Inneren des Caves befindet und die Welt somit spiegelverkehrt betrachtet. Diese Einstellungen konnten durch Funktionen der Beamer gelöst werden. Ein fließender Übergang von einer Fläche zur daneben liegenden war ebenfalls zu bedenken.

11.6.1 Getestete Medien

Nach dem Testen der Projektionsflächen des ersten Caves und der Erörterung der Vor- und Nachteile, sollte eine Fläche benutzt werden, die weder aus zwei Flächen pro Seite bestand, noch sollte sie so schwer und mühsam aufzubauen sein, wie es bei dem ersten Cave der Fall gewesen ist. Zudem war die Gefahr des Reißens der Flächen sehr groß. Verschiedene Aspekte spielten bei der Abwägung der "richtigen" Fläche eine Rolle:

- Kosten
- Qualität der Projektion
- Pro Seite eine Fläche ohne Naht oder ähnlichem
- Portabilität
- Stabilität bei häufigem Aufbauen des Caves

Folgende Materialien wurden getestet:

Plexiglas

Plexiglas ergab beim Test ein sehr gutes Ergebnis bei der Projektion. Qualitativ hätte man sich mit diesem Material sehr zufrieden geben können. Doch um eine gute Portabilität zu erreichen, kam dieses Produkt keineswegs in Frage. Sowohl die schlechte Mobilität durch das hohe Gewicht und die Größe, als auch die hohen Kosten, die durch das Zuschneiden auf die richtige Größe und die enorm hohen Materialkosten zustande gekommen wären, ließen eine Zustimmung für dieses Produkt nicht zu.

Glasfaserprodukte

Glasfaserprodukte erzielten zu keinem Zeitpunkt ein positives Ergebnis. Die Projektionseigenschaften waren so miserabel, dass wir uns trotz äußerst geringer Kosten für dieses Produkt nicht entscheiden konnten.

Mit Flusssäure behandeltes Glas

Mit **[Flusssaeure??]** behandeltes Glas ergab in der Qualität ähnlich gute Ergebnisse wie das Plexiglas, als auch ähnlich hohe Kosten und schlechte Mobilität durch das zu hohe Gewicht. So war auch dieses Produkt nicht annehmbar für unsere Anforderungen.

Verschiedene Textil-Stoffe

Wie bereits in Punkt 11.2.1 "Materialien - Projektionsflächen" erwähnt, viel der Entschluss auf das Stoffgemisch "Damast-Satin", welche in fast allen Punkten die Anforderungen erfüllte. Zwar war der Arbeitsaufwand durch das eigene Abmessen und Nähen sehr hoch, doch glich das gute Ergebnis bei der Projektion, die gute Mobilität und die bezahlbaren Kosten dieses wieder aus.

Kapitel 12

Engine

In diesem Abschnitt soll der Software-Entwurf und die Struktur der Kern-Engine des Systems beschrieben werden, insbesondere der Aspekt der Grafikdarstellung mit allen damit zusammenhängenden Themen sowie die Kopplung mit allen anderen Komponenten. Getrennt in eigenen Teilen beschrieben wird dabei das Soundsystem (15, S.197) sowie das Low-Level-Interface zu den Hardwarekomponenten (14, S.129).

12.1 Grafik

12.1.1 Grundüberlegungen

Aus der Anforderungsdefinition (siehe Anhang B auf Seite 221) ergeben sich die erforderlichen Leistungen der Grafik in der Engine. Insbesondere muss die Grafik in Echtzeit erzeugt werden (und somit unter Verwendung von 3D-Hardware), die Darstellung von großen Szenen erlauben und auch das verteilte Rendern auf mehreren Rechnern für die Ansteuerung mehrerer Projektorenunterstützen. In den folgenden Abschnitten wollen wir nun unsere Lösungen für diese und weitere Anforderungen beschreiben.

12.1.2 OGRE

Wie schon in der Übersicht beschrieben, verwenden wir das OGRE-Framework [**?**] als Basis für die Grafikdarstellung. OGRE (*Object-Oriented Graphics Rendering Engine*) ist eine quelloffene Rasterisierungs-Engine, die von Grund auf komplett **[objekt-orientiert??]** in C++ geschrieben ist. Dabei ist die Engine komplett *plattformunabhängig* und kann auf verschiedenen 3D-Bibliotheken, insbesondere OpenGL und Direct3D, aufsetzen, um eine hardwarebeschleunigte Darstellung zu bieten. OGRE ist allerdings *keine* vollständige Spiele-Engine, sondern bietet reine Grafikdarstellungsmöglichkeiten in einem flexiblen Framework.

Einige wichtige weitere Features von OGRE sind:

- Material/Shader-Management
- eigenes **[Mesh??]**-Format, Unterstützung von Animationen und Level-of-Detail
- Verwendung verschiedener Szenenmanager (z.B. eigener)
- Spezialeffekte (Partikelsysteme, Billboards, ..)
- **[PlugIn??]**-Architektur zur Erweiterung

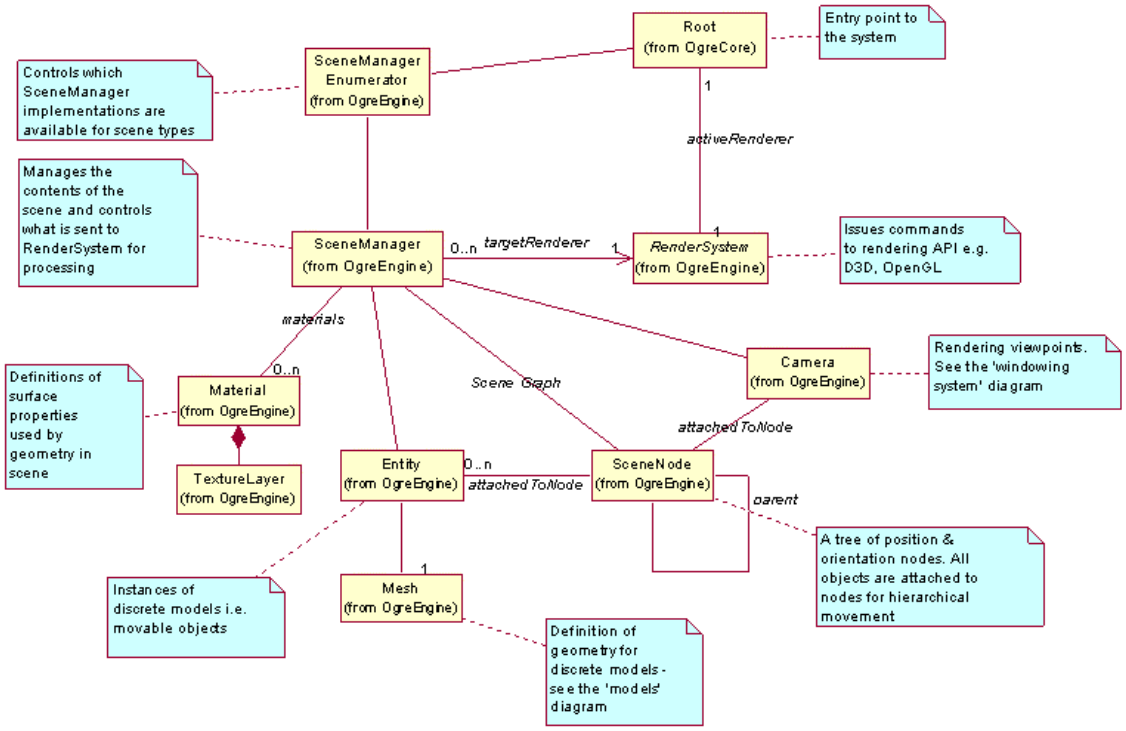


Abbildung 12.1: Grundlegende Klassenstruktur von OGRE

- lokale Schatten
- volle Dokumentation aller Klassen

Wir haben uns in unserem System für die Verwendung von OGRE aus mehren Gründen entschieden: Zum einen ist durch die komplette objekt-orientierte Struktur der Engine ein sauberes Einbinden in unser System und Erweitern von OGRE (siehe folgende Abschnitte) so gut möglich, zum anderen wollten wir unser System weitestgehend plattform- und auch 3D-**[API??]**-unabhängig halten. Dadurch, dass OGRE keine vollständige Spiele-Umgebung bereitstellt, sind wir auch nicht zur Verwendung einer vorgegebenen Umgebung gezwungen und konnten deswegen unsere Eigenentwicklungen verwenden.

Abbildung 12.1 zeigt die Struktur der wichtigsten OGRE-Klassen: insbesondere ist die Abstraktion von der unterliegenden 3D-**[API??]** (z.B. OpenGL, Direct3D) über die Klasse `RenderSystem` wichtig sowie die Verwendung einer Unterklasse von `SceneManager` für die Verwaltung des Szenengraphen mit der Liste der Modelle (`SceneNode`, `Entity`) in der Szene (so ist auch unserer eigener Szenenmanager etwa eine Unterklasse von `SceneManager`). Nicht sichtbar sind etwa Klassen zum Ressourcenmanagement (z.B. Materialien), die sich um das Laden und Verwalten von allen benötigten Daten kümmern.

12.1.3 Szenen Aufbau

Eine der Basis-Komponenten der Micarpet Engine ist die Unterstützung von ladbaren Level Dateien.

Jedes Level besteht dabei aus folgenden Teilen :

- Einer Datei <level_name>.mcs, die die Beschreibung des Levels enthält.
- Mehrere 'include' Dateien <objekt_typ>.mct, die eine Objekt-Typ Beschreibung enthalten.
- Mehreren Ressourcen für die Ogre Engine, wie Material-Skripten, Texturen, **[Mesh??]** und Sound Dateien.

Die Szenen Dateien der Typen Micarpet Scene (mcs) und Micarpet Typedef (mct) sind XML Dateien. Diese werden vom mcs/mct 3D-Studio MAX Exporter erzeugt.

MCS/MCT Exporter

Früh wurde entschieden, dass 3D-Studio MAX als Objekt- und Level-Editor benutzt werden soll (siehe auch Teil 13)

Für jedes modellierte Objekt wird dabei ein Objekt-Typ erzeugt und dann für jedes Auftauchen dieses Objekt-Typen in der Szene eine Objekt-Instanz erzeugt. Mit der Erzeugung von Objekt-Typen geht auch die Erzeugung des entsprechenden Material-Skriptes einher und der notwendigen Mesh Dateien für diesen Typ. Jede Objekt-Instanz erhält dann nur noch den entsprechenden Objekt-Typ und die entsprechende Position und Ausrichtung.

Für die Benutzung als Objekt-Editor wurde der 'mctexp' oder mct Exporter geschrieben. Dieser exportiert alle Objekte einer Szene als einen einzigen Objekt-Typ mit einer entsprechenden Anzahl an Entities, wenn nötig. Entities sind 'Unterobjekte', die zum gleichen Objekt gehören, aber ein anderes Material haben. In diesem Exporter kann man die Erzeugung von Kollisionsproxies angeben, wenn welche erzeugt werden sollen. Weiter erzeugt dieser alle notwendigen Material-Skripts und **[Mesh??]**es, die dieser Objekt-Typ und seine Entities benötigen.

Für die Benutzung als Level-Editor wurde der 'mcsexp' oder mcs Exporter geschrieben. Dieser exportiert alle Objekte einer Szene in eine mcs Datei. Für Objekte, die einen Namen haben, der der Konvention _typ_name entspricht, wird dessen Objekt-Typ nicht neu erzeugt, sondern es wird ein 'include' für die entsprechende <typ>.mct eingefügt. Dann wird für dieses Objekt eine Instanz mit dem Typen typ erzeugt. Diese Objekte werden also als Impostor für Objekte, die mit dem Objekt-Editor erzeugt worden sind, behandelt.

Laden von Levels

Wenn ein Level geladen wird, wird dessen mcs Datei von der Engine interpretiert und es werden die entsprechenden Objekte im Level erzeugt und initialisiert.

Eine mcs Datei ist grundlegend wie folgt aufgebaut :

- Allgemeine Informationen über das Level wie z.B. Einstellungen für den Fog oder die Textur des **[Skydome??]**s.
- 'inlcude' Anweisungen für die notwendigen mct Dateien.
- Objekt-Typen die in diesem Level gebraucht werden und in keiner mct Datei beschrieben sind.

- Objekt-Instanzen, die die Position/Ausrichtung/etc. der Objekte in der Welt angeben.

Damit man Objekte nur einmal erzeugen muss und dann mehrmals im Level benutzen kann, wird streng zwischen Objekt-Typen und Objekt-Instanzen unterschieden. Der Objekt-Typ gibt dabei an, wie das Objekt aussieht, wie es sich verhält und welche Eigenschaften es hat. Die Objekt-Instanz gibt an, wo sich eine Instanz des Objekt-Typs sich in der Welt befindet.

Die mct Dateien enthalten in der Regel nur genau eine Objekt-Typen Definition. Diese Definition kann dann von mehreren Level-Dateien aus referenziert werden. Dies wird zum Beispiel beim ’player’ gemacht, wobei es für diesen die Datei ’player.mct’ gibt. Das hat den Vorteil, dass wenn Änderungen an diesem Objekt gemacht werden, diese in alle anderen Level übernommen werden, ohne dass diese verändert werden müssten.

Der grobe Ablauf zum Laden eines Levels sieht wie folgt aus :

- öffnen der zum Level gehörenden mcs Datei
- Laden der ’Ambient’ Daten
- Laden aller referenzierten ’include’ Dateien
- Parsen aller Objekt-Typen und damit einhergehend das Erzeugen der entsprechenden Objekt-Typen Klassen Instanzen (siehe alle CBaseObject abgeleiteten Klassen).
- Erzeugen der Objekt-Instanzen anhand der Objekt-Typen
- Starten des Levels

Hier ist ein Beispiel für ein sehr einfaches Level :

```
<?xml version="1.0" encoding="utf-8" ?>
<scene name="level01">
    <ambient>
        <color r="0.8157" g="0.8157" b="0.8157" a="1"/>
        <background tiling="8" texture="Skybox_Map_121122752"/>
        <bgsound bg="../res/SSX - Hybrid - Finished Sym.mp3" />
        <fog mode="LINEAR" start="1300" end="5000" exp="0.08"/>
    </ambient>

    <scripts>
        <script name="level1.lua" />
    </scripts>

    <include filename="../res/player.mct"/>

    <objectinstance name="player" objecttype="player">
        <pos x="-1111.9600" y="543.6072" z="-2455.6638"/>
        <rot wx="-0.1407" wy="-0.0062" wz="-0.9900" ww="0.0009"/>
        <lookat x="42.4473" y="1229.8430" z="591.3674"/>
    </objectinstance>
</scene>
```

Die Funktionalität zum Laden der einzelnen Objekte stellt dabei die Klasse `CBaseObjectTemplate` zu Verfügung. Diese initialsiert die gemeinsamen Eigenschaften aller Objekt-Typen. So werden alle Lichter Definitionen, Entities, Kollisionsdaten, etc. von dieser Klasse für einen Objekt-Typ geladen und jeder Instanz dieses Objekt-Typs diese Daten zugeordnet.

mcs/mct Format Definitionen

Allgemeine Definitionen Rotationen können wie folgt definiert werden :

- `<rot wx="<float>" wy="<float>" wz="<float>" ww="<float>" />`
- `<rot x="<float>" y="<float>" z="<float>" deg="<float>" />`
- `<rot yaw="<float>" pitch="<float>" roll="<float>" />`
- `<quat wx="<float>" wy="<float>" wz="<float>" ww="<float>" />`
- `<orientation wx="<float>" wy="<float>" wz="<float>" ww="<float>" />`

Scale-Faktoren können wie folgt angegeben werden :

- `<scale x="<float>" y="<float>" z="<float>" />`
- `<scale f="<float>" />`

Jeder nicht übergegebene Parameter wird als <float> == 1.0f angenommen.

Farb Objekte kann es nur in einem Format geben und zwar wie folgt.
`<color r="<float>" g="<float>" b="<float>" a="<float>" />`
Wenn r, g oder b nicht angegeben wird, wird dieser Komponente der Wert 0.0f zugewiesen. Wenn a nicht angegeben wird, wird ebenfalls 1.0f zugewiesen.

Objekt-Typen Objekt-Typen werden wie folgt angegeben :

```
<objecttype name="<string>" type="<string>" >
    <entity name="<string>" filename="<string>">
        <pos />
        <rot />
    </entity>
    <physim>
        <collision_proxy />
    </physim>
    <light />
</objecttype>
```

Das Attribut <type> kann dabei den Wert ’dynamic’ oder ’static’ annehmen:

type = ’dynamic’ Objekt-Typen, die diesen Typ haben, sind Objekte mit *etwas* Intelligenz.

Diese können folgenden Node enthalten :
`<data movscale="<float>" maxspeed="<float>" />`

Diese Objekte können in der Instanz eine Pfadliste enthalten, diese sieht wie folgt aus :

```
<path>
    <waypoint x="112" y="2" z="35" />
    ...
</path>
```

type = 'static' Objekt-Typen, die diesen Typ haben, sind Objekte, welche keine eigene Bewegungslogik in der Welt haben.

Nodes vom Typ <entity> sind Kinder-Objekte innerhalb des Vater-Objekt-Typs. Genauer gesagt sind sie Unter-**[Mesh??]**es des Objektes, die, wenn das Objekt bewegt wird, alle relativ zum Vaterobjekt mitbewegt werden.

Die Positionen/Rotationen sind immer relativ zum Vater Ursprung (0,0,0).

Entities können keine Kollisions-Proxies enthalten, sie können aber ein Mesh haben (Filename Attribut).

Nodes vom Typ <light> bedeuten, dass der Objekt-Typ Lichter enthalten soll. Diese sind wie folgt aufgebaut :

```
<light name="bla" type="POINT">
    <specular r="" g="" b="" />
    <diffuse r="" g="" b="" />

    <attenuation range="<float>"
            constant="<float>"
            linear="<float>"
            quadratic="<float>" />

    <spotlightrange innerangle="<float>"
            outerangle="<float>"
            falloff="<float>" />
</light>
```

Das <type> Attribut des <light> Nodes kann einen der folgende Werte enthalten :

POINT Punktlicht, welches gleichmäßig in alle Richtungen abstrahlt.

DIRECTIONAL Lichtquelle, die parallele Lichstrahlen aussendet. Normalerweise wird diese Lichtquelle zur Simulation des Sonnenlichts benutzt.

SPOTLIGHT Spotlight Lichtquelle.

Kollisionsbehandlung/Proxies Die Kollisionsbehandlung wird über die Anbindung zu ODE ausgeführt 12.2. Dabei werden aus dem Objekt-Typ Node in der mcs/mct Datei die notwendigen Daten geholt, um die Kollisions-Proxies 12.2 zu erzeugen. Im folgenden werden die Elemente der mcs/mct Datei aufgelistet und beschrieben, die diese Daten enthalten.

Jeder Objekt-Typ, der Teil der physikalischen Simulation sein soll, muss den folgenden Node haben:

```
<physim phybody="<bool>"
        gen_proxy="<type>"
        gen_proxy_mass="<float>"
```

```
        gravity="<int>" />
```

phybody Gibt an, ob der Körper selbst dynamisch ist und mit der Welt interagieren kann und selbst agiert. Das bedeutet, dass der Körper bei Kollisionen mitbewegt wird. Zum Beispiel sollte die Landschaft phybody = 0 haben, da sie nicht bewegt werden soll.

gen_proxy sagt ob die Engine selbst einen Kollisions-Proxy für das Objekt erzeugen soll. Dieser wird aus dem **[Mesh??]** des Objekt-Typen erstellt, wobei das type Attribut einen der folgenden Werte annehmen kann :

- sphere erzeugt eine kleinste umschließende Kugel um den Objekt-Typ
- box erzeugt eine kleinste umschließende AABB um den Objekt-Typ
- trimesh erzeugt einen Körper aus Dreiecken, der als Kollisions-Proxy fungiert

gen_proxy_mass gibt die Masse des generierten Proxies an. Um sich eine vernünftige Beschleunigung des Körpers zu erhalten, sollte dieser eine Masse haben, ansonsten bekommt die Simulation Fehler durch Singularitäten.

gravity gibt an, ob sich das Objekt von der Gravitation beeinflussen läßt. Als Standard wird angenommen, dass der Körper nicht auf die Gravitation reagiert.

Die ODE Anbindung kann mit Hilfe der Daten im physim Node entweder durch die Angabe eines Typs in gen_proxy den Proxy selbst erzeugen oder über den im physim Node enthaltenen collision_proxy Node die Daten der Proxies auslesen.

Die Definition fuer eine Liste von collision_proxy Elementen steht innerhalb des physim Nodes und sieht wie folgt aus:

```
<collision_proxy>
    <trimesh mesh="<mesh_name>"/>

    <box x="<float>" y="<float>" z="<float>"
            mass="<float>" />

    <sphere r="<float>" mass="<float>"/>
</collision_proxy>
```

Der collision_proxy Node kann beliebig viele der folgenden Kollisions-Proxy Typen enthlaten, die dann zusammengefasst werden als Kollisions-Volumen um den Objekt-Typ herum.

trimesh Ein trimesh ist ein Kollisions-Proxy, der aus den Dreiecks-Daten des angegebenen **[Mesh??]**es generiert wird.

box Eine Box die als Kollisions-Proxy den Objekt-Typen umschließen sollte. Die Attribute x, y, z geben die Länge der Box entlang der X/Y/Z-Achse an, wobei die Box selber dann als Mittelpunkt (0/0/0) bekommt.

sphere Eine Kugel die als Kollisions-Proxy den Objekt-Typen umschließen sollte. Das Attribut r gibt dabei den Radius der Kugel an.

Ein Proxy kann Positions und Rotations Daten haben, die definieren wie dieser Proxy relativ zum Objekt-Typ orientiert/positioniert ist. Diese Daten sind immer relativ zum Ursprung des Objekt-Typs (0/0/0). Hier ein Beispiel :

```
<trimesh>
    <rot ... />
    <pos .../>
</trimesh>
```

Wenn kein collision_proxy im Objekt-Typ enthalten ist, wird davon ausgegangen, dass das Objekt nicht mit anderen Objekten kollidieren kann/soll. Wenn der collision_proxy leer ist, wird davon ausgegangen, dass ein trimesh mit 1:1 Abbildung genutzt werden soll.

Level Eigenschaften Die globalen oder 'ambient' Eigenschaften werden mit folgendem Node angegeben :

```
<ambient>
    <fog mode="<mode>"
        start="<float>" end="<float>"
        exp="<float>">
        <color ... />
    </fog>
    <phyworld gravity="<float>"
                deaccel="<float>"
                disable_factor="<float>" />
</ambient>
```

fog Bestimmt die Eigenschaften des Nebels, der ab einer bestimmten Entfernung dem Spieler den Blick auf weiter entfernte Teile des Levels versperrt.

mode Gibt an, ob der Modus des Nebels Linear, Exponentiell, etc ... von start nach end sich verdichtet. mode kann folgende Werte annehmen LINEAR, EXP, EXP2 und NONE.

start Der Abstand zum Spieler, an dem der Nebels anfängt.

end Der Abstand zum Spieler, an dem der Nebel undurchsichtig wird.

exp Parameter für die Modi EXP und EXP2 für den Nebelverlauf.

color Die Farbe des Nebels.

phyworld Enthält die Einstellungen, die die Simulation der physikalischen Gegebenheiten der Welt bestimmen.

gravity Enthält die Länge des Gravitationsvektor, der auf die Objekte wirkt, die das 'gravity' Attribut gesetzt haben.
Wenn dieser Wert nicht gesetzt wird, wird 0.981 als Länge genommen.

deaccel Dies ist der Faktor, der die Luftreibung (und ähnliche Faktoren) simulieren soll und jedes sich bewegende Objekt abbremst. Dieser sollte immer >= 0 sein, sonst beschleunigen die Objekte unkontrollierbar.
Wenn dieser Wert nicht angeben wird, wird ein Wert von 0.01 benutzt.

disable_factor Faktor, der angibt, ab welcher minimalen Geschwindigkeit, der Körper 'ausgeschaltet' werden soll, also nicht mehr mit anderen Objekten kollidiert. In manchen Situationen kann man mit diesem Faktor Zittern vermeiden.
Wenn dieser Wert nicht angeben wird, wird ein Wert von 0.05 benutzt.

12.1.4 Terrain-Visualisierung

Entwurf

Mit dem Entwurf des Projektthemas und der Anforderungsdefinition war klar, dass das Spielszenario eine offene Außenszene bedingte (Inseln im Meer). Daraus folgte, dass die Grafikengine zwingend eine gute Visualisierung für die darzustellende Landschaft brauchte, um eine möglichst realistische Darstellung der Landschaft zu bieten. Gleichzeitig sollte es der Modellierungsgruppe möglich sein, eigenständig und schnell das Terrain zu entwerfen und zu bearbeiten.

Generell gibt es für die Generierung und Darstellung von Terrain zwei grundlegende Möglichkeiten:

- Darstellung über eine *Heightmap* (siehe Abbildung 12.2). Hier wird als Basis für das Terrain ein Grauwertbild (oder mehrere) genommen und die Werte des Bildes als Höheninformationen interpretiert. Aus diesen Höhenwerten wird dann zur Laufzeit ein normales Dreiecksmesh erzeugt, das einfach als normales 3D-Objekt gerendert wird. Dieser Ansatz hat den großen Vorteil, konzeptuell simpel zu sein (es ist lediglich ein Gitter von Punkten zu erstellen und diese Punkte entsprechend den Werten in der Heightmap vertikal zu verschieben). Weiterhin lassen sich die Heightmaps einfach generieren (etwa mit einem Fraktal-Generator) und benötigen wenig Speicher.

  Nachteile dieses Verfahrens sind allerdings, dass das Terrain immer auf eine einfache Struktur beschränkt bleibt, d.h. die Steigung des Terrains ist limitiert, weiterhin sind keine Überhänge oder ähnliches möglich. Gleichermaßen sind natürlich nicht so einfach unregelmässig geformte Landschaften oder ähnliches möglich. Weiterhin ergibt sich gestalterisch das Problem, dass die Integration mit Standard-Programmen zur Modellierung schwierig ist: da die Heightmap natürlich von der übrigen Geometrie (Bäume, ..) völlig getrennt ist, fällt es bei der Gestaltung der Gesamtszene schwer, Objekte in der Landschaft zu positionieren, da die Landschaftsgeometrie nur in der fertigen Anwendung, nicht aber im Modellierungsprogramm sichtbar ist.

- Darstellung als normales 3D-Modell; dies ist der Ansatz, den wir schließlich gewählt haben. Der Vorteil ist hier, dass die Modellierer hier völlige Freiheit bei der Gestaltung des Terrains besitzen - so sind problemlos etwa überhängende Klippen und sehr unregelmässiges Terrain möglich. Die grundlegende Modellierung geschah hierbei über einen Terraingenerator, welcher durch eine Simulation ein Terrainmodell erstellte, was dann über einen Export in das Modellierungsprogramm übernommen werden konnte. Dort konnten dann die Modellierer das gegebene Terrain per Hand nachbearbeiten und alle weiteren Objekte in der Szene plazieren.

  Der Nachteil der Darstellung als 3D-Objekt ist natürlich der höhere Speicheraufwand (ein fertig detailliertes Terrain besteht im System aus normalerweise aus 10.000 - 50.000 Dreiecken). Weiterhin ist natürlich die Geschwindigkeit bei der Darstellung niedriger als bei Heightmaps, welche grundsätzlich rechteckig sind und bei welche man beim Rendern einfach im Detail reduzieren kann, wenn sie weiter entfernt sind. Weiter kann man bei Heightmaps sehr einfach feststellen, welches Terrain gerade überhaupt sichtbar ist.

  Ein letzter - praktischer - Nachteil ist schließlich, daß es zur Darstellung von Terrain als 3D-Modell kaum fertige Algorithmen gibt, weswegen wir zur Visualisierung (siehe unten) eine eigene Implementation schreiben mussten. Für Heightmaps gibt es hingegen viele fertige Algorithmen, die relativ einfach hätten eingesetzt werden können.

Implementation

Das Grundproblem bei der Visualisierung eines Terrains als fertiges 3D-Modell ist nicht die Darstellung des Modells an sich, dies kann ja über die normale Grafikausgabe wie bei den restlichen Modellen

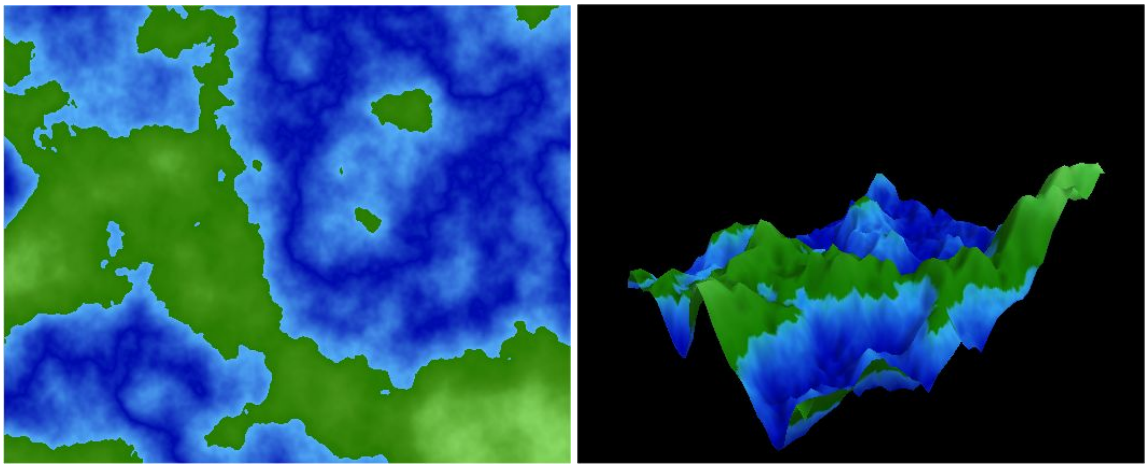


Abbildung 12.2: Eine Heightmap, generiert mit einem Fraktal-Programm und eine dazugehörige beispielhafte 3D-Visualisierung

geschehen. Wichtig ist dort nur, dass die Terrain-Modelle in kleinere Teile unterteilt werden, damit der Szenenmanager - welcher auf Granularität von ganzen Modellen arbeitet - die Teile des Terrains, die gerade nicht sichtbar sind, möglichst einfach ausblenden kann.

Das Problem, das sich vor allem stellte, war die „realistische“ Texturierung der vorgegebenen Terrain-Modelle. Allgemein ist eine Darstellung mit nur einer Textur für die Landschaft eher eintönig und kaum überzeugend. Bei echten Landschaften kann man etwa beobachten (natürlich im großen Maßstab gesehen), dass das Terrain sich aus mehreren Landschaftstypen (z.B. grün bewachsen im Flachland, dann schrittweise felsiger bei hügeligen Teilen und sandig in der Tiefe) zusammensetzt. Insofern ist naheliegend, das Terrain über eine Kombination von unterschiedlichen Texturen, die die entsprechenden Typen repräsentieren, darzustellen.

Es ist kaum realistisch, die komplette Texturierung aller Schichten der gesamten Landschaft per Hand im Modellierungsprogramm vorzunehmen, da eine Kombination mehrerer Texturschichten dort nur unter großen Umständen möglich ist oder nicht richtig in die spätere Anwendung exportierbar ist. Noch dazu würde eine spätere Änderung des unterliegenden Terrains dann auch immer eine Nachbesserung der Texturierung verlangen. Daher haben wir uns für eine Lösung entschieden, welche die komplette Texturierung vollautomatisch aufgrund von bestimmten Vorgaben des Modellierers vornimmt.

Im fertigen System lässt sich die Landschafts-Visualisierung in drei Teile unterteilen: Darstellung des Himmels, Darstellung des Wassers und Darstellung des eigentlichen Terrains.

Himmel Der Himmel kann über eine Textur (etwa Wolken) simuliert werden. Die Textur wird dabei auf eine Box um die gesamte Szene herum projiziert, wobei die Texturkoordinaten so verzerrt werden, dass der Eindruck eines gewölbten Himmels entsteht (*Skybox*). Die zu verwendende Textur wird in der Szenendatei vorgegeben und kann vom Gestalter schon im Modellierungsprogramm eingestellt werden.

Wasser Aufgrund des Szenarios „Inseln“ besteht auch die Notwendigkeit, größere Mengen Wasser für das umliegende Meer darzustellen. Ein fester Wasserspiegel führt außerdem dazu, dass man leicht den Eindruck einer fortgesetzten Szene erzeugen kann, indem das Wasser in alle Richtungen ausgedehnt wird (der Betrachter sieht somit keine feste Begrenzung der Szene). Wir stellen das Wasser als eine oder mehrere Ebenen in der Szene dar, welche mit beliebigen Textureffekten (z.B. Animation für die Illusion einer Wasserbewegung oder leichte Transparenz) belegt werden können. Die Modellierer können über

```
<?xml version="1.0" standalone=no>
<scene name="wassertext">
  <plane name="wasser1" width="50000" height="50000" material="Water/Water4" ypos="0"/>
  <plane name="wasser2" width="50000" height="50000" material="Water/WaterEffect4" ypos="15"/>
</scene>
```

Abbildung 12.3: Beispiel einer XML-Konfigurationsdatei für die Wasserebenen in der Szene

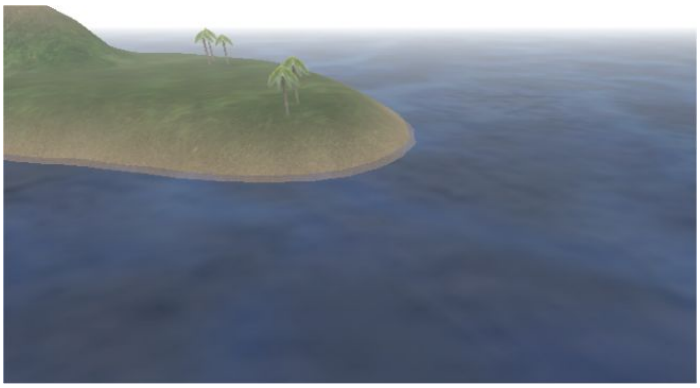


Abbildung 12.4: Visualisierung der Wasserebenen von der vorigen Abbildung im fertigen System

eine spezielle XML-Datei (siehe Abbildung 12.3) definieren, welche Ebenen in welcher Höhe dargestellt werden können. Die fertige Visualisierung des Wassers mit den zwei oben angegebenen Ebenen lässt sich in Abbildung 12.4 sehen.

Landschaft Wie schon oben beschrieben, ist die wesentliche Herausforderung zum Rendern des Terrains die Texturierung der Landschaft mit unterschiedlichen Textur-Schichten. Wir unterscheiden daher beim Rendern der Szene zwischen Objekten, die zur Landschaft gehören und allen sonstigen Objekten. Letztere werden ganz normal dargestellt (d.h. mit Textur/Materialinformationen aus dem Modell), alle Landschaftselemente werden aber dynamisch mit Materialien versehen.

Für die Durchführung der Texturierung ist nun ein Algorithmus zum Zuweisen der Texturen auf die Landschaftsmodelle nötig. Generell können wir eine Zuweisung der Texturschichten auf das Modell nur jeweils an den Eckpunkten (**[Vertices??]**) des Modells vornehmen, da bei der Darstellung immer komplette Dreiecke gerendert werden. Also muss für jeden Punkt der Landschaft entschieden werden, welche Schicht oder welche Kombination von Texturschichten er hat - die Grafikkarte interpoliert diese Angaben dann jeweils zwischen den drei Eckpunkten.

Hier gibt es zwei relativ einfache Zuweisungsmöglichkeiten: Zum einen kann man die Texturzuweisung rein über die Höhe berechnen (d.h. die y-Koordinate), so dass jeweils in einem bestimmten Bereich eine bestimmte Textur benutzt wird. Zum anderen kann man das Material über die Steigung des Dreiecks (d.h. die y-Koordinate des Normalenvektors) zuweisen. Beide Möglichkeiten lassen sich auf empirische Betrachtungen zurückführen: die Berechnung über die Höhe leitet sich aus den unterschiedlichen Landschaftstypen etwa an einem Berg ab, wobei mit steigender Höhe die Landschaft immer weniger Bewuchs zeigt; die Berechnung aus der Steigung rührt aus dem unterschiedlichen Pflanzenwachstum bei flachen Gebieten (Gras, Wald, ..) und steilen Flächen (eher felsig, eventuell Moose und ähnliches). Natürlich sind auch beliebig komplexe Modelle möglich, welche mit Simulationsmethoden wesentlich korrektere Zuweisungen als diese simplen empirischen Modelle vornehmen. Da eine realistische Berechnung aber nicht unser Ziel ist, sind beide für das System ausreichend.

```
<?xml version="1.0" standalone=no>
<scene name="rodi1" usecolors="0" usetextures="1" tiling="1000.0" min_y="-10.0f">
  <layer name="wasser1" r="0.5" g="0.5" b="0.5"
         texture="../textures/terrain/sand01.jpg" endheight="85.0"/>
  <layer name="grass" r="0.5" g="0.5" b="0.5"
         texture="../textures/terrain/grass02.jpg" endheight="400.0"/>
  <layer name="rock1" r="0.5" g="0.5" b="0.5"
         texture="../textures/terrain/rock01.jpg" endheight="800.0"/>
  <layer name="rock2" r="1.0" g="1.0" b="1.0"
         texture="../textures/terrain/rock04.jpg" endheight="1300.0"/>
  <layer name="snow" r="1.0" g="1.0" b="1.0"
         texture="../textures/terrain/snow05.jpg" endheight="1500.0"/>
</scene>
```

Abbildung 12.5: Beispiel einer XML-Konfigurationsdatei für die Landschaftsschichten in der Szene

In unserem Fall haben wir uns für die erste Möglichkeit - Schichtzuweisung über die Höhe - entschieden. Die Modellierer haben dabei ähnlich wie bei den Wasserebenen die Möglichkeit, über eine XML-Datei globale und szenenspezifische Vorgaben über die benutzten Texturschichten zu machen (siehe Abbildung 12.5). Wir bieten sowohl die Zuweisung von Texturen wie auch Farben für die einzelnen Schichten (oder auch die Kombination von beidem). Eine Schicht ist jeweils über eine Anfangs- und End-Höhe bestimmt, in der sie aktiv ist: so muss nicht für jede Höhe jede Schicht dargestellt werden, sondern nur im entsprechenden Bereich, was die Geschwindigkeit der Darstellung erhöht.

Natürlich fehlt nun noch ein Konzept für die Überblendung mehrerer Schichten, da ein abrupter Übergang sehr störend wirkt. Wir haben daher einen Algorithmus implementiert, der jeweils aufeinanderfolgende Schichten mit den Möglichkeiten normaler 3D-Hardwarekarten überblendet.

Dabei benutzen wir sogenanntes Multipass-Rendering, d.h. das selbe Objekt wird mehrmals gerendert und die Ergebnisse jeweils mit den vorigen kombiniert. Die Rendering-APIs erlauben dabei üblicherweise Standard-Rechenoperationen wie Multiplikation, Addition und Subtraktion der Ergebnisse, was wir uns hier zunutze machen. So wird für jeden Punkt im Modell für jede Schicht ein Gewicht berechnet und die Schicht später beim Rendern mit diesem Gewicht dargestellt. Wichtig ist hier natürlich, daß die Summe aller Gewichte für den Punkt immer 1 ergibt, damit letzlich eine einheitliche Helligkeit vorhanden ist.

Das realisieren wir mit Multipass-Rendering, indem wir das Gewicht als Alpha(Transparenz)-Wert des Punktes betrachten: wenn wir nun alle Texturen nacheinander mit dem entsprechenden Transparenzwert addieren, so ergibt sich schließlich genau das gewünschte Ergebnis einer gewichteten Kombination der Texturen. Wie sich dies auf die Renderläufe (Passes) verteilt, ist in Abbildung 12.6 zu sehen: In einem Pass wird jeweils der Texturwert mit dem Alphawert multipliziert und das Ergebnis auf das vorige Ergebnis (vor dem ersten Lauf: schwarzes Bild) addiert. Dies kann natürlich für beliebig viele Schichten geschehen, allerdings wird mit jeder Schicht die Geschwindigkeit geringer, weil für jede Schicht das Modell einmal komplett gerendert werden muss. Wie man sieht, ergibt sich als Ergebnis eine Art Shading-Baum für die Berechnung (insofern ähnlich zum allgemeinen Shading-Baum in [**?**]). Um auch noch ein allgemeines Zusammenblenden mit Farbwerten zu erlauben, kann (falls in der Konfiguration aktiviert) vor die Textur-Passes auch noch ein voriger Pass gesetzt werden, der das Terrain nur mit interpolierten Farben rendert.

Die Alpha-Werte für jeden Punkt werden auch jeweils dynamisch berechnet: dazu wird für jede Schicht eine eindimensionale Alpha-Textur generiert, die jeweils als Blending-Wert für die entsprechende Höhe angibt. Jeder Punkt erhält dafür eine zusätzliche Texturkoordinate, die seiner Höhe entspricht, und mit der dann beim Rendern auf die Alpha-Textur zugegriffen wird. Wie schon oben erwähnt, ist es wichtig, daß zu jedem Punkt die Summe der Alphawerte exakt 1 ergibt. Dies realisieren wir, indem wir für jede Schicht den Mittelpunkt des Bereichs nehmen und mit steigender Abweichung des Höhenwertes

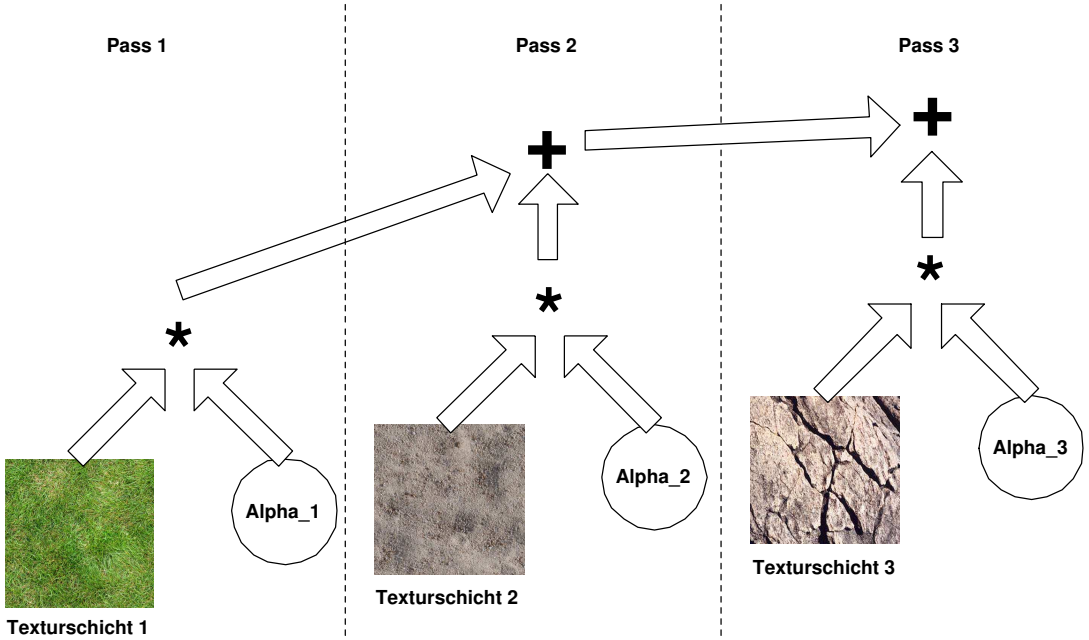


Abbildung 12.6: Rendern der Texturschichten mit mehreren Passes

von diesem Mittelpunkt die Gewichtung absinken lassen. Als Funktion bietet sich hier cos^2 an, weil dann die Gewichtungen benachbarter Schichten immer so gewählt werden können, daß die Summe 1 ergibt[1]. Zu sehen sind die entstehenden Gewichtungsfunktionen in Abbildung 12.7 zusammen mit den resultierenden Alpha-Texturen für die entsprechende Schicht.

Das fertige Ergebnis mit Überblendung ist schließlich in Abbildung 12.8 zu sehen.

[1]Das ergibt sich trivialerweise aus dem Satz $cos^2(x) + sin^2(x) = 1$ mit $sin(x) = cos(x - \pi/2)$

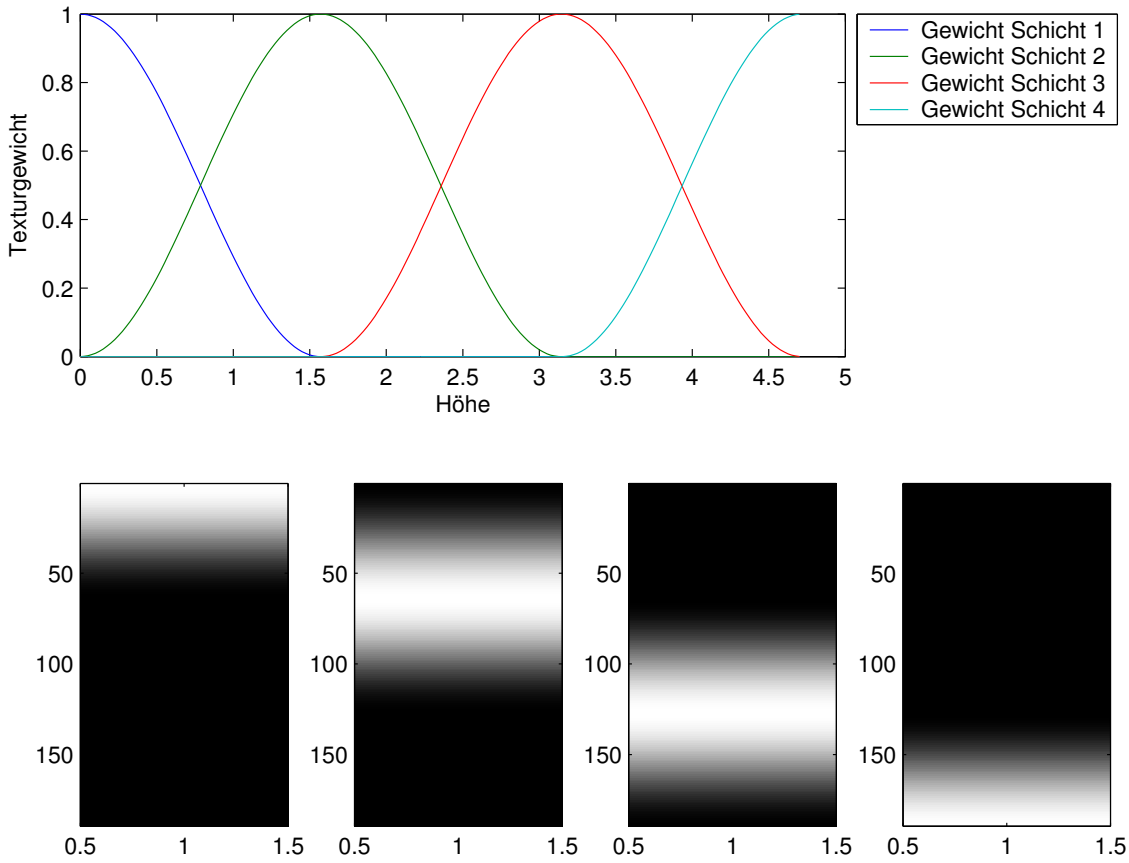


Abbildung 12.7: Gewichtung der einzelnen Texturschichten über cos^2: wie zu sehen ist, ist die Summe der Gewichte immer 1. Unten sind die entstandenen Alpha-Texturen für die einzelnen Schichten.

12.2 Physik-Simulation

Damit der Spieler in der Welt mit Objekten wie Bergen oder Kanonenkugeln kollidieren kann, wurde eine Physik-Simulation integriert, welche eine Simulation der physikalischen Welt zur Verfügung stellt.

Hier folgt eine kurze Beschreibung der Funktionsweise einer Physik-Simulation.

Grundprinzip einer Physik-Simulation Eine Physik-Simulation interagiert mit der Grafik-Engine über die Objekte in der Welt. Sie berechnet dabei die Position, Ausrichtung, Geschwindigkeit und Beschleunigung eines Objektes, während die Grafik-Engine dieses in der Welt darstellt. Dabei gibt es zwei Welten, einmal die Welt, die der Spieler sieht und die von der Grafikengine dargestellt wird. Die zweite Welt ist die, in der die Physik-Simulation abläuft, und eine geeignete Näherung der angezeigten Welt ist - in gewissen Fällen aber auch für den Spieler unsichtbare Objekte enthalten kann. Diese werden benutzt, um bei Kollision des Spielers mit diesem Objekt eine bestimmte Aktion in der Welt auszulösen. Dies sind dann die Trigger, auf die ein Skript reagieren kann und eine bestimmte Aktion auslöst. Wie oben erwähnt ist die Welt der Physik-Simulation nur eine Näherung an die sichtbare Welt, da die Berechnung bei vollem Detail viel mehr Rechenzeit verbrauchen würde als notwendig (insbesondere kann die Komplexität der 3D-Modelle geeignet reduziert werden).

Die Physik-Simulation simuliert die Welt nicht kontinuierlich, sondern tastet diese in gewissen Abständen ab, ebenso wie die Grafik-Engine etwa nur alle 40 Millisekunden ein neues Bild der Welt darstellt.

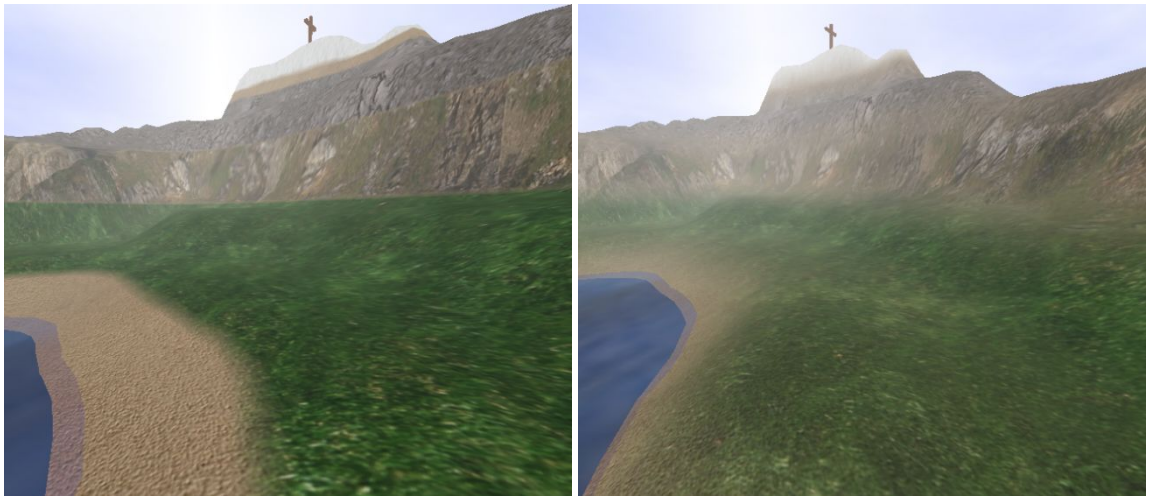


Abbildung 12.8: Das selbe Terrain-Modell, links gerendert ohne Überblendung zwischen den Schichten, rechts mit Überblendung wie im Text beschrieben

Jeder Abtast-Schritt beginnt damit, alle Objekte um den Betrag zu bewegen, den diese sich seit dem letzten Sample hätten bewegen müssen. Dann werden alle Objekte auf eine Kollision mit allen anderen Objekten überprüft. Wenn eine oder mehrere Kollisionen stattgefunden haben, wird allen Objekten, die kollidiert sind, eine neue Richtung und Geschwindigkeit gegeben, so dass diese wieder auseinander driften.

Jedes Objekt im Level hat 2 Repräsentationen. Die sichtbare ist meist ein **[Mesh??]** mit Textur. Die Physik-Simulation dagegen enthält für jedes Objekt eine Liste von Kollisions-Proxies[2], die das sichtbare Objekt geeignet annähern.

Kollisions-Proxies Die normalerweise in einer Dynamiksimulation vorhandenen Primitiven für Kollisions-Proxies sind Boxen, Kugeln, Zylinder und Dreiecksnetze (auch *Triangle*-**[Mesh??]** genannt). Die MiCarpet-Engine benutzt Boxen, Kugeln und Dreiecksnetze.

Dreiecksnetze werden normalerweise in der Micarpet-Engine nur für den Kollisions-Proxy des Terrains genutzt, da dieser Proxy-Typ sehr viel Rechenzeit benötigt. Das Terrain hat aber eine so unregelmässige Form, dass er nicht durch einen einfacheren Typ approximiert werden kann.

In Abbildung 12.9 ist eine Darstellung von einigen Objekten mit ihren dazugehörigen Kollisions-Proxies zu sehen (u.a. Dreiecksmesh für Terrain).

Manche Objekte bestehen aus mehreren primitiven Kollisions-Proxies. Als Beispiel ist hier in Abbildung 12.10 ein Baum zu sehen, dessen Kollisions-Körper aus 2 Boxen besteht. Wie offensichtlich ist, nähert dies die Struktur des Baums genauer an als ein Quader oder eine Kugel um das Gesamtmodell.

ODE

Um diese Schritte nicht selbst zu programmieren, benutzen wir das SDK *Open Dynamics Engine* (ODE,[**?**]). OpenDE oder ODE ist eine Open-Source-Bibliothek zur Simulation von 'Rigid Body'-Systemen (nicht-verformbare Körper). Dieses SDK nimmt uns im Grunde die drei oben genannten Schritte ab, musste allerdings noch in die Engine und das existierende Szenenmanagement integriert werden.

[2] *Proxy*, engl. für "Stellvertreter"

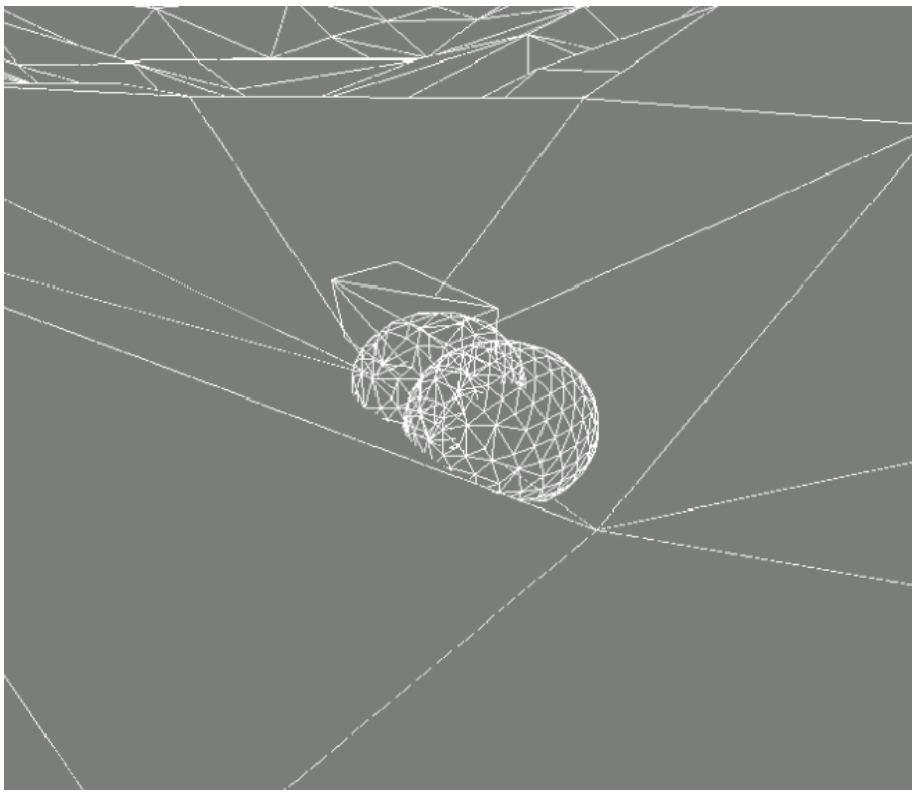


Abbildung 12.9: Darstellung Primitiver Kollisions-Proxies

ODE ist aufgebaut aus zwei Komponenten:

1. Die 'Rigid Body'-Simulation ist die erste Komponente. Diese erzeugt und löst das Gleichungssystem, für Kontakte zwischen Objekten innerhalb von Freiheitseinschränkungen. Das Gleichungssystem basiert dabei auf dem Energieerhaltungssatz in der Physik. Als Ergebnis dieser Berechnung wird dann für eine Kollision eine Kraft auf den entsprechenden Körpern ausgeübt, so dass diese sich dann gemäß dem Impulsgesetz verhalten. Weiter bietet ODE dabei an, Verbindungen zwischen Objekten zu simulieren. Diese Verbindungen führen Einschränkungen in den Bewegungsmöglichkeiten der verbundenen Objekte ein. Dabei werden als Primitive Verbindungstypen Gelenke, Schaniere und ähnliches angeboten.

2. Die Kollisionserkennung ist die zweite Komponente von ODE. Diese bietet einige Primitive als Kollisions-Körper an, die ODE in jedem Schritt auf Kontakte überprüft. Als Primitive werden von ODE die Typen Kugel, Box, Ebenen und Dreiecks-Netze (Triangle-**[Mesh??]**es, Trimesh), was den oben beschriebenen Stellvertretern für die Objekte entspricht.

ODE Anbindung

Die MiCarpet-Engine benutzt aus ODE die Primitiven Typen Kugel, Box und Trimesh. Weiter wird die 'Rigid Body'-Simulation nur soweit genutzt, dass die Objektbewegung und die Kollisionslösung von ODE simuliert wird. Die Gelenke werden von der MiCarpet-Engine momentan nicht unterstützt.

Es gibt mehrere Klassen, die die Verbindung zwischen ODE, OGRE und unserer internen Weltdarstellung zur Verfügung stellen.

CPhyWorld Diese Klasse wird als Manager für alle Objekte, die an der Physik-Simulation teilnehmen,

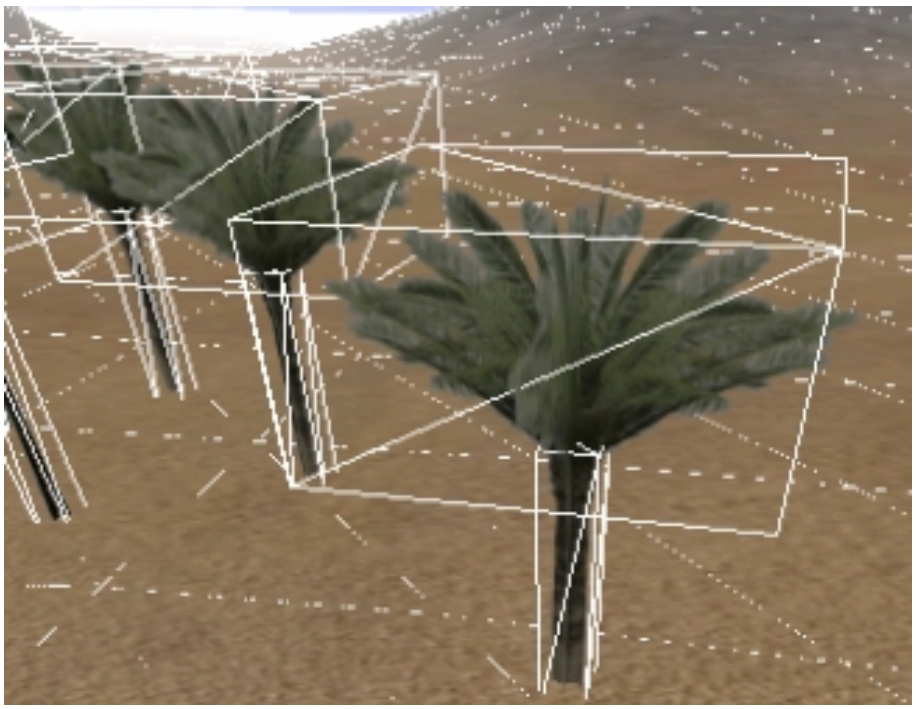


Abbildung 12.10: Darstellung der Kollisions-Proxies von Bäumen

verwendt. In jedem Level gibt es nur genau eine Instanz dieser Klasse.

CPhysicalObject Jedes Objekt, welches an der Physik-Simulation teilnehmen will, muss von dieser Klasse abgeleitet sein.

Eine abgeleitete Klasse kann die Methode `applyDynamics_()` überladen, um eine Synchronisation zwischen OGRE und ODE herzustellen. In dieser Methode lesen dann abgeleitete Klassen über `syncToOgreNode()` die Positions und Ausrichtungsdaten des ODE-Objektes aus und setzen diese im entsprechenden Knoten im Szenengraphen über OGRE.

Weiter kann eine abgeleitete Klasse die Methode `notifyCollided()` überladen, um ein spezielles Verhalten bei einer Kollision zu implementieren. Trigger können so über die Kollision zwischen ihrem Kollisions-Proxy und dem Spieler ausgelöst werden und dabei der Physik-Simulation mitteilen, dass diese Kollision nicht zu einem Auseinanderdriften der Objekte führen soll.

12.3 Eventsystem

Schon in frühzeitiger Planungsphase war klar, daß das System in der Lage seien sollte, diverse Ein- und Ausgabegeräte abseits Tastatur und Monitor zu unterstützen. Unter dieser Prämisse wurde ein **[PlugIn??]**-Interface gebaut, welches eben diese Bedingung erfüllt.

Schnittstelle Generell wird unterschieden zwischen Input- und Output-**[PlugIn??]**s, wo Input-**[PlugIn??]**s - wie der Name schon sagt - Eingaben annehmen und daraus eine Veränderung des Internzustandes der Welt produzieren und Output-**[PlugIn??]**s entsprechende Events verarbeiten, die auf Basis aktualisierter Weltdaten eine Reaktion für die Außenwelt produzieren (z.B. eine Kollision, die eine ForceFeedback Reaktion produziert); analog dazu wird ersteren das Interface `mica::IInput` und letzterem das Interface `mica::IOutput` bereitgestellt. Die jeweiligen **[PlugIn??]**s leiten dann von den entsprechenden Klassen ab und müssen `updateInput(..)` bzw. `updateOutput(..)` implementieren. Die Update-Methoden werden bei jedem Frame von der Engine aufgerufen und mit einem Pointer auf eine Eventliste sowie der Zeit seit dem letzten Update versehen. Die konkreten Implementationen der **[PlugIn??]**s füllen dann diese Liste mit Events, die von der Engine anschließend abgearbeitet werden.

Damit **[PlugIn??]**s überhaupt durch `updateInput(..)` bzw. `updateOutput(..)` aufgerufen werden können, müssen sie bei der Engine registriert werden. Dies wird von der Klasse `CExtModuleManager` geregelt, welche eben diese Funktion übernimmt und intern zwei Listen mit Referenzen auf die jeweiligen `IInput`- und `IOutput`-Objekte hält. Für jedes **[PlugIn??]** muß nach dessen Initalisierung beim Hochfahren der Engine ein `addExtObject(..)` auf dem Modul-Manager aufgerufen werden. Der Methode muß die Referenz auf die **[PlugIn??]**-Instanz, welche von `IInput` bzw. `IOutput` abgeleitet wurde, übergeben werden. Damit sind die **[PlugIn??]**s beim System registriert und werden bei jedem Update von der Engine entsprechend aufgerufen.

Events Die Eventliste vom Typ `tEventList` ist eine **[STL??]**-Liste und speichert die konkreten Events der verschiedenen **[PlugIn??]**s. Dabei ist jeder Event vom abstrakten Typ `CEvent`. Die Implementation der konkreten Klassen hängt dann von der Art des Events ab. Je nachdem was das **[PlugIn??]** in den Event an Nutzdaten schreiben will bzw. was für Informationen übertragen werden sollen, benötigt man verschiedene Implementationen. Dabei muß nicht jedes **[PlugIn??]** eine eigene Event-Implementation einführen; idR. reichen Standardmodelle wie `CEventAcc` für Beschleunigung oder `CEventOrientation` für Ausrichtung.

Jede Implementation muß einen eindeutigen Typ haben, wodurch jedes `CEvent` Objekt anhand seiner ID bzgl. seiner Implementation unterschieden werden kann. Die folgende Liste enthält die verschiedenen ID bzw. Typen von Events auf:

eEVENT_KEY Tastatur-Event

eEVENT_ACC Beschleunigungs-Event

eEVENT_ORIENTATION Neue Ausrichtung direkt setzen

eEVENT_ORIENTATIONCH Änderung der Ausrichtung des Spielers

eEVENT_ORIENTATION_AXIS Setzen der Ausrichtung des Spielers über die Achsen

eEVENT_REPOSITION Versetzt den Spieler an eine Position

eEVENT_RESETPLAYER Vetzt den Spieler auf die Startposition

eEVENT_COLLISION Kollision aufgetreten; Outputevent

Aktualisierung Die **[PlugIn??]**s werden bei jedem Frame aktualisiert. Diese Aktualisierung findet von `updateScene(..)` aus statt. Zuerst aktualisiert der Modul-Manager alle Input-**[PlugIn??]**s und somit wird für jedes **[PlugIn??]** die `updateInput(..)` Methode aufgerufen. Nach der Rückkehr des Modul-Managers liegt eine mehr oder weniger gefüllte Liste mit Input-Events vor, die nun als Eingabedaten für die Welt verarbeitet werden müssen.

Dies geschieht in `updateWorld(..)` welches u.a. auf jedem dynamischen Objekt ein `updateObject(..)` mit der Eventliste als Parameter aufruft. Die Events sind (auf die Objekte bezogen) in der aktuellen Form der Engine im Wesentlichen nur für das Player-Objekt interessant.

Nachdem die Welt mit ihren Objekten in einen neuen Zustand gesetzt wurde, kann ein Update der Output-**[PlugIn??]**s auf Basis dieser Weltdaten erfolgen. Dies geschieht analog zu den Input-**[PlugIn??]**s, indem der Modul-Manager für jedes Output-**[PlugIn??]** ein `updateOutput(..)` aufruft. Als Parameter stehen somit die erzeugten Output-Events in den Parametern. Jedes **[PlugIn??]** verarbeitet die anstehenden Events entsprechend und produziert die angebrachte Ausgabereaktion.

Nach den beiden Updatevorgängen werden die jeweiligen Eventlisten noch komplett entleert und der Speicher der Events wieder freigegeben, um anschließend im nächsten Update wieder gefüllt zu werden.

12.4 Netzwerksystem

Wie schon in der Übersicht erwähnt wurde, ist das System fähig, mehrere Computer parallel in einem Netzwerk zu nutzen, um mehrere Ansichten der Szene verteilt auf mehreren Rechnern zu rendern. Das hat den Vorteil, dass jeder Rechner nur eine Standard-Grafikkarte mit einem Ausgang benötigt. Es wäre zwar auch theoretisch möglich, nur einen Rechner zu benutzen, das aber erfordert eine Ausstattung mit einem Grafikausgang für jede Projektionswand und ist somit nur sehr schlecht zu realisieren. Insbesondere wäre ein Rechner mit dem gleichzeitigen Rendern mehrerer Ansichten bei interaktiven Bildwiederholungsraten überfordert.

Da wir keine stereoskopische Ansicht im System nutzen (für die pro Wand auch zwei Projektoren oder ein Spezialprojektor mit extrem hoher Bildwiederholungsrate notwendig wäre), sind die Aufgaben für das Netzwerksystem, welches die verteilte Grafikausgabe koordiniert, folgende:

- Finden der Rechner im Netz, Initialisieren und Beenden einer Sitzung
- Synchronisation des Renderns aller Rechner auf Framebasis
- Synchronisation der Szene (Animation, Änderungen im Objektstatus)

Wir haben auf Basis des Netzwerkprotokolls TCP/IP**[TCP/IP??]** (das Standardprotokoll im Internet, siehe [**?**]) ein Netzwerksystem für die Engine entwickelt, welches diese Punkte erfüllt. Unser System setzt dabei auf dem Protokoll UDP auf, welches eine verbindungslose und schnelle Datenübertragung ermöglicht. Das übliche Protokoll TCP schied dabei aus, weil es verbindungsbasiert ist und so bei jedem Paket noch weitere Verbindungsinformationen (Datenbestätigungen, ...) übertragen werden müssen, was insgesamt die Latenzzeit erhöht. In unserem Fall ist aber die Latenzzeit entscheidender als die Verbindungssicherheit, da jede Verzögerung durch Datenübertragung die Reaktionszeit des Systems erhöht. Die durch die Verwendung von UDP fehlende Übertragungssicherheit (UDP garantiert keine Erkennung und Behebung bei Paketverlusten) fällt nicht ins Gewicht, da wir in einem lokalen Netzwerk arbeiten, in dem im Normalfall keine Verluste entstehen sollten. In den meisten Fällen kann außerdem das System einen solchen Ausfall ausgleichen.

Aufbau Das Protokoll baut auf dem Master/Slave-Prinzip auf, d.h. es gibt einen bestimmenden Rechner (Master) und n untergeordnete Clients (Slaves). Wie in Abbildung 10.2 auf Seite 71 schon zu sehen war, hat üblicherweise der Master auch alle Peripherie-Geräte und die Slaves haben nur jeweils einen Grafikausgang. Es wäre aber auch einfach möglich, dass der Master selbst keine Grafik ausgibt und nur die Slaves Projektoren ansteuern. Änderungen an der Szene und dem Status kann allerdings nur der Master hervorrufen, die Slaves selbst haben ansonsten keine Mitwirkungsmöglichkeiten.

Die Kommunikation im Netzwerksystem über UDP/TCP/IP kann auf drei verschiedene Arten erfolgen:

- **Unicast**. Entspricht jeweils einer 1:1 Verbindung. Der Master schickt also Pakete, die an alle Slaves gesendet werden sollen, nacheinander einzeln an alle Slaves, was natürlich gerade bei steigender Anzahl Slaves länger dauern kann. Dementsprechend skaliert diese Lösung schlecht mit einer steigenden Menge an Slaves, bei einer kleinen Menge von Slaves (2-3) ist die dadurch entstehende Verzögerung bei unseren Tests noch vernachlässigbar gewesen.
- **Broadcast**. Hier schickt der Master Pakete grundsätzlich an alle Rechner im Netzwerk, also auch alle, die gar nicht im System teilnehmen. Dadurch muss jedes Paket an die Slaves natürlich nur einmal geschickt werden, in einem größeren Netzwerk erhalten aber viele Rechner Pakete, die sie nichts angehen und es kann so zu einer Störung des Datenverkehrs kommen (insbesondere, weil

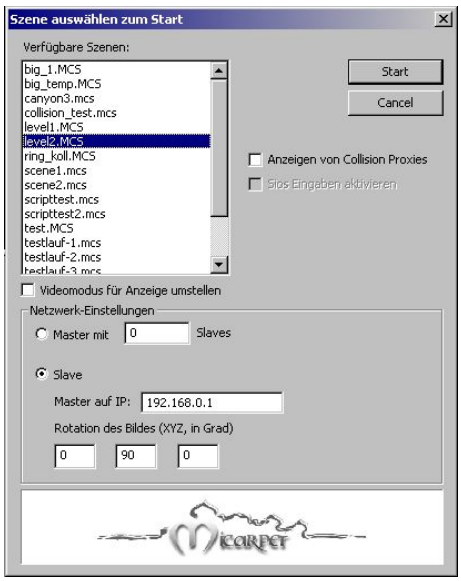


Abbildung 12.11: Verbindungsdialog für Netzwerksystem

das System sehr viele Meldungen pro Sekunde verschicken kann). In einem abgetrennten lokalen Netzwerk ist diese Lösung aber praktikabel.

- **Multicast**. Dies ist quasi die ideale Lösung, da mit Multicast Pakete verschickt werden können, die nur an eine bestimmte Gruppe im Netz geht. Die Empfänger müssen vorher ihre Gruppenzugehörigkeit über die Gruppenadresse (z.B. `224.0.0.106`) einstellen und empfangen dann alle Pakete, die an die Gruppe geschickt werden (und können auch an alle senden). Zwar ist Multicast im eigentlichen Internet noch nicht sehr verbreitet, da die meisten Router Multicast-Pakete noch nicht beachten, in einem lokalen Netzwerk auf Ethernet-Basis (welches eine Umsetzung von Multicast auf Ethernet-Gruppen unterstützt), ist dies aber eine sehr gute Lösung, bei der der Master auch bei größeren Mengen von Slaves nur ein Paket schicken muss.

Zu beachten ist, dass diese Unterscheidung fast nur den Verkehr von dem Master zu den Slaves betrifft, die Slaves selbst schicken mit Ausnahme des ersten Pakets (siehe nächsten Abschnitt) nur Daten direkt an den Master und kennen auch die anderen Slaves nicht.

Paketformat Das Paketformat ist ein sehr einfach aufgebautes Binärformat. Pakete bestehen dabei aus einem 4 Byte langen Header, wobei die ersten 3 Bytes nur zur Identifikation des Protokolls dienen und das vierte Byte den Typ des Paketes angibt (siehe Abbildung 12.12). Optional können nach diesem Header noch Nutzdaten vorhanden sein; ob das der Fall ist, entscheidet der Typ. Folgende Tabelle listet in Kurzform alle vorgesehenen Pakettypen auf:

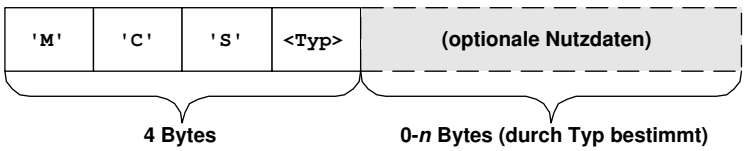


Abbildung 12.12: Paketaufbau im Netzwerksystem

Name	Zweck	Nutzdaten?
SLAVEBROADCAST	Slave sucht Masterserver	Nein
MASTERANNOUNCE	Master gibt Präsenz und eigene Adresse bekannt	Nein
JOIN	Slave will Sitzung beitreten	Nein
JOINACK	Master bestätigt Beitritt und sendet Szenendaten	Ja
FRAMEUPDATE	Master sendet Updatedaten für neuen Frame	Ja
FRAMEUPDATELAST	Master sendet Updatedaten / letztes Update für diesen Frame	Ja
FRAMEFINISH	Slave meldet, dass Frame fertig gerendert	Nein
NEWLEVEL	Master meldet Szenenwechsel	Ja
NEWLEVEL_ACK	Slaves bestätigen Szenenwechsel	Nein
SHUTDOWN	Master beendet Sitzung	Nein

Ablauf einer Sitzung Generell besteht eine Sitzung aus zwei Phasen: einer Startphase, in der der Master auf Slaves wartet, und einer Laufphase, in der das Rendern in einer Schleife abläuft.

Eine Sitzung beginnt damit, dass der Master gestartet wird. Dieser startet lokal mit den vorgegebenen Einstellungen (u.a. erwartete Anzahl der Slaves), öffnet aber sonst nur einen Verbindungskanal (Port) und wartet auf eingehende Verbindungswünsche. Startende Slaves müssen nun als erstes den Masterserver ausfindig machen. Bei Unicast muss dazu zwingend vorher direkt die IP-Adresse bekannt sein (einstellbar beim Start), bei Broadcast schickt der Slave einfach das Discover-Paket an alle Rechner, bei Multicast an die Gruppe. Falls ein Master in der Startphase ist, antwortet er darauf mit einem Announce-Paket an den Slave, in dem er seine eigene Adresse bekannt gibt. Der Slave kann nun eine Anfrage zum Beitritt (Join) stellen. Hat der Master noch freie Plätze, so schickt er ein Join-Acknowledgement als Antwort, in dem unter anderem die Szene angegeben ist, welche geladen werden soll.

Hat der Master alle erwarteten Slaves gefunden, so beginnt er die Render-Schleife, die für jeden zu rendernden Grafik-Frame durchlaufen wird. Am Anfang jedes Frames sendet er ein oder mehrere Update-Pakete (sind die Slaves noch in der Startphase, so gelangen sie mit dem ersten Update-Paket in die Laufphase) an alle Slaves. In diesen Paketen sind die Änderungen an den Objekten in der Welt (siehe nächsten Abschnitt) kodiert. Da diese Informationen beliebig umfangreich werden können (je nach Anzahl der Änderungen), werden die Informationen eventuell auf mehrere Pakete aufgeteilt, da UDP eine maximale Paketlänge hat. In diesem Fall zeigt das Paket mit dem Typ FRAMEUPDATELAST an, dass dieses Paket das letzte für diesen Frame ist. Nachdem die Slaves alle Update-Informationen haben, wenden sie diese auf ihre lokale Kopie der Szene an. Danach starten sie das Rendern der Szene mit den aktuellen Einstellungen; sind sie damit fertig, sendet jeder Slave ein Finish-Paket an den Master, um anzuzeigen, daß er bereit zum Umschalten auf das neu gerenderte Bild ist. Hat der Master die Bestätigungen von allen Slaves erhalten, ist dieser Frame beendet und der Master sendet das nächste Update-Paket und so fort. Das nächste Updatepaket dient damit auch als Signal für die Slaves, das gerade fertiggestellte Bild bei sich sichtbar zu schalten.

Zum Beenden der Sitzung kann der Master schließlich ein Shutdown-Paket schicken. Dies führt auch dazu, dass alle Slaves die Ausführung der Engine beenden und herunterfahren.

Die Abbildung der Pakettypen auf die Klassenstruktur im System ist in Abbildung 12.13 zu sehen.

Synchronisation Wie schon erwähnt, werden mit jedem neuen Frame vom Master ein oder mehrere Update-Pakete generiert, in denen die Änderungen an der Welt im Vergleich zum vorigen Frame kodiert sind. Solche Unterschiede können etwa Änderungen der Position, der Orientierung, der Größe, aber auch an der Existenz eines Weltobjekts sein. Alle Objekte haben durch ihre Ableitung von `CWorldObjectBase` (siehe Abschnitt 12.1.3) die Möglichkeit, ihre variablen Eigenschaften über die Methoden `load/saveObjektDiff()` binär zu kodieren (insbesondere ist der Betrachter selbst auch ein Objekt, seine Bewegungen werden also über den selben Mechanismus erfasst). Vor jedem neuen Frame muss der Master also nur die Objekte in der Szene durchgehen und aus den individuellen Unterschieden der Objekte einen `CDiffStream` erstellen (Sammlung von binär kodierten Unterschieden). Dieser wird dann (falls nötig) auf einzelne Pakete aufgeteilt und über Update-Pakete verschickt. Die Slaves konstruieren daraus wieder einen `CDiffStream` und können über die entsprechenden Gegenfunktionen die Unterschiede auf die lokalen Weltobjekte anwenden, so dass die Szene nun auch den selben Zustand wie im Master hat.

Das Klassendiagramm der Hauptklassen für Master und Slaves ist in Abbildung 12.14 zu sehen.

12.5 Scriptsystem

Die Rolle des Scripting in einem System ist in erster Linie darauf zu reduzieren, die Funktionen, die das Interface bzw. die Engine liefert, auf einer tieferen Schicht abzubilden und ggf. neu zu strukturieren, um darauf - eingebettet in ein System - zur Laufzeit Einfluß zu nehmen. Der Hintergrund dieser Maßnahme besteht darin, daß man so die Scriptsprache als Werkzeug einsetzt, um durch die Menge der vorgesehenen Funktionen eine Ablauflogik zu erstellen. Somit entlastet man den Programm Code, da sonst zahlreiche Funktionen für bestimmte, teilweise einmalig gebrauchte Detaillösungen diesen überfluten würden. Dazu kann sobald das Interface zum Hauptprogramm definiert ist, die Entwicklung von Scripten praktisch unabhängig von der Entwicklung und auch noch in späteren Phasen erfolgen, da die Scripte idR. nicht kompiliert werden (und wenn dann unabhängig vom Hauptsystem). Die zur Verfügung gestellten Sprachmittel sollten sich dadurch auszeichnen, daß sie es ermöglichen, die typischen Probleme direkt zu lösen; also häufig eingestetzte Funktionalitäten und Abläufe so zu strukturieren, daß nicht lediglich die Funktionalität des Interfaces wiedergegeben wird, sondern diese Funktionen innerhalb des Sriptsystems zu neuen Lösungsmitteln für Standardprobleme zur Verfügung gestellt werden, ohne den Umweg über das komplette Programm Interface zu nehmen; also einer Steigerung des Abstraktions-Levels.

die Sprache LUA Die Sprache LUA [**?**] ist eine relativ populäre, dynamisch getypte Scriptsprache, die als *Freie Software* zur Verfügung steht und ein C-Programm Interface mit entsprechenden Bindung an ANSI C Code zur Verfügung stellt (siehe [**?**])I. Die LUA Virtual Machine interpretiert das entsprechende Script oder führt den zuvor kompilierten Bytecode aus. Die nötige Bindung an C++ Strukuren wird von der LUA Erweiterung luabind übernommen.
Binden bedeutet in diesem Fall, daß bestimmte Funktionen im C-Code LUA bzw. der zur Laufzeit aktiven Virtual Machine bekannt gemacht werden und aus LUA Scripten heraus diese Funktionen aufgerufen, angesprungen werden können und zum Script zurückgekehrt werden kann.

Funktionsabbildung Die Funktionsbindung zwischen LUA und dem System wird damit von luabind [**?**] übernommen. LUA bietet lediglich die Möglichkeit, C-Funktionen des Typs `static int` mit der Signatur `lua_State*` - einem Pointer auf einen LUA Stack - an die Virtual Machine zu binden. Die Grenzen liegen somit klar darin, daß man für jede einzelne Funktionalität eine eigene Funktion benötigt und man Parameter manuell auf den Stack legen und behandeln muß, daß sie fehlerrobust (bei Aufrufen aus LUA unterschiedlicher Signaturen) wieder entnommen werden.
Hier kommt luabind zum Einsatz welches eine Meta Struktur über Template Klassen anbietet um die Bindung von C++ Funktionen beliebiger Signatur, wie Rückgabetypen und Klassen zu ermöglichen. Bei Klassen können ebenso die Member (teilweise) gebunden, Vererbung der Stuktur zwischen C++ und LUA erfolgen. Dabei stellt luabind keine eigene Bindung an LUA her, sondern übersetzt die Bindung von C++ Strukturen über das Interface in LUA Bindungen, welche über die zusätzlichen Verwaltungs- und Kommunikationsstukturen verfügt, um LUA eine Klassenstruktur zu bieten.

Die Kernfunktionalität des Scriptsystem auf Engineseite ist in den Klassen `CScript` und `CScriptInterface` beschrieben. Ersteres ist für die Funktionalität des Scriptsystems bzw. der Virtual Machine, letzteres für die Kommunikation zwischen Engine und Scriptsystem zutändig. Beim Starten der Engine wird nach dem Hochfahren der Virtual Machine und dem Dazuladen diverser Libraries die Bindung der deklarierten Funktionen, Klassen und Member vorgenommen:

1. `luabind::def( “foo”, &lua_foo );`
2. `luabind::class_< CLuaObj >( “Obj” )`
3. `.def( “getType”, &CObj::getType )`

Mit 1. wird eine Funktion `lua_foo` aus dem Namensraum unter dem Namen `foo` bei der Virtual Machine registriert. Zur Laufzeit steht damit die Funktion in LUA zur Verfügung und wenn ein Script dann also `foo()` aufruft, wird die Funktion `lua_foo()` angesprungen und ausgeführt. Bei 2. kommt die Template Klasse `class_` zum Einsatz, welche ein `CLuaObj` in C++ als Objecttyp `Obj` unter LUA registriert. 3. ist eine Member-Funktion, die an 2. gebunden wird. Dazu gibt es noch viele Modifikationen für abstrakte, bzw. abgeleitete Klassen, Konstruktoren und ähnliches.

Skripte Um ein Script zur Laufzeit auszuführen, muß auf der `static` Klasse `CScript::exec("script.lua")` ausgeführt werden. Das Script wird dann von der Virtual Machine interpretiert.

Den Scripten, die konkrete Aktionen innehaben, sind eine Reihe von Hilfsfunktionen zur Seite gestellt. Die Definitionen dazu befinden sich im Unterverzeichnis `./header/`, die immer nach Funktionsgruppen zusammengefasst sind und über Funktionspointer in Tabellen verlinkt sind. Dies bedeutet, wenn man beispielsweise auf das Titel-Textfeld zum *Head-Up Display* zugreifen will, findet sich dies unter `HUD.title("Text")`.
So wie wir es hier in einem einfachen Beispiel Script, welches beim Laden eines Levels geladen werden könnte, sehen:

```
OBJ.getDynamic("level1"):setVisible( false )

ring_map = {"null","Ring1","Ring2","Ring3","Ring4","Ring5","won"}
ring_map_index = 1
ring_map_maxindex = table.getn(ring_map)

HUD.title("Level 1", 4000)
HUD.subtext("durchfliege alle Ringe", 4000)

OBJ.getDynamic("Ring1"):setVisible( false )
```

`OBJ.getDynamic("name")` liefert die Referenz auf ein `CDynamicObject` auf dem dann alle freigegebenen C++ Funktionen des Objekts wie hier `setVisible(..)` aufgerufen werden können. Direkt darunter sieht man mit der Tabelle eine von LUAs Standardstrukturen. Praktisch alle weiteren Strukturen und Erweiterungen sind über die Tabellenstruktur realisiert.

Es ist zu beachten, daß jedes Script augenblicklich zurückkehren muß, da die Abarbeitung blockierend erfolgt. Eine geplante Ablaufsteuerung kann nur über Trigger erfolgen.
Um mit Triggern bzw. Kollisionen allgemein relativ leicht arbeiten zu können, existiert das Script `additional-collisions.lua` bzw. die entsprechende Handler-Stuktur in `header/collisions.lua`. Hiermit lassen sich Kollisionen unter Objekten konkreten Typs durch die Funktionen `__add_(..)` und `__add_t(..)` mit Behandlungen belegen. Also durch den Aufruf

```
__add_("player", "coin1", __coin_hit_ )
```

legt man fest, daß bei einer Kollision zwischen den Objekten des Typ `player` und `coin1` die Funktion `__coin_hit_` aufgerufen wird. Dies wird für u.a. für interaktive Elemente wie das Piratenschiff, die einzusammelnden Münzen, Levelübergänge oder jedes Soundevent verwendet.

Grenzen und Probleme Grenzen liegen klar im blockierenden Charakter des Scriptsystems und somit lassen sich nur Scripte einsetzen, die sofort zurückkehren. Um die Funktionalität für nicht-blockierende Scripte zu gewährleisten, müßte die Engine *multithreadingfest* sein, so daß Änderungen durch laufende Scripte eines im seperaten Thread laufenden Scriptsystems nicht in inkonsistenten Internzuständen resultieren dürfen, oder alternativ ein komplett anderes eventbasierendes Modell über Transitionen von Zustandsmaschinen; wobei dann die Zustandsmaschine in der Engine implementiert sein muß.

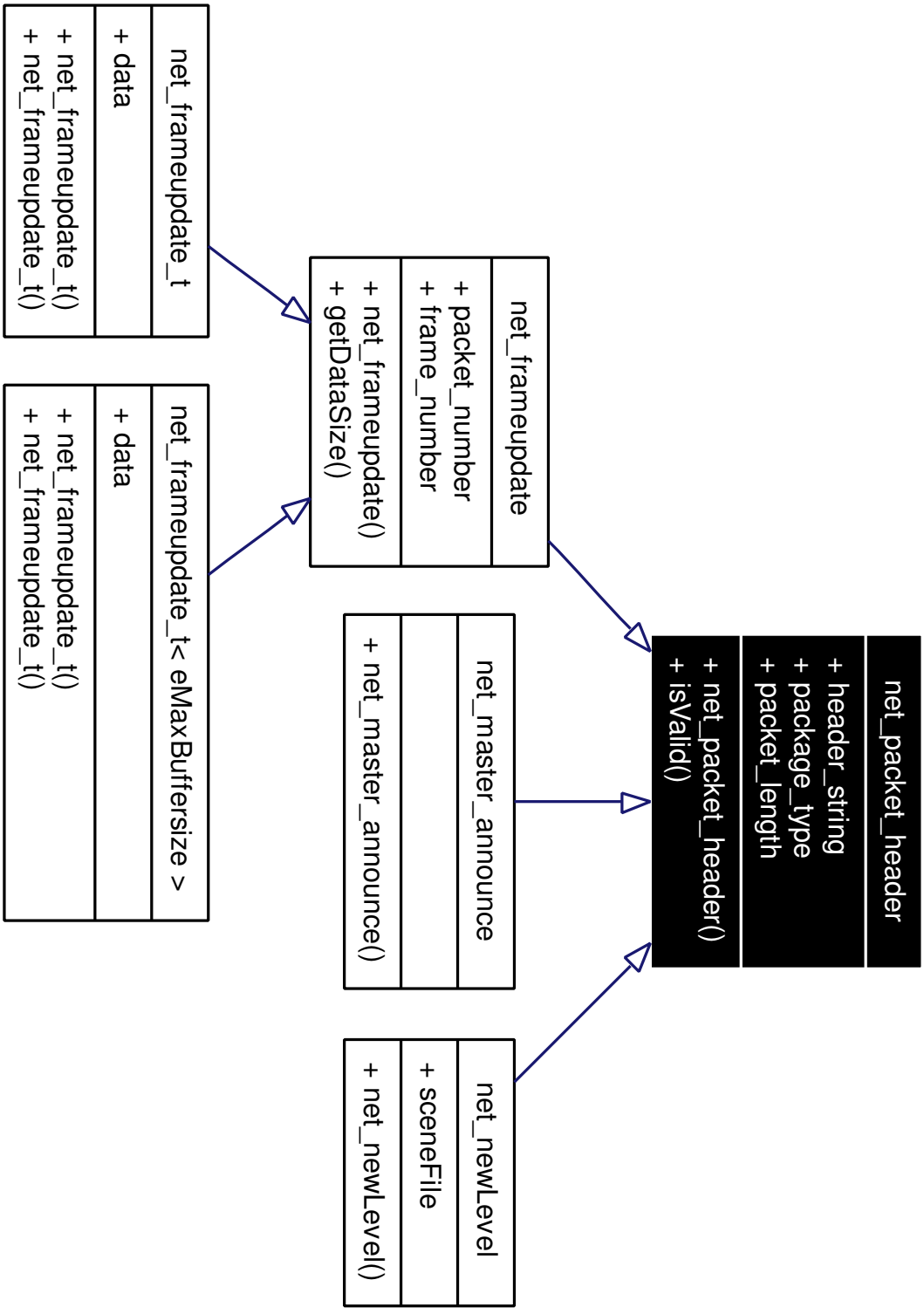


Abbildung 12.13: Klassenstruktur für die Netzwerk-Pakettypen

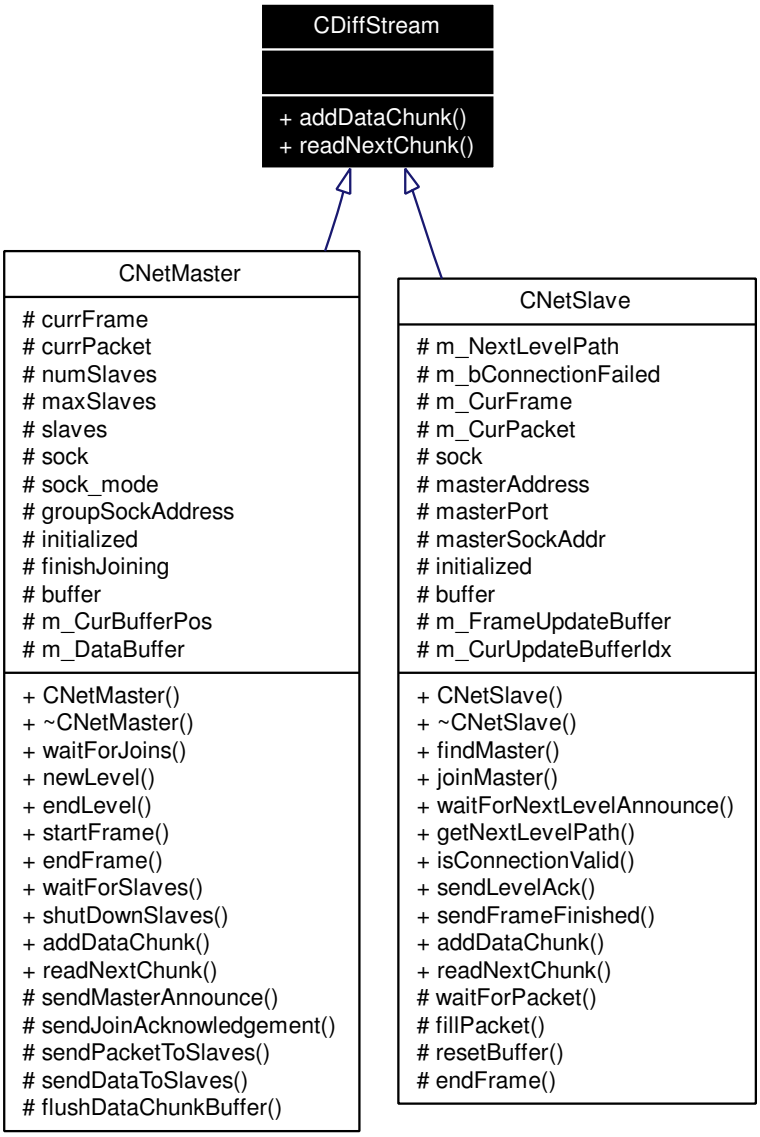


Abbildung 12.14: Klasse `CNetMaster` und `CNetSlave`, die die wesentliche Netzwerkfunktionalität implementieren

Kapitel 13

Grafik-Modellierung

“Der Hauptzweck der dreidimensionalen Computergrafik besteht in der Erstellung des zweidimensionalen Bildes einer Szene oder eines Objektes aus einer Beschreibung oder einem Modell des Objektes.“ [**?**] S.43

Die Aufgabe der Modellierung besteht also darin, Gegenständen eine auf dem Computer darstellbare Gestalt zu geben. Diese Gegenstände bilden den Teil des Systems, welche der Benutzer als Spielewelt erkennt.

13.1 Was ist ein Modell

Ein Modell ist die Beschreibung eines geometrischen Körpers auf Basis von Punkten im dreidimensionalen Raum. Bei den im MiCarpet-System verarbeiteten Modellen handelt es sich um polygonale Modelle (“Objekte werden durch ein Netz planarer **[Polygon??]**-Flächen angenähert. Mit diesem Modell können wir ein Objekt jeder Form mit beliebiger Genauigkeit darstellen.“ [**?**] S.45). Die Annäherung beschreibt hierbei runde Flächen jeglicher Art, wie sie z.B. bei Kugel und Zylindern auftreten. Neben den polygonen Modellen gibt es auch noch Modelle, welche durch andere Verfahren erstellt werden. Die einzelnen Verfahren werden unter dem Punkt 13.2 genauer beschrieben.

Wenn von einem Modell die Rede ist, dann fällt auch sehr schnell der Begriff **[Mesh??]** (das Netz), welcher die Darstellung des Objektes als Gitter kennzeichnet und somit Informationen über die Grösse und die Form des Objektes liefert. Bezüglich der Darstellung des Objektes sagt das Mesh aber noch ziemlich wenig aus. Es fehlen die Informationen darüber, wie die Oberfläche, gemeint ist damit der sichtbare Teil des Objektes, dargestellt wird. Diese Informationen werden im allgemeinen als **[Textur??]** bezeichnet. Sie bezeichnen Werte zur Farbe, Transparenz und über die Eigenschaften des Materials. Aspekte der **[Texturierung??]**, die ein gesteigertes Interesse verdienen sind unter Punkt 13.3 aufgeführt.

Ist ein Körper, bestehend aus einem Objekt und einer Beschreibung der Oberfläche, erstellt, fehlt noch die Information über ein mögliches Verhalten. Dieses beschreibt, wie sich das Mesh und die **[Textur??]**daten im Verlauf der Zeit zueinander oder in Bezug auf andere Objekte verhalten. Die Verhaltensänderung bzgl. der Zeit wird Animation genannt. Unter Punkt 13.4 wird die Animation anhand von Bones beschrieben. Ein Beispiel für eine Objektanimation wäre die Skalierung einer Kugel. Beim Stauchen eines Objektes handelt es sich zwar auch um eine Objektanimation, jedoch wird hier von einer eine geometrischen Transformation gesprochen. Eine Translation, Rotation oder positionsveränderndes Verhalten beschreiben hingegen die Position zu anderen Objekten in der 3D Welt und haben keine Änderungen der Objektgeometrie zur Folge.

Matrixnotationen In der Matrixnotation wird ein Punkt V durch Translation, Skalierung und Rotation wie folgt transformiert:

$V' = V + D$
$V' = S\,V$
$V' = R\,V,$

wobei D ein Translationsvektor ist, S und R Skalierungs- bzw. Rotationsvektoren.[**?**] S.17
Des Weiteren existieren auch Objekte, welche aus mehreren "Unterobjekten" zusammengesetzt sind. Hier müssen zusätzlich zu den Informationen der Unterobjekte ihre Lagebeziehungen gespeichert werden. Auch ein komplettes Szenario setzt sich so zusammen.

13.2 Verfahren der Modellerstellung

Neben dem oben angesprochenen polygonalen Modellen gibt es auch noch Modelle, denen andere Modellierungsmethoden zugrunde liegen. Einige der Methoden werden nachfolgend aufgelistet und kurz beschrieben:

- **NURBS (Non Uniform Rational Basic [Spline??])** sind Freiformflächen, mit deren Hilfe komplexe Objekte auf einfache Weise erstellt werden können. Im Gegensatz zu **[Polygon??]**en, entstehen bei NURBS keine »Polygonalen« Kanten. Die Detailstärke der Darstellung des Modells ändert sich dabei auch bei stärkster Vergrösserung nicht.

  Da NURBS nur Flächen darstellen, werden sie häufig mit der CSG (Constructive Solid Geometry) kombiniert, um Festkörper zu erzeugen. Durch die Kombination beider Verfahren sind Festkörper in allen Variationen möglich.

  NURBS werden vor allem zur Konstruktion von Fahrzeugen oder anderen technischen Dingen verwendet. Durch Abrundungs-Funktionen lassen sich so sehr schnell komplexe und realistische Körper erstellen.

- **CSG (Constructive Solid Geometry)** bezeichnet die Festkörpergeometrie. Über einfache Grundkörper, die durch Boolesche Operationen verschmolzen oder voneinander abgezogen werden, sind einfache Objektkonstruktionen möglich. Da nur einfache Grundkörper wie Würfel oder Kugeln zur Verfügung stehen, sind die Möglichkeiten sehr begrenzt. Um die Einschränkungen etwas aufzuheben, werden meist auch Rotationskörper oder ähnliches unterstützt.

- **Metaball** werden verwendet, um organisch anmutende Körper oder auch Flüssigkeiten zu generieren und zu simulieren. Der Name Metaballs kommt daher, dass Körper aus kugelförmigen Gebilden zusammengesetzt werden, die wie Wasser- oder Quecksilbertropfen zusammenschmelzen. Ein Metaball ist definiert als ein Punkt, der von einem (ungerichteten) Feld umgeben ist, das mit der Entfernung abnimmt.

  Mit Metaballs lassen sich schnell organische Körper und Oberflächen erzeugen. So kann z.B. eine menschliche Hand erzeugt werden, indem die Metaballs der Finger in Gruppen aufgeteilt werden, damit sie nicht verschmelzen. Allerdings ist es sehr schwer, präzise Formen und Objekte wie technische Gegenstände oder ähnliches zu modellieren.

Für die Modellierungs-Gruppe wurden neben den NURBS zwei Methoden der Modellerstellung verwendet, welche im folgenden Abschnitt genauer beschrieben werden.

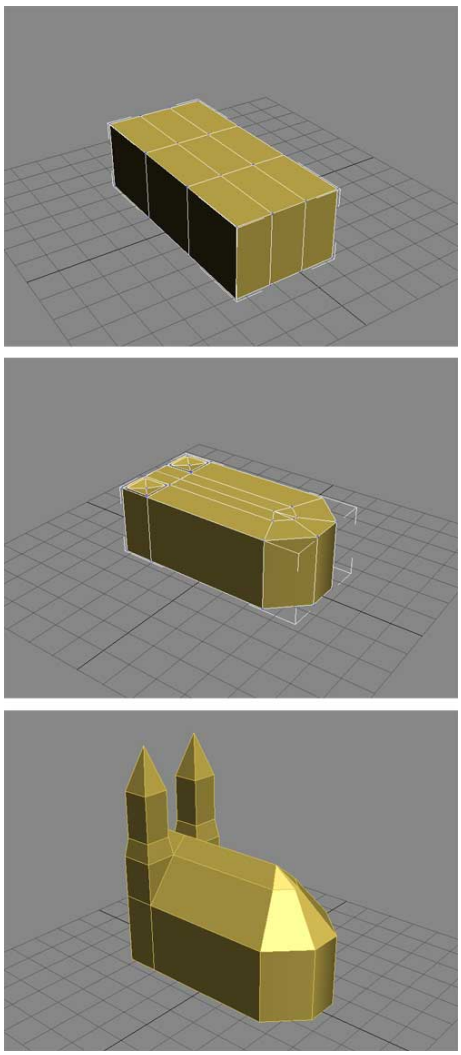


Abbildung 13.1: Low-Polygon-Modelling - Vom einfachen Quader zum fertigen Modell

13.2.1 Low-Polygon-Modelling

Da die meisten Grafikengines, die im Spielebereich eingesetzt werden, eine Einschränkung bzgl. der maximalen **[Polygon??]**anzahl einzelner Objekte vorgeben (siehe auch Punkt 13.9 auf S.125), um die Darstellung in Echtzeit zu gewährleisten, eignet sich das Low-**[Polygon??]**-Modelling am bestens, um Modelle mit einer geringen Anzahl an Vertices und **[Polygon??]**en zu erstellen.
Bei dieser Methode dient ein Quader als Ausgangspunkt für die weitere Entstehung des Objektes. Der Quader wird dabei bei der Erstellung mit frei gewählten Unterteilungen in allen 3 Ebenen definiert. Vertices und **[Polygon??]**e, welche durch die Aufteilung erzeugt wurden, stellen dann wiederum einzeln oder in Gruppen die Grundlage für weitere Transformationen dar. Durch das Erzeugen von neuen Vertices lässt sich die Komplexität und der Detailgrad des Objektes erhöhen.
Auf der Abbildung 13.1 ist ein Erstellungsprozess anhand von drei Zuständen dargestellt.

13.2.2 Spline-Modelling

Bei der Modellierung mit **[Spline??]**s dient eine algorithmisch berechnete Kurve - die **[Spline??]** - als Ausgangspunkt. Diese Kurve hat einen Start- und einen Endpunkt und wird durch einen oder mehrere

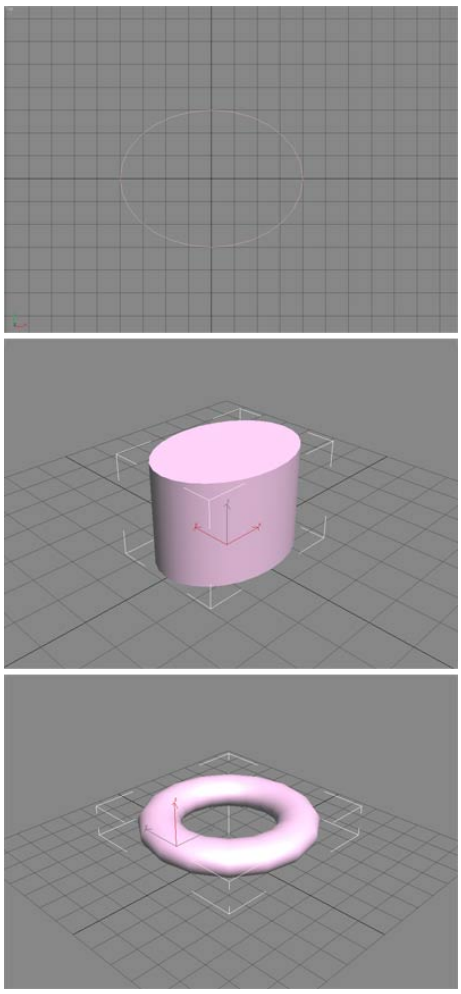


Abbildung 13.2: Modellierung mit **[Spline??]**s

Kontrollpunkte ergänzt. Soll aus einer **[Spline??]** ein Volumenkörper erstellt werden, so wird zuerst einmal eine Fläche - begrenzt durch die **[Spline??]** - erzeugt. Die Fläche kann dann entweder extrudiert oder um eine Achse gedreht werden, um einen Volumenkörper generieren zu lassen. Angenommen die **[Spline??]** beschreibt den Rand einer Ellipse, so bildet die Ellipse bei der Extrusion senkrecht zu der Ellipsenebene die Grundfläche eines Zylinders mit entsprechender Extrusionshöhe. Die Abbildung 13.2 zeigt ganz oben die Grundfläche (Ellipse) aus der dann ein Zylinder durch Extrusion (Bild in der Mitte) und ein Ring (unteres Bild) durch Rotation erzeugt wurden. Liegt bei der Drehung die Dreh-Achse innerhalb der Ebene, in der auch die Ellipse liegt und gibt es keinen gemeinsamen Schnittpunkt, so ist die Ellipse die Schnittfläche eines Ringes, welcher durch die Drehung entsteht (siehe 13.2 - unteres Bild).

13.3 Techniken der Texturierung

Versucht man Objekte mit einem realistischem Aussehen zu erzeugen, so braucht es eine gute Oberflächendarstellung, die möglichst realistisch erscheint. Dies wird grundlegend durch die verschiedenen Möglichkeiten der **[Texturierung??]** ermöglicht, welche die Materialeigenschaften eines Objektes beschreiben. Lichtquellen und Methoden zur Darstellung von Schatten bauen dann auf diesen Eigenschaften auf und geben dem Objekt ein plastisches Aussehen.

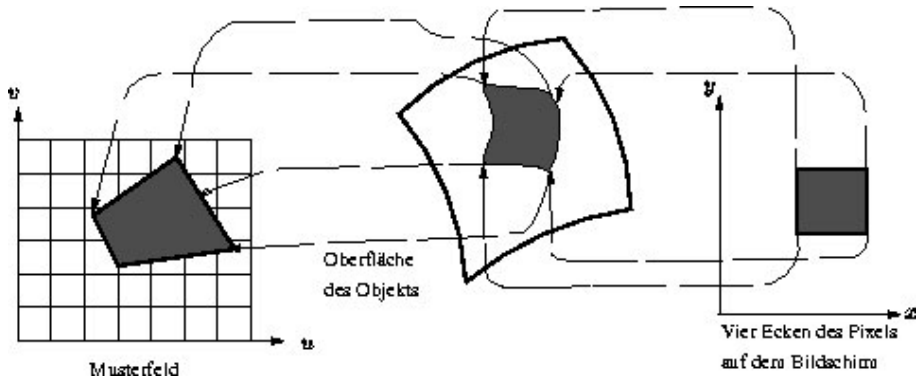


Abbildung 13.3: Transformationskette beim Texture Mapping

Nachfolgend werden einige Aspekte der **[Textur??]**ierung betrachtet:

- **Texture Mapping** Zur realistischen Gestaltung von Oberflächen verwendet man ein zweidimensionales Musterfeld (texture map), aus dem für jedes Pixel die zugehörige Farbe ermittelt werden kann. Zugrunde gelegt sind zwei Abbildungen:

  – Abbildung des Musters: Musterraum (u,v) -> Objektraum (x0,y0,z0)

  – Projektion: Objektraum (x0,y0,z0) -> Bildraum (x,y)

  Die Verknüpfung der zugehörigen inversen Transformationen liefert die zum Einfärben eines Pixels benötigte Information.

  Zum Einfärben eines Pixels im Bildraum wird die gewichtete Durchschnittsintensität aller überdeckten Pixel im Musterraum benutzt.

- **Bump Mapping** Wenn auf ein Objekt nicht nur ein farbiges/buntes Muster projiziert werden soll, sondern der Oberfläche auch eine Struktur “aufgeprägt“ werden soll, kommt das Bump Mapping zum Einsatz. Hierbei werden mit Hilfe von Störvektoren der glatten Oberfläche andere Reflexionseigenschaften verliehen, aus denen bei der Beleuchtung des Objekts eine Oberflächenstruktur ensteht, da die Berechnung der Helligkeit eines Punktes von dessen Oberflächennormalen abhängt (siehe Abbildung 13.4). Bump Mapping ist deshalb hervorragend geeignet, um eine Rinde von einem Baum oder ein faltiges Stück Leder zu simulieren. Ein mögliches Problem dabei sind die Ränder der Objekt- Silhouette. Sie erscheint glatt, da die Modellierung die Oberflächenstruktur nicht wirklich verändert. Durch die Normalenperturbation sollten daher keine allzu zerklüfteten Strukturen beschrieben werden. Diese sollten eigens modelliert werden. Bei der Erzeugung muß darauf geachtet werden, daß die Störungen immer an den selben Stellen auftreten, sonst würde sich die Oberflächenstruktur ändern, wenn sich das Objekt in einer animierten Sequenz befindet. Dies wird erreicht, indem das Ausmaß gespeichert wird, mit der die Normale in einem bestimmten Punkt ausgelenkt wird.

  In der Praxis kommen hauptsächlich die folgenden Verfahren zum Einsatz, auf die jetzt aber nicht weiter eingegangen werden soll.

  – Emboss Bump Mapping

  – Environment Bump Mapping

  – Dot Product Bump Mapping

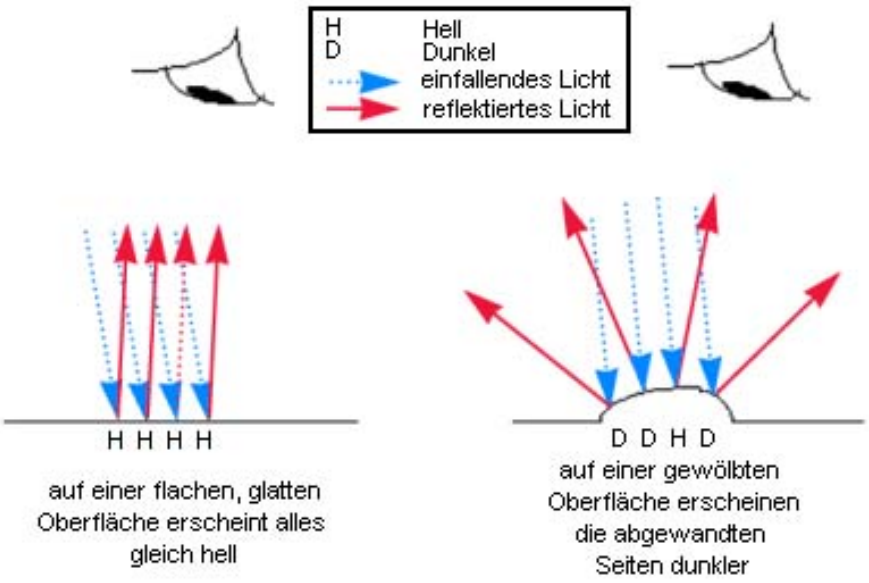


Abbildung 13.4: Reflexionseigenschaften beim Bump Mapping

13.4 Modell-Animation mit Bones

“Unter einem Bone-System versteht man eine gegliederte hierarchische Verbindung von Bone-Objekten, die zur Animation anderer Objekte oder Hierarchien verwendet werden kann.“ [**?**]

Bei einem solchem System werden zuerst die Boneobjekte erstellt. Diese sollten eine zweckgebundene Form besitzen und eine möglichst geringe Detailhöhe. Ein Bone ist ein längliches, spitz zulaufendes Objekt wie ein Kegel. Am spitzen Ende des Bones befindet sich sein Schwerpunkt, also der Punkt um den sich der Bone rotieren lässt. Innerhalb des Bonesystems werden sogenannte kinematische Ketten definiert. Diese können je nach belieben invers oder vorwärtsgerichtet verlaufen. Man bestimmt so die Abhängigkeiten der Bones-Objekte zueinander. Beispielsweise kann man nicht sein Bein heben, ohne das sich der Fuss vom Boden hebt. Man kann nun bei einer Animation einen Punkt einer solche kinematischen Kette bewegen und alle abhängigen Punkte werden der Bewegung folgen.

Danach werden die Vertices der Boneobjekte mit den relevanten Vertices referenziert. Dabei wird auch die Stärke der Relation definiert. So kann man anhand des Bone-Systems das tatsächliche Objekt steuern. Bone-Systeme haben ihre Anwendung bei dynamischen Objekten mit zusammenhängender Haut und definierbaren voneinander abhängigen Unterobjekten. Sie bieten eine sehr gute Möglichkeit zur Steuerung von Figurmodellen wie Tieren oder anderen Charakteren.

Leider hat das MiCarpet-System zwar die Fähigkeit erlangt solche Systeme zu verarbeiten, aber nicht die Möglichkeit, sie in einem automatisierten Prozess zu erstellen.

13.5 Fahrplan für das Projekt

Nachdem die Gruppe gebildet wurde (siehe Kapitel 9.5)), konnte diese im 3. Semester des Projektes die Arbeit aufnehmen. Zu Beginn ging es für die Mitglieder erst einmal darum, sich mit der gesamten Materie vertraut zu machen und sich darin einzuarbeiten. Durch die begleitende Lehrveranstaltung “Grafische Datenverarbeitung“ bei Prof. Frieder Nake waren die Grundlagen der Computergrafik vorhanden;

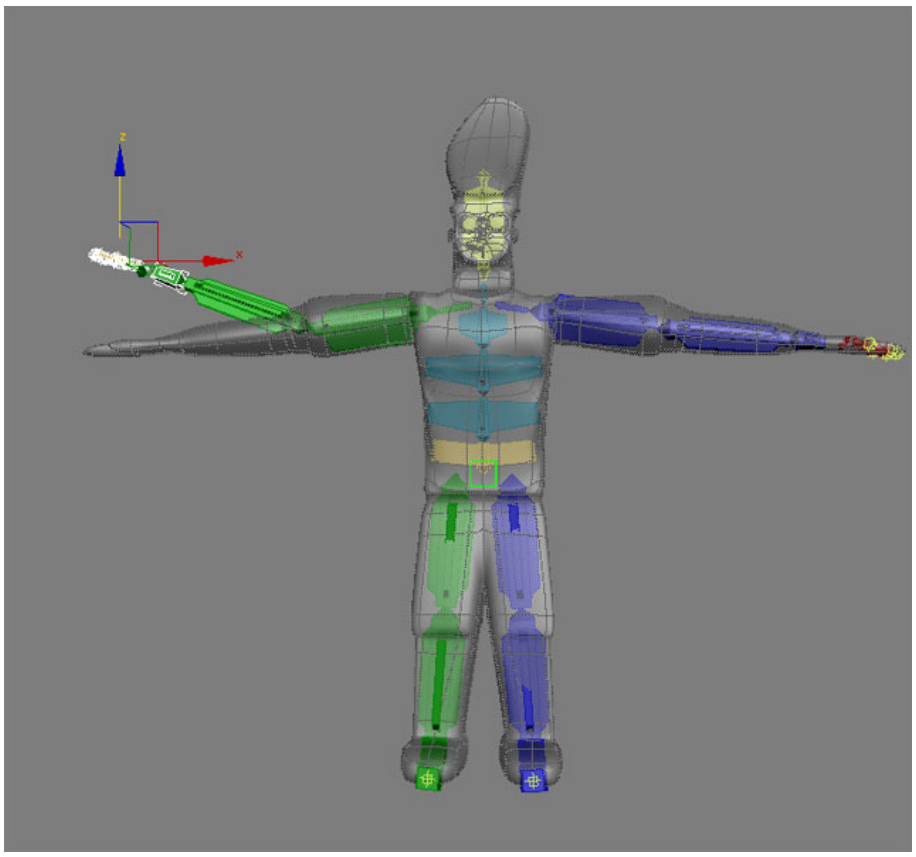


Abbildung 13.5: Charakter mit Bones

in Bezug auf die Modellierung von Objekten und Szenen für die Darstellung musste ein umfassenderes Wissen erst einmal erarbeitet werden. Dies geschah vornehmlich unter Zuhilfenahme der Software, eigenen Tutorials und der Recherche zu ganz speziellen Modellierungsthemen. Ein Schulungskurs innerhalb der Modellierungs-Gruppe diente dazu, die Teilnehmer mit der Software vertraut zu machen und den Stand des Wissen anzugleichen.

Um der Spielidee der Anwendung gerecht zu werden, wurden Überlegungen angestellt, welche Art von Objekten benötigt werden und wie die Erstellung zeitlich am besten eingeteilt werden könnte. Als Ergebnis dieser Überlegungen ist die folgende Roadmap entstanden:

1. Primitiven modellieren, wie Münzen, Ringe, Herzen, Schriftrollen, Palmen, Kakteen (28.11.2003)
2. Spielelemente abschließen, Landschaftselemente komplettieren und erste Entwurfszeichnungen (Handskizzen) des Piratenschiffs vorlegen (26.12.2003)
3. Piratenschiff zusammenstellen, einzelne Komponenten zusammenfügen (23.01.2004)
4. Weitere Landschaftselemente fertigen (06.02.2004)
5. Siedlungen, Ruinen, Oasen (und weitere komplexe Gebilde) zusammenstellen und Instanzen schaffen (20.02.2004)
6. Lebewesen modellieren (30.04.2004)
7. Einbinden der Modelle in die Welt

Die darin aufgeführten Objekte lassen sich in zwei Kategorien aufteilen. Die statischen und die dynamischen Objekte:

Statische Objekte

Statische Objekte besitzen keine Animationen und behalten eine fixe Position innerhalb der Szene.

Beispiele für statische Objekte sind:

- Bauwerke
  - Ruine
  - Haus
  - Beduinenzelt
  - Pyramide
  - Tempelruine
  - Windmühle
  - Siedlung
- Pflanzen (Flora)
  - Kakteen
  - Palmen
  - Büsche
- Spielelemente
  - Münzen
  - Ringe
  - Items
  - Schriftrollen
- Landschaftselemente
  - Oase
  - Fluss

Dynamische Objekte

Die dynamischen Objekten führen bestimmte Animationen aus, und/oder verändern ihre Position innerhalb der Szene.

- Lebewesen
  - Vögel
  - Kamel
- Spielfigur
- Piratenschiff

13.6 Komplexe Szenen

Für die Szenen, welche für die spätere Darstellung der Welt verwendet werden, müssen folgende Komponenten beachtet werden:

- das Terrain
- die Objekte
- die Anordnung der Objekte
- die Definition der Kollisionsobjekte

Die nachfolgenden Punkte beschreiben diese im Detail.

13.6.1 Erstellen der Terrains

Das Hauptmerkmal des Bodens liegt in seiner Grösse. Die Aufgabe besteht darin einen grossen unregelmässigen Körper zu Erstellen, der eine abgeschlossene Oberfläche, also keine Löcher hat. Zuerst versucht man die Art des Bodens zu charakterisieren: eine Insellandschaft hat beispielsweise starke Höhenunterschiede, aber im Gegensatz zu einer Berglandschaft einen beachtlichen Teil seiner Fläche unter der “Nulllinie“, also unter dem Wasser. Hat man diese Eigenschaften spezifiziert, gibt es verschiedene Möglichkeiten ein entsprechendes Modell zu erstellen. Es folgen die beiden Methoden, welche im Projekt MiCarpet benutzt worden sind.

Modellieren mit Worldbuilder

Der Worldbuilder ist eine Software, welche speziell dazu entwickelt wurde, Terrains zu erstellen. Hierzu werden die Höheninformationen anhand von Linien angegeben. (siehe 13.6). Die Punkte, welche die Linien beschreiben, sind die eigentlichen Daten. Es wird ein Mesh auf Basis dieser Punkte erstellt. Dieses gibt die Detailstärke des Mesh an und die Punkte des Mesh werden auf Basis dieser gesetzt. Dabei wird die Position des Vertex anhand der benachbarten Punkte durch Interpolation gemittelt. Dieses Mesh kann nun im Dateiformat „.3ds“ exportiert werden, so dass es in 3D Studio Max importiert und weiter verarbeitet werden kann. Sehr hilfreich ist die im Programm integrierte Bibliothek mit vordefinierten Informationen für Berge, Täler und andere Landschaftsstrukturen.

Modellieren mit Displacement-Mapping

Hierbei wird eine Fläche erstellt, deren **[Polygon??]**anzahl die spätere Detailhöhe des Terrains bestimmt. Ausserdem wird ein Bild - es werden verschieden Formate (.jpg, .bmp, .gif, ...) unterstützt - herangezogen. Dieses Bild enthält nur Informationen über Graustufen, welche die Höheninformationen darstellen. Es ist empfehlenswert ein den Seitenverhältnissen der Fläche angepasstes Bild zu verwenden, da ansonsten Fehler entstehen können, wenn das Bild in eine entsprechende Form skaliert wird. Nun wird die Fläche hergenommen und auf Basis der Informationen des Bildes verändert. Hierzu werden die Grauwertinformationen ausgelesen und als Werte zur Höhenverschiebung herangezogen. Dabei bilden Schwarz und Weiss die Extremwerte. Zusätzlich kann mit einem Parameter der Grad der Änderung angegeben werden. Die Abbildung 13.7 zeigt die Heightmap (oben) und das resultierende Modell (unten) nach Anwedung des Displacement Mapping.

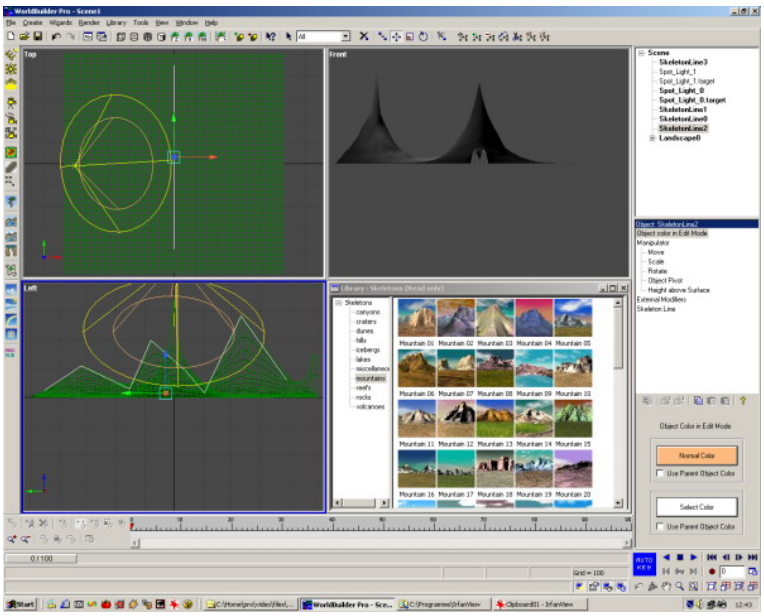


Abbildung 13.6: Modellierungsansicht im Worldbuilder

13.6.2 Objekterstellung

Die unter Punkt 13.5 beschriebenen Objekte wurden unter der Verwendung verschiedener Methoden (siehe Punkt 13.2.1 auf S.115 und Punkt 13.2.2 auf S.115) erstellt. Abbildung 13.8 zeigt das Modell des Piratenschiffes in 3D Studio Max.

Sobald der Volumenkörper eines Objektes vorlag, wurden passende **[Textur??]**en gesucht bzw. entworfen und dem Objekt zugewiesen. Für das Piratenschiff waren diese **[Textur??]**en oftmals Materialien, die das Aussehen von Holz wiedergeben sollten. Abbildung 13.9 zeigt eine **[Textur??]**, welche für die Visualisierung des Rumpfes verwendet wurde. Die Abbildung 13.10 zeigt das Modell des Piratenschiffs mit kompletter **[Textur??]**ierung in einer der Entwicklungsphasen des Objektes, welches später noch verfeinert und optimiert wurde. Die entgültige Version des Schiffes in einer der kompletten Szenen ist dann auf der Abbildung 13.11 zu sehen.

13.6.3 Anordnung der Objekte in einer Landschaft

Nachdem die Objekte entwickelt wurden, war der nächste Schritt, die Szene mit den Objekten zu füllen. Dank der Weiterentwicklung des Exporters war es zum Ende der Entwicklungszeit möglich, Objekttypen zu erstellen und zu exportieren. Dies hatte den Vorteil, dass wir bei der Anordnung der Objekte nicht mehr die Objekte direkt mit einbauen mussten, sondern einfach ein Referenzobjekt anlegen konnten, welches dann mittels abgestimmter Benennung auf den entsprechenden Objekttyp referenzierte. Im Vorfeld wurden also alle einzubindenden Objekte als Objekttypen exportiert und in einem zweiten Schritt wurde dann die Landschaft mit den entsprechenden Referenzen versehen, um so das Leveldesign umzusetzen. Die Abbildung 13.12 zeigt ein solches Landschaftsmodell mit den entsprechenden Referenzmodellen. Für das korrekte Einbinden bzw. die korrekte Auflösung der Referenzen ist die Engine zuständig. Mehr dazu unter Punkt 12.1.3 auf Seite 87.

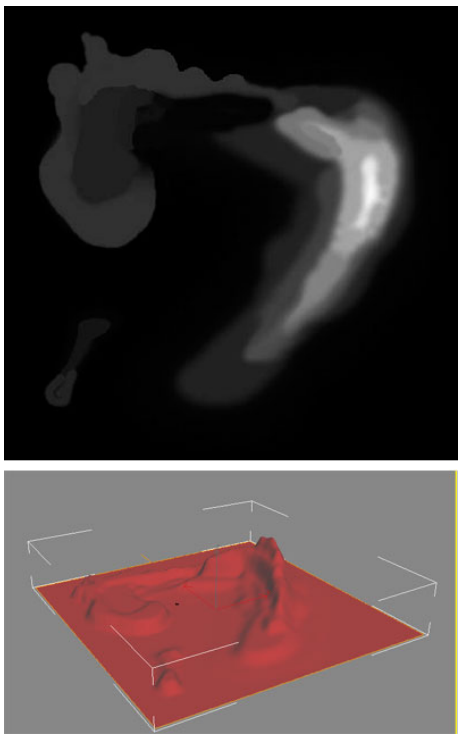


Abbildung 13.7: Heightmap und daraus generiertes Modell

13.6.4 Kollisionsobjekte definieren

Die Engine-Gruppe hat der Modellierungs-Gruppe eine einfache und elegante Möglchkeit bereit gestellt, wie die verschiedenen Objekte mit Kollisionsobjekten (siehe die Beschreibung der Physik-Simulation unter 12.2 auf Seite 87) versehen werden können um später dann zugehörige Events (Kollision, Einsammeln der Münzen, Begrenzung des Levels auszulösen. Die Engine OGRE unterstützt dabei drei Möglichkeiten ein Objekt mit einem Kollisionsobjekt zu belegen: eine Kugel, einen Quader oder eine Objekt mit gleichem Mesh. Aus Performancegründen können nicht alle Objekte mit der dritten Variante bestückt werden. Diese ist einzig und allein dem Landschaftsobjekt vorbehalten (siehe Punkt 12.1.4 auf Seite 93ff.) Bei dem MiCarpet System kamen überwiegend Quader für die Generierung von Kollisionsobjekten zum Einsatz. Auf der Abbildung 13.13 sind die Gitter der Kollisionsobjekte zu sehen. Deutlich wird hierbei der Unterschied zwischen dem Gitter, welches auf der Landschaft liegt und die Gitter um die Stämme und Kronen der Palmen. Einmal haben wir das Kollisionsobjekt als Mesh realisiert und einmal als Quader. Einer fertig modellierten Szenen wurden vor der finalen Version noch weitere Kollisionsobjekte eingefügt, die das Level für den Nutzer begrenzten. Diese Levelbegrenzung wurde mit Hilfe von sechs Quadern (Boxen) realisiert. Zwei Boxen für die Begrenzung der Höhe, also eine für den Himmel und eine für das Wasser, sowie vier weitere Boxen in den vier Himmelsrichtungen. Der dadurch entstehende Raum entsprach wiederum einem Quader und stellte den Raum dar, in dem sich der Spieler frei bewegen kann.

13.7 Software

Innerhalb der Modellierungsgruppe fand die folgende Software Verwendung.

- **Photoshop 6.0** Ein Bildbearbeitungsprogramm, welches unter anderem für die Erstellung von

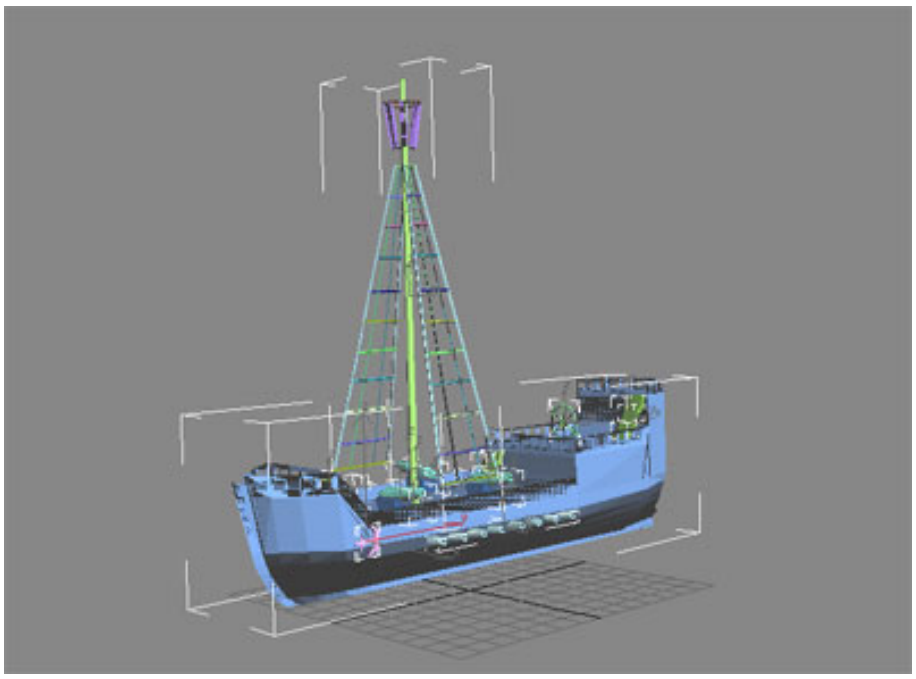


Abbildung 13.8: Das Modell des Piratenschiffes ohne **[Textur??]**en

Abbildung 13.9: Verwendete **[Textur??]** für den Rumpf des Piratenschiffes

Displacement-Maps und **[Textur??]**en zum Einsatz kam.

- **Wordbuilder 3.5** Software für die Erstellung von Landschaftsobjekten (Terrain). Siehe auch Punkt 13.6.1.
- **3D Studio Max 3.0** Ein Modellierungsprogramm, welches für das grundsätzliche Erstellen der einzelnen Modelle, sowie für den Aufbau der komplexen Szenen eingestzt wurde.
- **UltraEdit-32** Für das Editieren der XML-Dateien (siehe Szenen-Management (12.1.3) auf S.87)

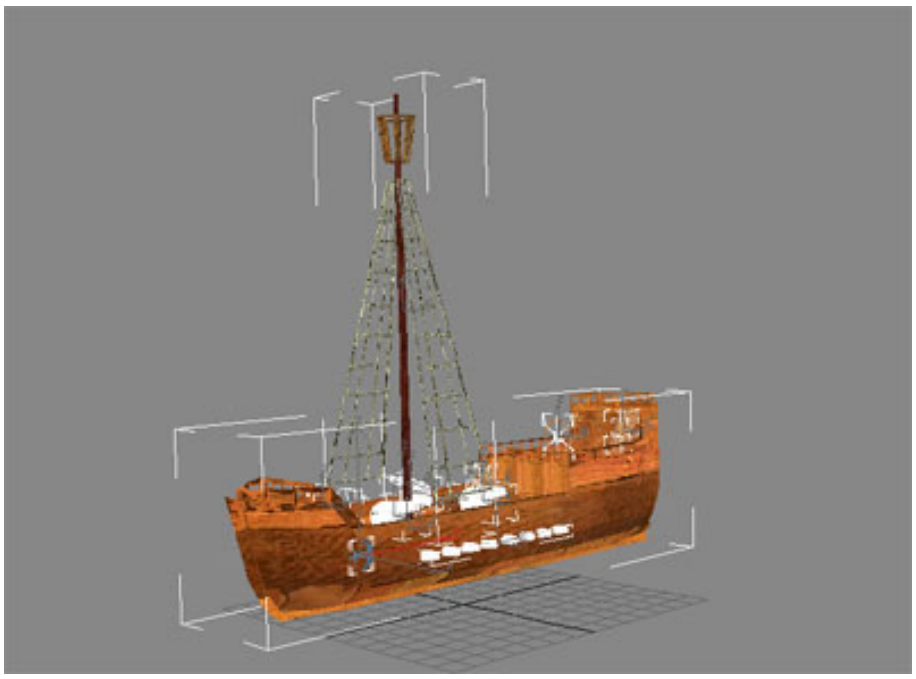


Abbildung 13.10: Das Modell des Piratenschiffes mit kompletter Textuierung

wurde der Editor Ultra Edit verwendet.

Software, die zum Einsatz gekommen ist, und nicht einer Open-Source-Lizenz unterlag (z.B. GPL) wurde mit Lizenzen der Mitglieder verwendet.

13.8 Exporter

Um die erstellten Modelle auch in die Gesamtapplikation sinnvoll mit einzubinden, müssen die Modelle in eine spezielle Umgebung - das Level - integriert werden, welches in Form einer XML-Datei (siehe auch 12.1.3) von der Engine eingelesen wird. Da das verwendete Modellierungsprogramm 3D Studio Max dies in der Standardversion jedoch nicht ermöglichte, wurde von der Engine-Gruppe ein entsprechender Exporter entwickelt, welcher diese Aufgaben übernahm. Durch die ständige Weiterentwicklung des Exporters, welche auf der Umsetzung von Wünschen der Modellierungs- und Engine-Gruppe begründet war, traten weitere Probleme auf, die im nächsten Kapitel erklärt werden.

13.9 Probleme

Die Probleme sind größtenteils durch das permanente Testen und Integrieren der entwickelten Objekte in die Applikation aufgetreten. Einige dieser Probleme wurden ausschließlich durch die Verwendung der Grafik-Engine OGRE (siehe auch 12.1.2 auf S.85ff) verursacht, andere ergaben sich durch deren Zusammenspiel mit dem Exporter.

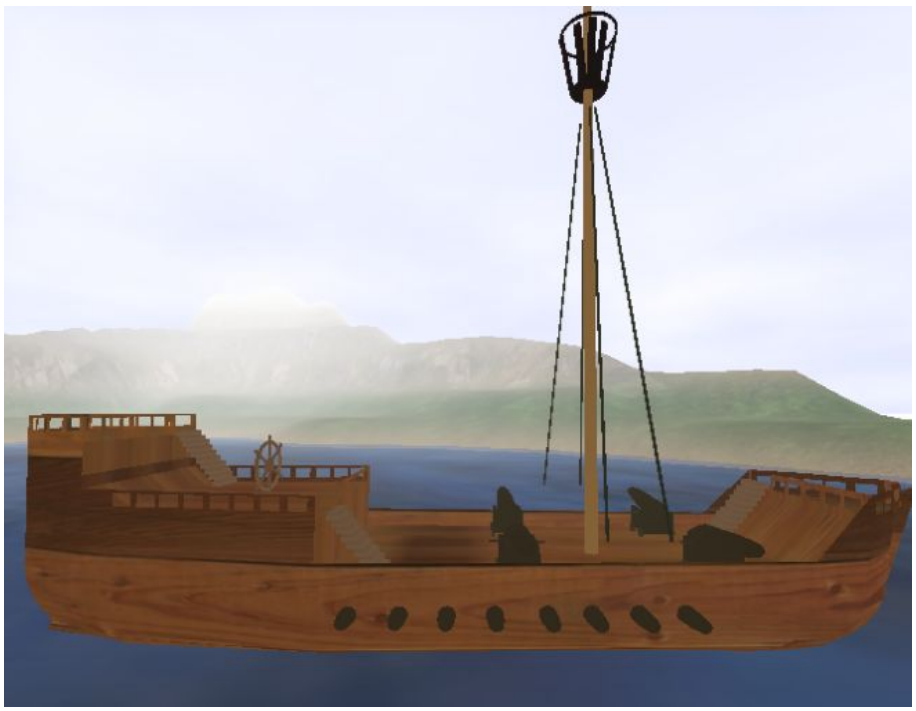


Abbildung 13.11: Visualisierte Version des Piratenschiffes in der Engine

13.9.1 Einschränkungen

Da die Engine (OGRE) nur Objektdateien mit maximal 32.000 Vertices verarbeiten konnte, mussten die einzelnen Objekte möglichst in mehrere Unterobjekte aufgeteilt werden oder aber von einem sehr hohen Detailierungsgrad Abstand genommenwerden, um die Anzahl der Punkte so gering wie möglich zu halten.

Für die **[Texturierung??]** der Objekte unter Zuhilfenahme des Exporters wurde nur die Verwendung einfacher **[Textur??]**schichten unterstützt. Verfahren, wie die Verbindung von **[Textur??]**schichten und Bump Mapping wurden aus Perfomancegründen nicht umgesetzt. Die Grafikengine unterstützte jedoch das Einbinden von mehrschichtigen und animierten **[Textur??]**en über die Verwendung von Materialskripten.

13.9.2 Optimierung

Durch die oben angesprochenen Einschränkungen war es nötig, die in einem ersten Prozess der Modellierung entstandenen Objekte zu optimieren, um somit die Anzahl der Vertices und **[Polygon??]**e zu reduzieren. Die entsprechenden Möglichkeiten mussten durch die Modellierungs-Gruppe erst erlernt und getestet werden, was gerade am Anfang zu einem der Arbeitschwerpunkte wurde. Zufriedenstellende Ergebnisse konnten durch die in 3D Studio Max integrierte *Optimierungsfunktion* und durch den Einsatz von unterschiedlichen Techniken beim Erstellen der Objekte (siehe Kapitel 13.6.1 auf S.121 und Punkt 13.6.1 auf S.121) erreicht werden.

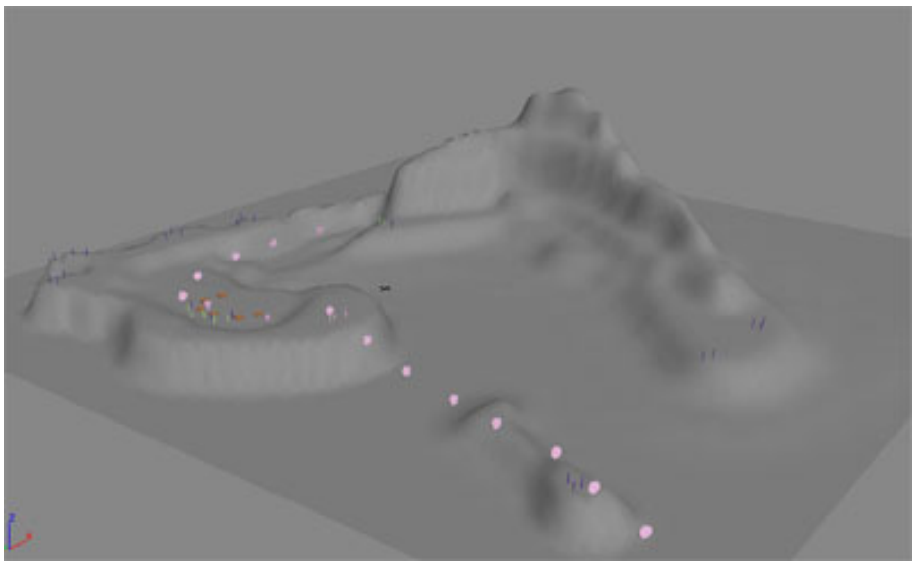


Abbildung 13.12: Landschaftmodell mit Refenzobjekten

13.9.3 Parallele Entwicklung

Durch die permanente Weiterentwicklung der Engine und der damit verbundenen neuen Anforderungen an den Exporter kam es zu einigen zeitlichen Verzögerungen beim Testen der Objekte und Szenen. So dienten die Ergebnisse des Exporters der Modellierungsgruppe dem Debugging und der Weiterentwicklung des Exporters durch die Engine-Gruppe. Eines der ersten Probleme, welches bei der parallelen Entwicklung des Exporters und der Szenarien auftrat, war die unterschiedliche Definition der Koordinatensystem von 3D Studio Max und OGRE. So waren die exportierten Objekte oftmals in mindestens einer der Achsen nicht korrekt in der Applikation wieder zu finden. Am Anfang musste sich damit beholfen werden, das Objekt vor dem Exportieren in 3D Studio Max einfach entlang der betroffenen Achse so zu drehen, sodass es später die gewünschte Lage in der Anwedung besaß. Duch ein entsprechendes Update des Exporters konnte dieses Problem vorerst beseitig werden, welches dann aber mit der Einführung der Unterstützung von Ober- und Unter-Objekten durch den Exporter erneut auftrat. Mehrere Updates des Exporters mit intensivem Feedback der Modellierungs-Gruppe konnten dieses Problem zeitnah lösen.

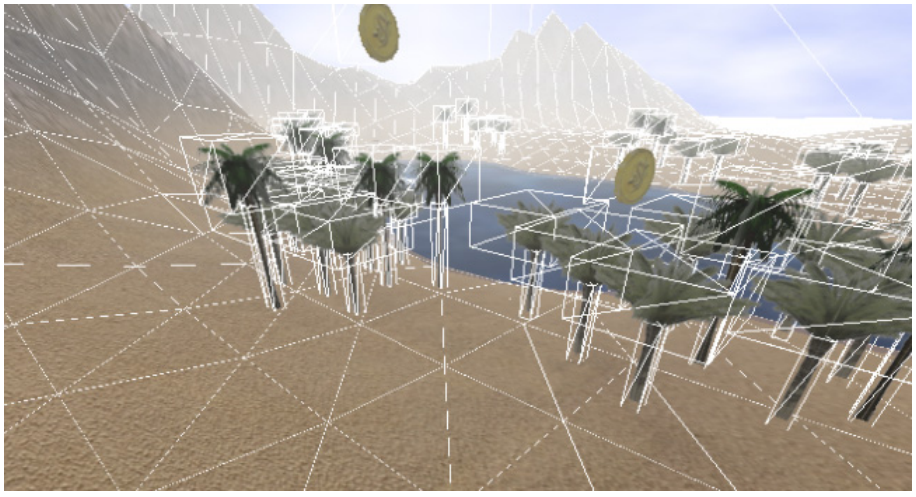


Abbildung 13.13: Szene mit angezeigten Kollisionsobjekte

Kapitel 14

Ein-/Ausgabegeräte

14.1 Motion Platform „Magic Carpet“

In diesem Kapitel wird die Arbeit der Gruppe „**[motionplatform??]**“ in ihren Arbeitsschritten bezüglich der hardwareseitigen Realisierung und in Hinsicht auf ihr Endprodukt sowie die Einbettung dessen in die gesamte Projektarbeit näher beleuchtet.

Die Gruppe bestand seit Ende des ersten Projektsemesters (WS02/03) und befasste sich zunächst mit den generellen Machbarkeitsüberlegungen einer **[bewegungsplattform??]** im Rahmen des Projekts. Als nächster Schritt wurde dann eine passive Plattform entwickelt, die erste sensorische Komponenten beinhaltete, bereits als Eingabegerät fungierte und bei einer ersten Präsentation verwendet wurde.
Die Anbindung an das Computersystem erfolgte dabei bereits über die Speicher-Programmierbare-Steuerung (SPS) „**[SIOS??]**“ der Firma *modul bus*, für die eine softwareseitige Lösung programmiert werden musste. Näheres hierzu wird im Kapitel „**[SIOS??]**“ erläutert.
Im Anschluss daran wurden Konzepte erarbeitet, die die Integration von aktorischen Komponenten vorsahen. Nach mehrmonatiger Planungsphase entstand so eine Konstruktion, die den Bedürfnissen der Anforderungsdefinition im Rahmen des Drei-Stufen-Plans gerecht wurde. Die dabei bis zum Projekttag Anfang Juli 2004 entstandene „**[motionplatform??]**“ dient nun als sensorisch/aktorische Hauptkomponente des Gesamtsystems.

14.1.1 Grund für diese Interfaceform / Vorabplanung

Zum Ende des ersten Projektsemesters wurde festgelegt, dass eine CAVE entwickelt werden sollte, die in Anlehnung an das bekannte Spiel „Magic Carpet“ eine Flugsimulation mit einem Teppich durch eine virtuelle Wüstenlandschaft vorsah.
In diesem Zusammenhang galt es zu erkunden, welche Interface-Form die geeignetste war, um dem Benutzer der CAVE ein möglichst realitätsnahes Fluggefühl/-erlebnis nahe zu bringen.

Erste Experimente mit Eingabegeräten gab es bereits zur Präsentation des „Rapid Prototyping“-Prototyps zum Projektbeginn. Dabei wurde beispielsweise eine „Tanzmatte“ gebaut, mittels derer allein man sich jedoch in der virtuellen Umgebung nur auf einer 2D-Ebene bewegen konnte. Zusätzliche Eingabegeräte wie beispielsweise ein Tracking-Handschuh standen zur Verfügung und sollten die Eingabemöglichkeiten des Benutzers erweitern.
Im Zusammenhang mit der Spiellogik machte diese Form der Eingabe nach dem Abschluss der Ideenfindungsphase keinen Sinn mehr, da, wie bereits oben erwähnt, eine Flugsimulation von den Mitgliedern des Projekts als CAVE-Anwendung favorisiert wurde. Die Tanzmatte als Zwischenlösung einer Eingabeform konnte und sollte im Rahmen dessen nicht weiterverwendet werden, da sie für eine Flugsimulation als ungeeignet angesehen wurde.
Eine weitere Überlegung stellte beispielsweise ein beweglicher Stuhl dar, der, erweitetert durch einen Joystick oder weitere Eingabegeräte, als sensorische und aktorische Komponente dienen sollte. Dabei sollte um den Stuhl herum ein Cockpit gebaut werden, dass im Rahmen einer moderneren Flugsimulation Flugdaten an den Benutzer weitergeben sollte. Diese Idee und viele weitere, die im Rahmen der Ideenfindung erdacht wurden, konnten und sollten nicht verwirklicht werden, da sie bei dem oben erwähnten Realisierungskonzept einer „Magic Carpet“-CAVE unpassend gewesen wären.

Aus diesen Grundüberlegungen heraus entstand die Idee, eine **[bewegungsplattform??]** zu konstruieren, die das Aussehen und das Fluggefühl eines fliegenden Teppichs realitätsnah simulierte. Zudem ermöglichte dieses Konzept, dass der Benutzer mit seiner Umgebung in Kontakt treten konnte, indem er beispielsweise mit seinen freien Händen trackbare Bewegungen vollführen konnte.

Sie richtete sich nach den Vorgaben, die im Rahmen der Plena gesetzt wurden. Es wurde überlegt, wie eine CAVE konstruiert werden könnte, die dem Benutzer möglichst realitätsnah aber kostengünstig das Gefühl vermittelt, auf einem fliegenden Teppich über eine reale Landschaft zu fliegen. Dabei ist man auf ein Konzept gekommen, das nicht vorsah, dass sich der Benutzer frei in der CAVE bewegen können sollte. Im Zuge dessen war es der „**[motionplatform??]**“-Gruppe möglich, eine **[bewegungsplattform??]** zu entwerfen, die einen verhältnismäßig großen Teil des CAVE-Innenraums einnehmen konnte.

Die **[bewegungsplattform??]**, die im Zuge erster Überlegungen bis zum ersten Projektwochenende in Verden erdacht wurde, sah wie folgt aus.

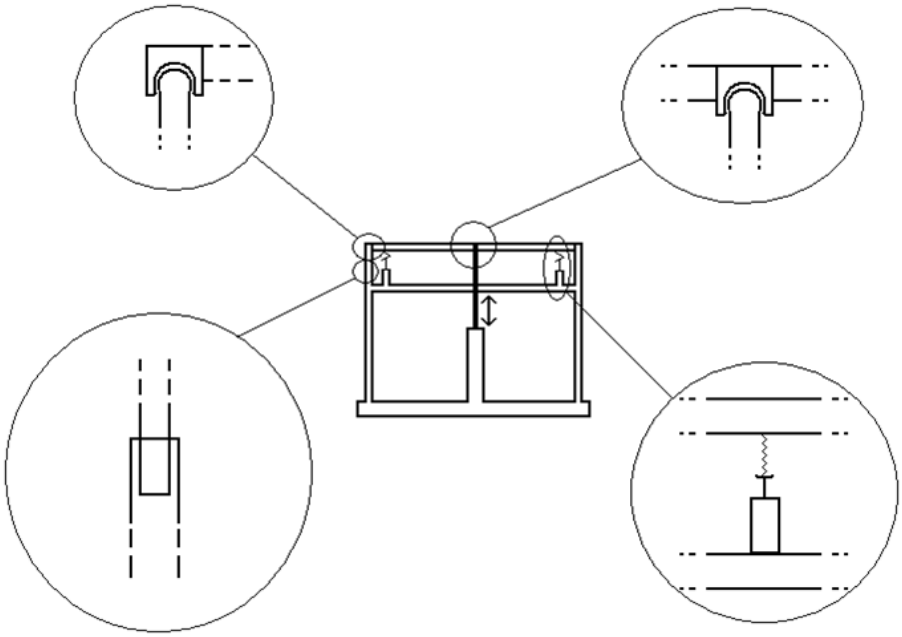


Abbildung 14.1: Erste Realisierungsüberlegung

Als Voraussetzungen für die Konstruktion der **[bewegungsplattform??]** wurden folgende Überlegungen getätigt. Zum einen sollte die Plattform ausreichenden Platz für eine Person bieten, da von der Spielidee her nur ein Benutzer für die Steuerung des Teppichs vorgesehen war. Des Weiteren sollte sie die Möglichkeit bieten, durch mehrere reale Bewegungsrichtungen dem Benutzer mehr virtuelle Interaktionsmöglichkeiten zu geben. Angedacht waren hierbei die Neigungsrichtungen vorne/hinten und rechts/links sowie das Heben und Senken. Dies stellt eine „**[motionplatform??]**“ mit drei **[freiheitsgrade??]**n („3-**[dof??]**“: „Pitch“, „Roll“, „Heave“) dar, mit deren Hilfe eine Steuerung des virtuellen Teppichs im dreidimensionalen Raum ermöglicht werden konnte.
Auf Grundlage dieser Voraussetzungen wurden unter anderem Überlegungen angestellt, aus welchen Werkstoffen und mit welchen Bauteilen die **[bewegungsplattform??]** konstruiert werden könnte. Anfangs war eine Metallkonstruktion geplant, da die entstehenden Kräfte so groß eingeschätzt wurden, dass eine Konstruktion aus Holz unpraktikabel erschien.
Zu diesem Zweck sollten neben Pneumatik-Zylindern und diversen anderen Pneumatik-Bauteilen auch Kugelgelenke und Federn verwendet werden. Zudem sollte die gesamte Unterkonstruktion und die Ständeraufbauten aus Metall erstellt und geschweißt werden.
Auf diese Weise sollte die **[bewegungsplattform??]** eine Tragkraft aufbieten, die für die Unterkonstruktion, die Platte, den Teppich und den Benutzer ausreichte. Das zu tragende Gesamtgewicht wurde auf 200 kg geschätzt.

Es wurden zwei konkrete Umsetzungsvarianten erarbeitet.

Variante A

Die erste Umsetzungsentwurfsvariante bestand aus fünf Steuer-Pneumatikzylindern, die durch eine SPS über einen PC angesteuert werden konnten. Dabei waren die Zylinder mit einem Wegemesssystem ausgestattet, das eine genaue Positionserkennung der Zylinder ermöglicht.

Abbildung 14.2: Zylinder mit Wegemesssystem (siehe[**?**])

Über die Plattenneigung durch Gewichtsverlagerung des Benutzers sollte die Steuerung des fliegenden Teppichs im Virtuellen erfolgen. Dabei sollten die Zylinder so angebracht werden, dass die obere Platte der Plattform frei auf den Zylindern gelagert war, sodass auf diese Weise die angestrebten drei **[freiheitsgrade??]** angefahren werden konnten. Im Urzustand sollten die Zylinder eingefahren sein. Nachdem der Benutzer die Plattform bestiegen hätte, würden die Zylinder so weit ausgefahren, dass durch die Druckluft-Kräfte die Plattform bei einem beliebigen Körpergewicht des Benutzers exakt auf der Hälfte des maximalen Hubs der Zylinder austariert wären.
Mit Hilfe einer Stütz-Ständerkonstruktion unterhalb der oberen Platte sollte ein seitliches Abkippen dieser verhindert werden. So sollte der Platte an senkrechten Führungsschienen entlang ein Lauf nach oben bzw. unten möglich sein. Außerdem sollte die gesamte Unterkonstruktion leicht erhöht zum Fussboden erstellt werden, sodass eine Projektion auf eine Fußbodenfläche möglich war. Alle anderen Komponenten sollten unterhalb des Cavebodens platziert werden.

Variante B

Die zweite Umsetzungsentwurfsvariante war der ersten sehr ähnlich. Sie bestand ebenfalls aus fünf Pneumatikzylindern, wobei einer davon als ForceFeedback-Zylinder genutzt und mittig unter der oberen Platte platziert werden sollte. Die anderen vier Zylinder waren Teil eines Zylinder-Feder-Verbundsystems, dass federseitig die benutzergewollte Neigung zulassen und zylinderseitig die systemgewollte Neigung ermöglichen sollte. Auch hier sollten alle Komponenten mittels einer SPS an den PC angeschlossen werden, um eine Verbindung zum Restsystem zu gewährleisten. Die Ermittlung der Neigung der oberen Platte sollte durch ein digitales Messverfahren erfolgen, da in dieser Variante kein Wegemesssystem zur Verfügung stand.
Bei dieser Variante hätte die Plattform nicht tariert werden müssen. Es war ein stabiler Urzustand vorgesehen, der dem Benutzer ein leichtes Besteigen dieser ermöglichen sollte. Hierzu sollten die Zylinder ausgefahren werden, um die Wirkung der an ihnen befestigten Federn zu neutralisieren. Nachdem der Benutzer auf der Plattform Platz genommen hatte, sollten die Zylinder einfahren, wodurch die Plattform nur noch auf den Federn und dem mittleren Zylinder gelagert war. Durch den mittleren Zylinder waren durch seine tragenden Eigenschaften ForceFeedback-Aktionen im Heben/Senken-Segment möglich.
Auch in dieser Variante war eine Stütz-Ständerkonstruktion wie in Variante A vorgesehen, die eine Projektion der Bodenfläche ermöglicht hätte.

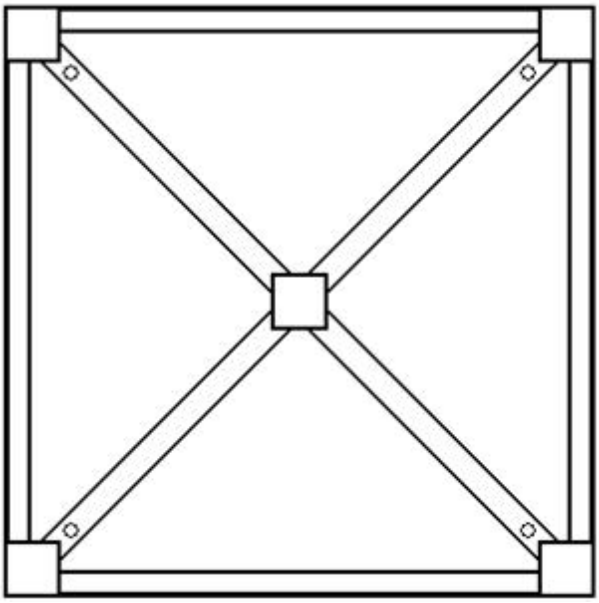


Abbildung 14.3: Unterkonstruktion

14.1.2 Konzepte / Benutzerinteraktion

Basierend auf den beiden Varianten, die als bloße Anfangsüberlegungen zu verstehen sind, wurden tiefergehende Recherchen angestellt, um sich Fachwissen anzueignen. Auf diese Weise wurden Konzepte zu Teilaspekten entwickelt beziehungsweise herangezogen, die es ermöglichten, insgesamt eine realisierbare, funktionstüchtige „**[motionplatform??]**“ zu planen. Gleichzeitig stand der **[lowcost??]** im Vordergrund der Planung, da das System neben einer möglichst hohen Portabilität und einem hohen Innovationsgrad auch möglichst geringe Kosten haben sollte.
Zunächst sei jedoch die Benutzerinteraktion in Bezug auf die eigenen Konzepte beleuchtet, die bisher im Raum standen. Wie oben bereits beschrieben, sollte es sich bei der „**[motionplatform??]**“ für das „Micarpet“-Projekt um ein Ein-/Ausgabegerät handeln, bei dem die sensorischen und aktorischen Komponenten eine Symbiose bilden.

Zum einen sollte die **[bewegungsplattform??]** als Eingabegerät fungieren, bei dem der Benutzer durch Verlagerung seines Körpergewichts die Neigung der Plattform bestimmen und auf diese Weise Einfluss auf den virtuellen fliegenden Teppich nehmen kann. Dabei sollte die Neigung stufenlos möglich sein, was bedeutet, dass die Plattform mit einem Dämpfungssystem versehen ist, dass auf die Gewichtsverlagerungen des Benutzers variabel reagiert und gleichzeitig auf diese Weise das Gefühl vermittelt, als reale Person auf einem realen, schwebenden Körper zu sitzen.
Zum anderen sollte die **[bewegungsplattform??]** als Ausgabegerät verwendet werden, dass auf dem Benutzer Kräfte auswirkt, die aus seinen Aktionen in der virtuellen Welt resultieren. So sollte beispielsweise die Kollision mit einem Körper in der Spielwelt zu einer tatsächlichen „ForceFeedback“-Ausgabe führen, die dem Benutzer ebenfalls einen wirklichkeitsnahen Eindruck vermitteln sollte.

Gestützt auf diese Vorgaben wurden Recherchen angestellt, um Wissen aus bereits vorhandenen Systemen mit denselben oder ähnlichen Eigenschaften zu akquiereren. Bei umfangreichen Internetsuchen und Sichtungen von wissenschaftlichen Materialien sind wir damals auf diverse Umsetzungsvarianten von „**[motionplatform??]**s“ gestoßen, die auszugsweise im Folgenden näher beleuchtet werden sollen.

Beispiel 1: Full Motion Flight Simulation Platform

Bei dieser Umsetzung (siehe [**?**]) handelt es sich um eine privat geplante und erstellte **[bewegungsplattform??]**, die gelagert auf sechs Pneumatik-Zylinder sechs **[freiheitsgrade??]** erreichen kann.
Bei der Umsetzung liegt das „**[steward??]**“ zu Grunde, bei dem zwei dreieckige Ebenen um 30° versetzt zueinander parallel angeordnet und mit in einer sich gegenseitig stützenden Dreieckform angeordneten Aktuatoren miteinander verbunden sind.

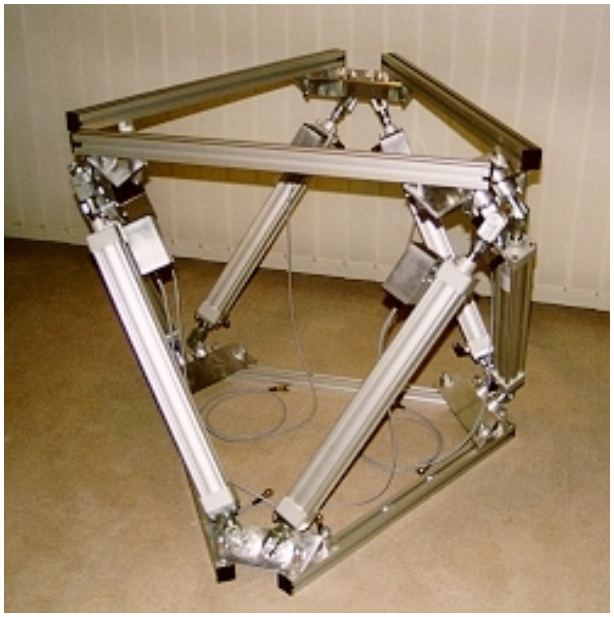


Abbildung 14.4: Plattform nach dem **[steward??]**

Auf Grund dieser Anordnung ist es erst möglich, sechs **[freiheitsgrade??]** zu erreichen. Bei diesen sechs **[freiheitsgrade??]**n handelt es sich um Bewegungsmöglichkeiten mit den englischen Bezeichnungen „pitch“, „roll“, „yaw“, „surge“, „heave“ und „sway“, die an dieser Stelle verwendet werden sollen, um Missverständnisse zu vermeiden, die durch deutsche Begriffe entstehen könnten. Diese Bewegungsmöglichkeiten einer „**[motionplatform??]**“ stellen deren Simulationsrealismus dar. Man kann sagen, dass je mehr **[freiheitsgrade??]** eine „**[motionplatform??]**“ besitzt, desto höher ist die Realitätsnähe der simulierten Kräfte. In der folgenden Skizze sind diese Bewegungsrichtungen und ihre Bedeutung näher gezeigt.

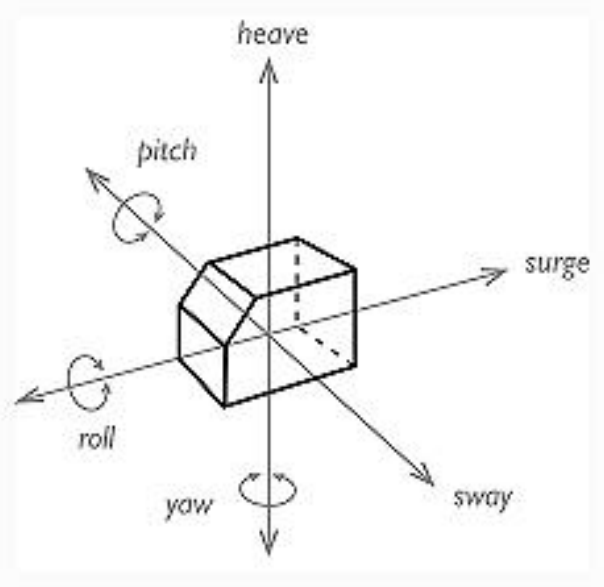


Abbildung 14.5: Schema der sechs möglichen **[freiheitsgrade??]** (vgl. [**?**])

Beispiel 2: „Integrating Dynamic Full-Body Motion Devices in Interactive 3D Entertainment“

Hierbei handelt es sich um ein wissenschaftliches Paper (siehe [**?**]), in dem eine Realisierung einer „**[motionplatform??]**“ in Form eines beweglichen Sitzes beschrieben wird. Hierbei kann der Sitz mittels Pneumatikzylindern in den Bewegungsrichtungen nach vorn und hinten sowie rechts und links gekippt werden (2-**[dof??]**).

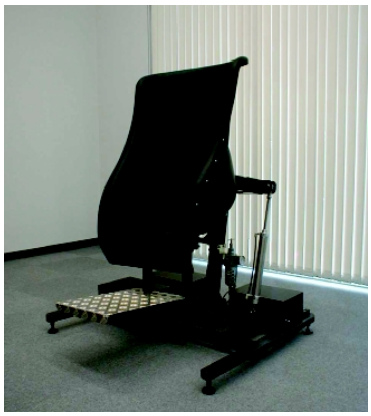


Abbildung 14.6: Dynamic Motion Chair

Ferner wird unter anderem in dem Paper auf die Möglichkeit der Ansteuerung der „**[motionplatform??]**“, deren Netzwerkfähigkeit und die „Multi-User“-Fähigkeit eingegangen.
Dabei ist beispielsweise die Ansteuerung der Plattform interessant, da sie auf einer komplexen pneumatischen Schaltlogik beruht, die es ermöglicht, mit verschiedenen Geschwindigkeiten/Kräften die Neigungen des Stuhls zu verändern.

Die Schaltung beruht hierbei auf zehn elektronischen Schaltventilen, die einzeln angesteuert werden können. Die Druckluftschaltung ist dabei so aufgebaut, dass für jeden Zylindereinlass ein Ventil für die Luftversorgung vom Kompressor zuständig ist, was den Zylinder mit Druck beaufschlagt. Die übrigen

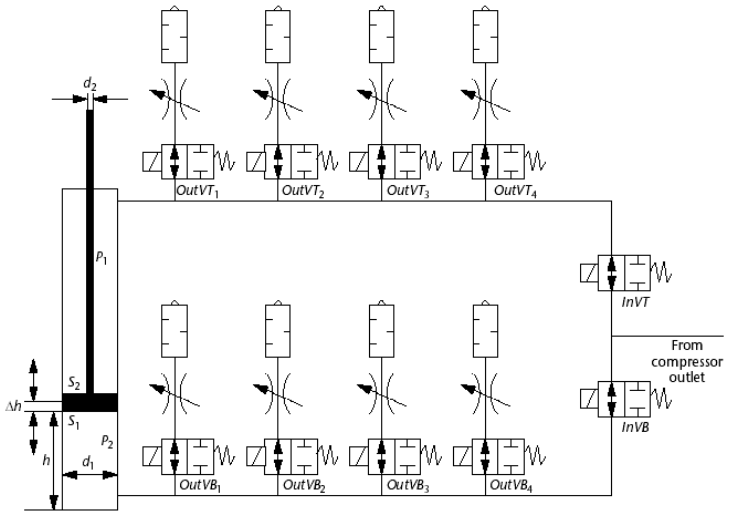


Abbildung 14.7: Pneumatischer Schaltplan

vier, parallel geschalteten Ventile bestimmen die Luftmenge, die kontrolliert aus der Druckluftschaltung entweichen kann. Durch Kombination der vier Ventile sind somit eine Vielzahl (2^4) Schaltungsmöglichkeiten gegeben.
Ergänzt wird der „Motion Chair“ durch einen „Joystick“, der die Steuerung der Spieleanwendung ermöglicht. Beide sind über „Interface Modules“ an ein „LAN“ angeschlossen, das die „Interfaces“ mit dem Restsystem verbindet.

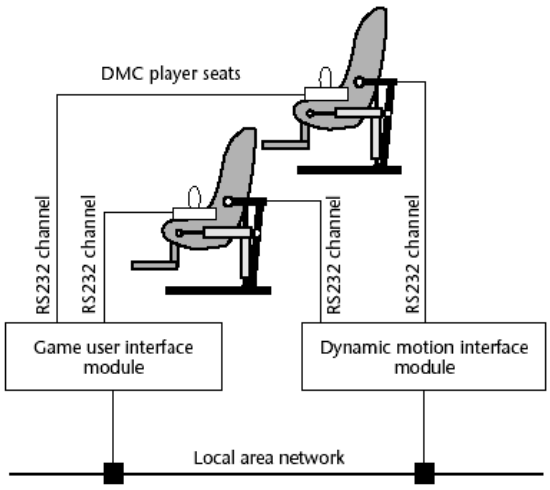


Abbildung 14.8: Anbindungsschema der „DMC Player Seats“

Die oben aufgeführten Konzepte stellen nur einen kleinen Teil der Vielfalt an bereits existierenden Realisierungen dar, die gesichtet wurden. Alle Konzepte sind komplex und beruhen auf pneumatischen, hydraulischen oder elektrischen Komponenten. Dennoch wurde durch das Projekt Micarpet ein eigenes Konzept erstellt.

Eigenes Konzept

Auf Grund der Projektanforderungen galt es jedoch, wie oben bereits erläutert, ein Ein-/Ausgabegerät zu entwickeln, das nicht nur aktorische Elemente aufweist, sondern das zusätzlich dem Benutzer als sensorisches Interface dient. Keines der gesichteten Konzepte wurde diesen Anforderungen gerecht, da die Eingabemöglichkeit nur durch zusätzliche (herkömmliche) Eingabegeräte gegeben war.
So verfügt die „**[motionplatform??]**“ aus Beispiel eins zwar über pneumatische Elemente, doch ist die Schaltlogik derart konzipiert, dass die Bewegung der Plattform nur durch Steuersignale des angeschlossenen Computersystems durchgeführt werden kann. Außerdem wurde nach eingehender Überlegung entschieden, dass zwei oder höchstens drei **[freiheitsgrade??]** für die zu realisierende Plattform ausreichend seien.
Auch das Konzept des „Motion Chairs“ aus Beispiel zwei erweist sich nicht als optimale Lösung für dieses Projekt, da er eine aktive Steuerung durch den Benutzer nicht ohne zusätzliche Eingabegeräte erlaubt. Zwar besitzt diese Realisierung gerade zwei **[freiheitsgrade??]**, die für unser Projekt ausreichend wären, doch erfordert die Umsetzung der pneumatischen Schaltungen einen erhöhten Einsatz pneumatischer Bauteile, die nur unter hohem Kostenaufwand beschafft werden könnten. Dies stellt aber einen Widerspruch zum **[lowcost??]** dar. Zudem soll es sich bei der Plattform um ein System handeln, dass einem fliegenden Teppich auch optisch nachempfunden ist.
Aus diesen Gründen wurde beschlossen, ein komplett neues, eigenes System zu entwerfen, dass völlig ohne Aufbauten wie einem Sitz oder ähnlichem ausgestattet und im nächsten Kapitel 14.1.3 näher beschrieben ist.

14.1.3 Entwürfe

Ausgehend von der Anforderungsdefinition, den wie bereits im Kapitel 14.1.2 erwähnten Materialsichtungen und der damit einhergehenden Entscheidung, eine völlig neue **[bewegungsplattform??]** zu entwickeln, wurden verschiedene Teilaspekte näher betrachtet, um mit deren Zusammenspiel ein Gesamtein-/ausgabesystem zu konzipieren.
Im Folgenden sollen diese Teilaspekte näher beleuchtet werden.

Grundwerkstoff

Zunächst galt es zu klären, ob die Plattform wie in der Vorabplanung (Kapitel 14.1.1 Grund für diese Interfaceform / Vorabplanung) beschrieben aus Metall konstruiert werden muss oder vielleicht doch aus einem anderen Werkstoff gefertigt werden kann. Metall als Werkstoff besitzt zwar hervorragende Stabilitätseigenschaften und verkraftet daher hohe Kräfte, jedoch eignete es sich für die „Plattform-Gruppe“ nicht als primären Werkstoff, da die Grundfähigkeiten in der Metallverarbeitung unter den Gruppenmitgliedern nicht vorhanden sind.
Als naheliegende und ebenfalls im Rahmen der Vorabplanung diskutierte Alternative zu Metall galt der Werkstoff Holz, der auf Grund seiner leichten Verarbeitungsmöglichkeiten optimal erschien. Auch bei Holzprodukten ist durch spezielle Verleimungsverfahren eine hohe Kräfteverträglichkeit gegeben. Da die Führungsschienen aus den beiden Varianten des Vorabentwurfs bei konkreteren Überlegungen entfielen, wie später erklärt wird, war Metall auch in diesem Fertigungssequment nicht notwendig.

Konstruktion in zwei Ausbaustufen - Phase eins

Da im Rahmen des Drei-Stufen-Plans, der durch die Projektmitglieder beschlossen wurde, zwei unterschiedliche Fertigungsstufen der Plattform geplant waren, mussten Konstruktionsüberlegungen dahingehend getätigt werden, dass möglichst viele verwendete Bauteile der ersten Ausbaustufe auch in der zweiten weiterverwendet werden konnten.

Die **[bewegungsplattform??]** in ihrer ersten Ausbaustufe sollte dabei lediglich als reines Eingabegerät fungieren. Daher konnte auf sämtliche aktuatorischen Elemente verzichtet werden. Auf Grund der feststehenden Innenfläche des Caves von 2m x 2m wurde die Größe der Plattform-Oberplatte auf 1m x 1m festgesetzt. Diese sollte auf einem Gelenk lagern, dass eine Bewegung der Oberplatte in zwei **[freiheitsgrade??]**n erlaubt. Dieses Gelenk wiederum sollte auf einem Stützblock ruhen und zusammen mit diesem den nötigen Abstand zur Unterplatte bieten. Dabei sollte der Abstand zwischen Ober- und Unterplatte dem Einsatz der verwendeten Sensoren und der gewünschten Plattformneigung (anfänglich $\pm 30°$) entsprechen.
Ein Problem stellte dabei zum einen die Art des Gelenks da, zu dem diverse Überlegungen angestellt wurden. Außerdem musste eine Möglichkeit gefunden werden, die Plattform-Oberplatte in einer Waagerechten zu halten, wenn sie gleichmäßig bzw. gar nicht belastet wurde. Zu diesem Zweck musste ein Dämpfungsmaterial gefunden werden, dass eine Neigung zuließ, wenn der Benutzer dies wünschte. Gleichzeitig sollte dieses aber einen hohen Rückverformungswert besitzen. Dadurch sollte die Plattform-Oberplatte in Ruheposition immer in eine Mittelposition gedrückt werden.

Gelenk: Wie schon bereits erwähnt, musste als Gelenk ein solches gefunden werden, dass einerseits die geforderten Neigungen erlaubte, andererseits aber für die nötige Stabilität der Oberplatte sorgte und die Hauptlast trug. Dabei lag ein besonderes Augenmerk auf der Leichtgängigkeit des Gelenks.
Zu Beginn der Suche nach einem passenden Bauteil wurde bereits ein Kardangelenk als bestmögliche Lösung favorisiert. Da sich aber im Laufe der Suche herausstellte, dass ein solches nicht auf einfachem

Wege kommerziell zu beschaffen sein würde, wurden weitere Lösungsmöglichkeiten erarbeitet.
So wurde zum Beispiel in Betracht gezogen, die Oberplatte auf einer Metallkugel zu lagern und eine spezielle Halterung dafür zu entwerfen.

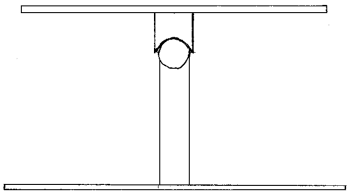


Abbildung 14.9: Erste Entwurfsmöglichkeit: Kugelkopf

Die erste Entwurfsmöglichkeit sah vor, die Kugel mit einem Rohr zu verschweißen, welches fest mit dem Standfuß/Bodenplatte verbunden war. Des Weiteren sollte ein zweites Rohr bzw. dessen Rohrquerschnitt als Auflagering auf der Kugel dienen und durch Schmiermittel leichtgängig gehalten werden. Dabei sollte das zweite Rohr fest mit der Oberplatte verbunden sein.
Ähnlich geartet war eine parallel erarbeitete Entwurfsmöglichkeit, die die Kugel durch einen spitzen Metallstab ersetzte, auf dem, durch eine Vertiefung gehalten, die Oberplatte lagern sollte.

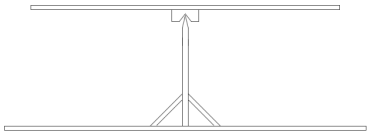


Abbildung 14.10: Erste Entwurfsmöglichkeit: spitzer Stab

Diese Ideen wurden schnell wieder verworfen, da sie als nicht praktikabel eingestuft wurden. Gründe hierfür waren unter anderem das Risiko, dass sich die Oberplatte komplett von der Kugel beziehungsweise vom Stab lösen und somit herabfallen könnte.

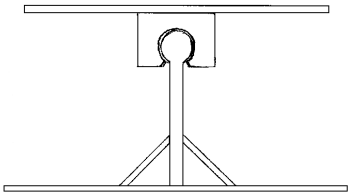


Abbildung 14.11: Zweite Entwurfsmöglichkeit: Kugelkopf mit Ummantellung

Eine zweiter Entwurf sollte das Problem beheben und sah vor, dass die Kugel, die wiederum fest mit dem Standfuß/Bodenplatte verbunden war, so durch eine Haltevorrichtung umschlossen werden sollte, dass ein Herausrutschen unmöglich werden könnte.
Dieser Entwurf wurde genauso wie der vorherige verworfen, da bei genauerer Betrachtung die Reibung der Kugel im Halterungsmantel als zu groß eingeschätzt wurde. Zudem wurde der mögliche maximale Neigungswinkel als zu klein erachtet, der sich durch die Umschließung der Kugel durch den Mantel ergibt.

So wurde erneut verstärkt nach einem Kardangelenk gesucht, dass alle Anforderungen erfüllt, aber keine Nachteile im Sinne des Entwurfs aufweist.
Letztendlich wurde ein Kardangelenk für den Preis von 30 Euro über persönliche Kontakte beschafft,

das aus einem Automobil ausgebaut und mit speziellen Flanschen verschweißt wurde. Auf diese Weise konnte das Kardangelenk mit geeigneten Schrauben und Muttern mit der Oberplatte und dem Stützpfosten verbunden werden. Das Kardangelenk hat folgende technische Eckdaten:

Bezeichnung	Maße	Stärke	Bohrungsabstand (*)	Bohrungsdurchmesser
Oberer Flansch	79mm x 79mm	5mm	8mm	8mm
Unterer Flansch	100mm x 100m	5mm	19mm	8mm
Gesamthöhe	92mm			
Gesamtgewicht	1530g			
Neigung	> 30°			

Tabelle 14.1: Kardangelenk: Technische Eckdaten

(* von beiden angrenzenden Kanten nach innen verlagert)

Abbildung 14.12: Kardangelenk mit Flanschen; zwei Ansichten

Dämpfungsmaterial: Die zweite wichtige Komponente der „**[motionplatform??]**“ in ihrer ersten Version stellt die Dämpfung dar, für die ein geeignetes Material gefunden werden musste. Dieses musste, wie schon erläutert, einerseits einen hohen Absorbtionswert besitzen und sich verformen können, um die vom Benutzer ausgeübte Kraft aufzunehmen, andererseits auch die Fähigkeit haben, sich eigenständig vollständig wieder zurückzuverformen. Auf diese Weise soll sichergestellt sein, dass die Oberplatte stets wieder in die „Mittelposition“ zurückgestoßen wird.
Als kostengünstige Variante wurde relativ schnell Schaumstoff als geeignetes Dämpfungsmaterial für den ersten Prototypen ausfindig gemacht. Zu diesem Zweck wurde ein Fachhändler aufgesucht, bei dem diverse Schaumstoffe auf ihre Steifigkeit und ihren Rückverformungsgrad sowie -geschwindigkeit getestet wurden.
Dabei wurde letztendlich der Schaumstoff mit der größten Steifigkeit gewählt, da er die gewünschten Eigenschaften am ehesten erfüllte. Der Gesamtpreis für die erste Dämpfungseinheit betrug weniger als zehn Euro.

Abbildung 14.13: Schaumstoffblock zur Dämpfung

Der Schaumstoff wurde in diesem Fall auf 300mm x 170mm x 120mm (H x B x T) zugeschnitten und bot somit knapp 30 cm Dämpfungsweg, was einer Neigung von mehr als 30° entspricht. Hierbei wurde der Schaumstoffblock auf zehn Zentimeter hohen Holzsockeln angebracht, um einerseits die verwendeten Sensoren, die später erläutert werden, vor Beschädigungen zu schützen und andererseits den Dämpfungsweg beziehungsweise die Neigung zu begrenzen.

Das erste lauffähige Gesamtsystem beinhaltete darüberhinaus noch Distanzsensoren, die den Abstand zwischen Unter- und Oberplatte messen. Aus diesem Abstand konnte so die Neigung der Plattform per Software errechnet werden. Zudem wurde ein Lagerblock verwendet, auf dem das Kardangelenk und somit die obere Platte ruhten. Die Höhe des Lagerblocks und des Kardangelenkes waren zusammen genau so hoch wie die Schaumstoffblöcke mit ihren Holzsockeln. Auf diese Weise war sichergestellt, dass die Plattform stets waagerecht ausgerichtet war.

Diese erste Plattform-Version wurde im Rahmen des ersten Zwischenpräsentationstermins des Projekts im Oktober 2003 erstmals verwendet. Im Anschluss daran folgte die Planung zum Umbau der Plattform zu einem Ein-/Ausgabegerät.

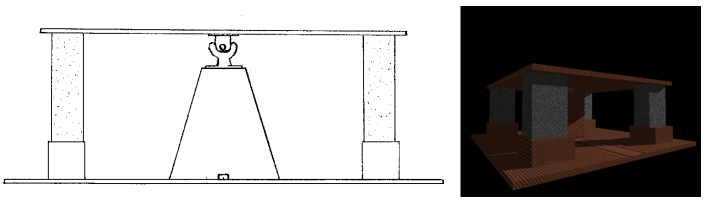


Abbildung 14.14: Erste Version der Plattform

Konstruktion in zwei Ausbaustufen - Phase zwei

In ihrer zweiten Ausbaustufe sollte die Plattform neben den sensorischen Elementen auch aktorische Bauteile erhalten. An den Sensoren wurde festgehalten, weil sie auf Grund ihrer einfachen Ansteuerbarkeit leicht zu verwenden waren.
Aktuatoren sollten eingebunden werden, um die Plattform zu einer wirklichen „**[motionplatform??]**“ zu machen, wozu sie in der Lage sein musste, „**[forcefeedback??]**“-Signale der Engine in reale Kräfte umzuwandeln und an den Benutzer weiterzugeben.
Im Rahmen der Vorabplanung wurden hierzu bereits erste Überlegungen getätigt. Diese waren jedoch relativ schwierig umzusetzen, da sie neben einer Dämpfung durch ein Federsystem ein aufgesetztes

Pneumatik-System besassen, dass allein für die aktuatorischen Fähigkeiten der Plattform verwendet wurde. Nach umfangreichen Recherchen der „Plattform-Gruppe“ kam die Idee auf, ein Dämpfungssystem zu entwerfen, dass ohne Federn und nur mit Pneumatik-Zylindern arbeiten konnte. Andererseits sollte dieses System aber in einem separaten „**[forcefeedback??]**“-Modus“ in der Lage sein, aktiv Kräfte auszuüben.
Außerdem erwies sich die Frage der Wegemessung als schwierig, durch die in der ersten Variante der Vorabüberlegung die Neigung der Plattform durch eine Vielzahl von möglichen Zylinderpositionen bestimmt werden sollte. Dieses System erwies sich jedoch bereits im Vorfeld als zu kostenintensiv und widersprach somit dem **[lowcost??]**. Daher musste ein einfacheres System entworfen werden, mit dem zumindest drei Positionen auf jeder Neigungsachse der Plattform angesteuert werden konnten. Gemeinsam mit Herrn Gathmann vom **[artec??]**, der ein umfangreiches Fachwissen auf dem Gebiet der Pneumatik aufweist, kam die Idee auf, zwei Pneumatik-Zylinder gegeneinander arbeiten zu lassen. Dabei sollten sie so aufgebaut werden, dass die Zylinderböden fest miteinander verbunden werden und die Kolbenstangen somit gegen die Ober- bzw. die Unterplatte arbeiten. Diese Idee garantierte genau drei definierte Zustände:

- komplett ausgefahren
- feste Mittelposition
- komplett einfahren

Bei dieser Idee wurde allerdings der Nachteil festgestellt, dass die Gesamthöhe der „Plattform“ sehr stark anwachsen würde, wenn man beide Zylinder auf die oben genannte Art und Weise mit einander verbindet. Daher wurde das Konzept verfeinert, indem man beide Zylinder nebeneinander arbeiten lässt. Nachdem damit im Rahmen der Gesamtplanung grundsätzlich geklärt war, dass Pneumatik-Zylinder als Dämpfungs- und als **[forcefeedback??]**-Bauteile verwendet werden können, mussten nun konkrete Realisierungsüberlegungen angestellt werden, die im Folgenden näher erläutert werden.

Gelenk: Da sich das bereits in der ersten Ausbaustufe verwendete Kardangelenk bewährt hatte und an der allgemeinen Plattform-Konstruktion mit zwei **[freiheitsgrade??]**n festgehalten wurde, stand der Verwendung des Gelenks auch in dieser Umsetzungsvariante nichts im Wege. Abhängig war das Gesamtkonstrukt „Gelenk-Mittelpfosten“ jedoch noch von der Länge der verwendeten Zylinderpaare, die noch zu bestimmen war.

Pneumatikschaltkreis für Dämpfungs- und „[forcefeedback??]“-Modi: Wie oben bereits erwähnt, musste ein geeigneter Schaltkreis entworfen werden, der einerseits die Dämpfungs- und ForceFeedback-Modi unterstützte, andererseits der Gesamtplanung entsprach und hauptsächlich den **[lowcost??]** verfolgte.
In den Ansätzen wurden bereits zu Beginn zwei Druck-Kreisläufe verwendet, die mit unterschiedlichen Arbeitsdrücken arbeiteten. Interessant bei der Realisierung war daher, wie zwischen den einzelnen Kreisläufen auf einfache Art gewechselt werden konnte.

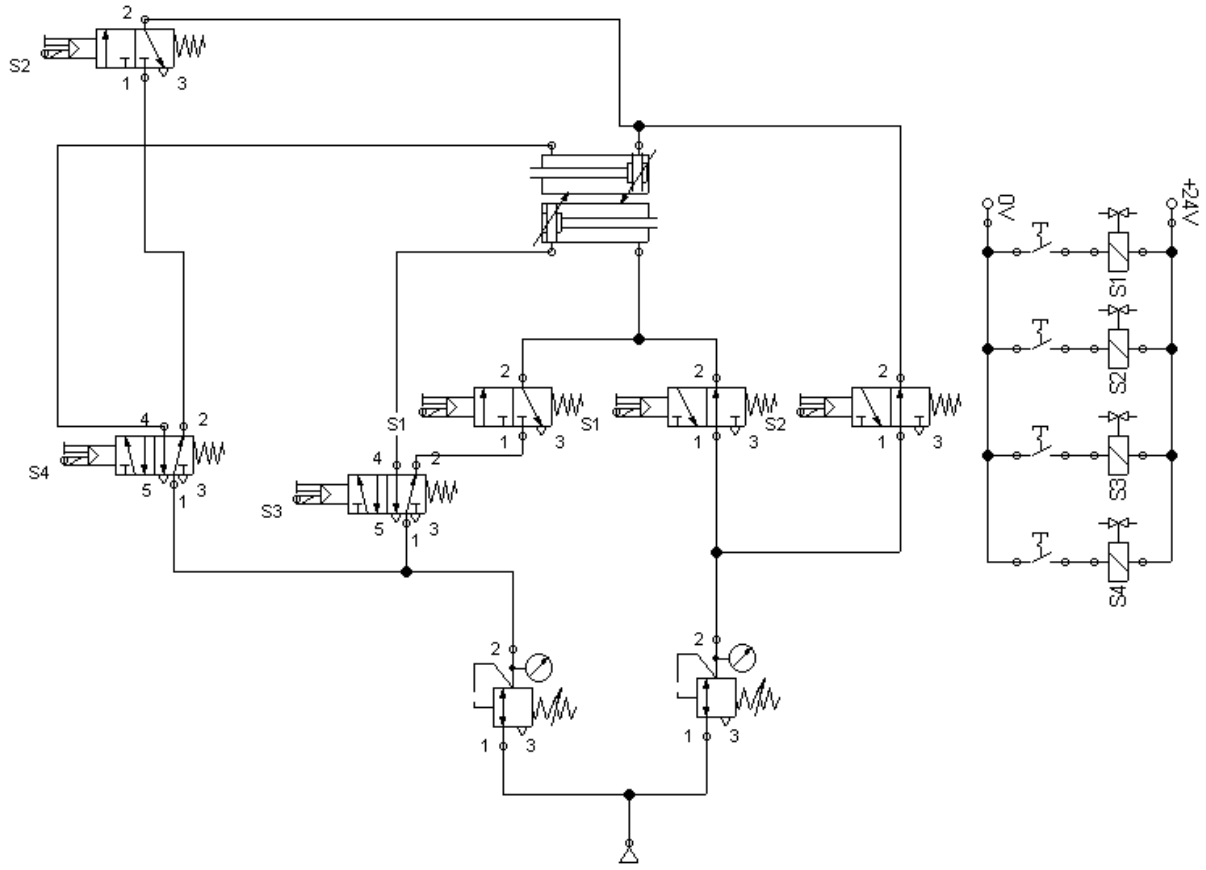


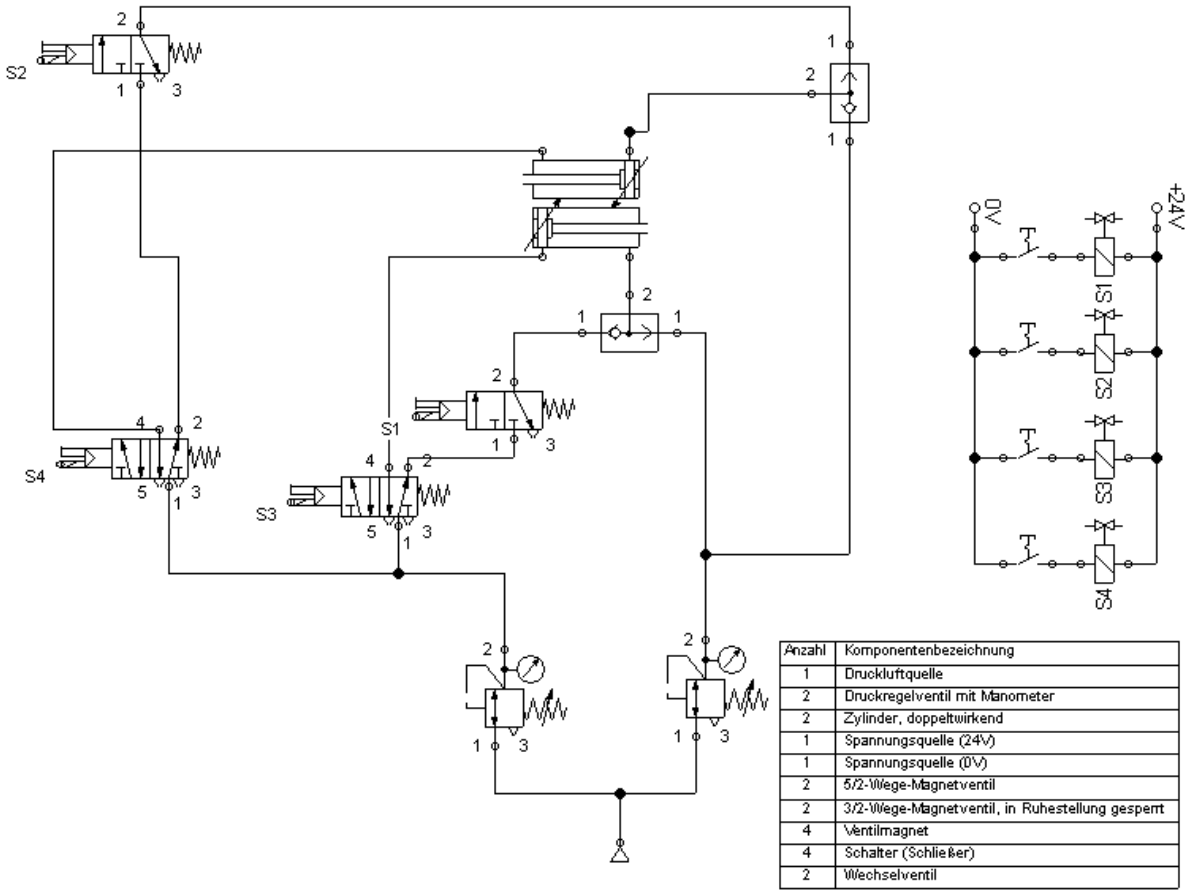
Abbildung 14.15: Erster Entwurf des Pneumatikkreises

Dieser Schaltplan wurde exemplarisch für zwei Zylinderpaare entworfen, zeigt jedoch nur ein Zylinderpaar mit entsprechenden Schaltelementen.
Mittels zweier Druckregelventile wurden die Drücke zweier Schaltkreise festgelegt. Ein Schaltkreis („**[forcefeedback??]**“-Kreis) sollte dabei einen hohen Druck von beispielsweise 5 bar aufweisen, der andere (Dämpfungskreis) sollte mit einem abgeminderten Druck von beispielsweise 1 bar eine praktikable Dämpfung gewährleisten.
Um von einem zum anderen Druckkreislauf wechseln zu können, wurden in dieser Lösungsvariante insgesamt sechs elektronisch ansteuerbare Magnetventile verwendet. Innerhalb des Dämpfungskreises wurden zwei „schließende **[3-2-wegeventil??]**e“ verwendet. Damit wurde standardmäßig einer der beiden Zylinder eingefahren und einer ausgefahren. Auf diese Art und Weise wurde eine Mittelposition der Plattform angefahren, die sich grob gesagt aus dem Abstand zwischen ausgefahrenem Kolbenstangenende und eingefahrenem Kolbenstangenende ergab.
Wollte man nun einen „**[forcefeedback??]**“-Impuls auf die Plattform ausgeben, konnte man die Magnetventile im Dämpfungskreis schließen und je nach Wunsch den hohen Druck des „ForceFeedback“-Kreises auf die Zylinder beaufschlagen. Dafür waren im „**[forcefeedback??]**“-Kreis vier Magnetventile vorhanden; zwei „**[5-2-wegeventil??]**e“ und zwei „öffnende **[3-2-wegeventil??]**e“ sorgten entsprechend für die Druckbeaufschlagung der jeweiligen Zylinderanschlüsse. Auf diese Weise war es möglich, die Pneumatikzylinder gezielt mit unterschiedlichen Drücken ein- beziehungsweise auszufahren und so entweder der Plattform entsprechende Neigungen zuzuführen oder die Plattform auf Kräfte durch den Benutzer dämpfend reagieren zu lassen.

Auf Grund der Bauweise der Magnetventile musste diese Version jedoch verworfen werden. Ohne erste Erfahrungen auf dem Gebiet der Pneumatik war uns dabei nicht bewusst, dass die Ventile in geschlos-

sener Stellung den Druckkreislauf teilweise entlüfteten. Hierdurch wäre es gar nicht möglich gewesen, den vollen Arbeitsdruck aufzubauen. Daher wurde auf eine neue Variante zurückgegriffen, die nach eingehender Recherche entstand.

Bei diesem Entwurf wurden die beiden „schließenden **[3-2-wegeventil??]**e“ aus Variante eins durch



Anzahl	Komponentenbezeichnung
1	Druckluftquelle
2	Druckregelventil mit Manometer
2	Zylinder, doppeltwirkend
1	Spannungsquelle (24V)
1	Spannungsquelle (0V)
2	5/2-Wege-Magnetventil
2	3/2-Wege-Magnetventil, in Ruhestellung gesperrt
4	Ventilmagnet
4	Schalter (Schließer)
2	Wechselventil

Abbildung 14.16: Zweiter Entwurf des Pneumatikkreises

„**[oderglied??]**er“ ersetzt. Ein solcher Baustein reagiert derart auf zwei angeschlossene Druckstärken, dass er an seinen Ausgang den höheren Druck weitergibt und den geringeren sperrt. So wurde das Problem auf einfache Art und Weise umgangen, dass Luft aus dem Gesamtsystem entweicht. Zudem wurde die Elektroniksteuerung der Magnetventile vereinfacht, da die „**[oderglied??]**er“ autonom ohne einen elektronischen Impuls schalten.

So war es möglich, durch Schließen des Schalters S3 das dazugehörige Magnetventil umzuschalten und dadurch den (in dem Schaltplan) unteren Zylinder gegen den Dämpfungsdruck auszufahren. Durch Schließen des Schalters S1 und Öffnen des Schalters S3 war es zudem möglich, den unteren Zylinder mit „**[forcefeedback??]**“-Druck einzufahren. Für den oberen Zylinder verhielt sich die Schaltlogik analog mit dem Unterschied, dass S2 und S4 geschaltet werden mussten.

Im Rahmen der Simulation mit FluidSim (siehe [**?**]), mit der auch die Schaltpläne erstellt wurden, trat das Problem der fehlenden Entlüftung im Dämpfungskreis auf. Wurde beispielsweise der untere Zylinder mittels des „**[forcefeedback??]**“-Kreises ausgefahren, konnte die Luft im anderen Zylinderteil nicht entweichen, wodurch der Druck im gesamten Dämpfungskreis leicht anstieg. Dies war kein sehr großes Problem, da durch diese Wechselwirkung lediglich der Dämpfungseffekt leicht verstärkt wurde, und zwar genau in die Richtung, in die auch der „**[forcefeedback??]**“-Impuls wirkte. Jedoch wurde im Rahmen der Endrealisierung für dieses Problem kontinuierlich an einer Lösung gearbeitet, was in Kapitel 14.1.6

näher beleuchtet wird.

Im Grundsatz wurde dieser zweite Entwurf des Pneumatikkreises jedoch als brauchbar angesehen und konkret mittels Festo-Didactic-Pneumatik-Bauteilen im **[artec??]**-Labor getestet. Auch diese realen Tests waren sehr vielversprechend, da bereits mit den sehr kleinen, bei einem Arbeitsdruck von 6 bar geringe Kräfte entwickelnden Lehrzylindern relativ gute Ergebnisse erzielt wurden.

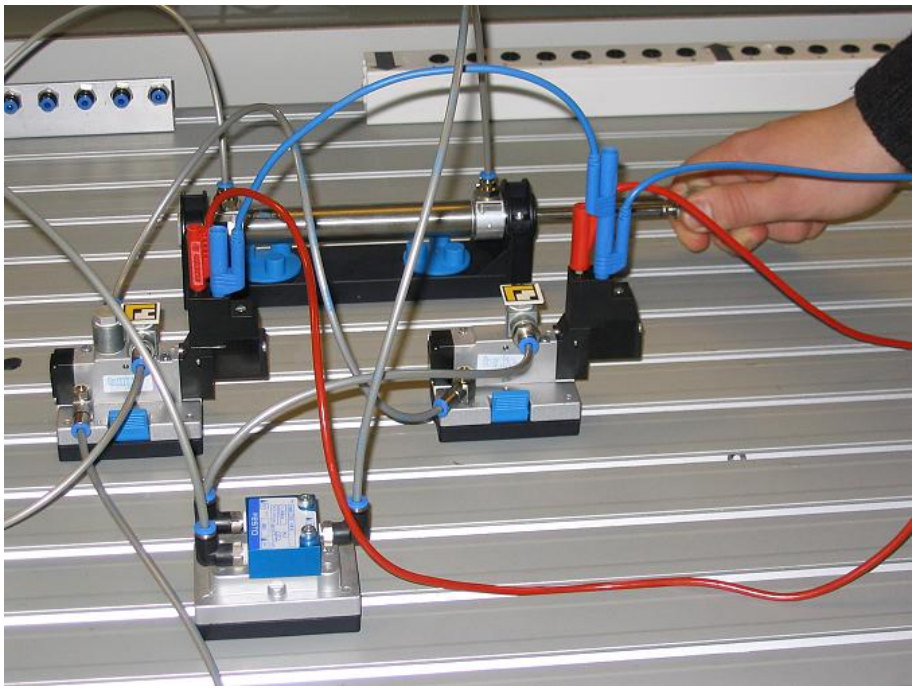


Abbildung 14.17: Realer Test mit Festo-Didactic-Bauteilen

Mit der Gewissheit, dass der geplante Schaltkreis den gewünschten Anforderungen entspricht, wurde mit der konkreten Beschaffung anhand der Materialliste begonnen. Hierzu musste aber zunächst ein Sponsor gefunden werden, was im Kapitel 14.1.4 näher beleuchtet wird.

14.1.4 Beschaffung / Sponsoring

Auf Grund des vom Projektplenum vorgeschriebenen **[lowcost??]** und der konsequenten Verfolgung dieses Ziels musste grundsätzlich nach Möglichkeiten gesucht werden, benötigte Bauteile kostengünstig zu beschaffen. Diese Aufgabe bedeutete für die „**[motionplatform??]**“-Gruppe einen erhöhten Arbeitsaufwand, wodurch nicht zuletzt zeitliche Verzögerungen entstanden, die an anderer Stelle wieder aufgeholt werden mussten.

Für die Realisierung der ersten Plattform-Version, die oben bereits ausführlich beschrieben wurde, wurden Bauteile verwendet, die auf Grund von privaten Beziehungen günstiger erworben werden konnten oder gänzlich gesponsort wurden. Die Tischlerplatten, aus denen die Ober- und auch die Unterplatte der **[bewegungsplattform??]** gefertigt wurden, wurden von der Tischlerei für zirka 50 Euro günstig erworben, die auch das Holz für die Cave-Konstruktion gesponsort hat. Dieses Material wurde auch in der zweiten Version der Plattform komplett übernommen und aus den Verschnittresten wurden später die Seitenarme gefertigt, auf denen die Pneumatik-Schaltungslogik und die Sensoren angebracht wurden. Hierauf wird im Kapitel 14.1.5 eingegangen.

Das Kardangelenk, welches ebenso für beide Plattform-Versionen in Verwendung stand, wurde ebenfalls durch private Kontakte bezogen. Das Gelenk, das ursprünglich Teil einer Automobil-Antriebswelle ist, wurde von Personen aus dem Umfeld des Motorsports aus einem Unfallfahrzeug ausgebaut und mit Flanschen mit vorgegebenen Bohrungen versehen. Eine Bearbeitung des Gelenks von der Gruppe her wäre nicht möglich gewesen, da es aus speziell gehärtetem Stahl besteht. Aus diesem Grund hat es einen unermesslichen Wert für uns, da wir es aus anderer Quelle nur schwer hätten beschaffen können. Und gleichzeitig erfüllt es alle Anforderungen, die an ein solches Gelenk gestellt wurden.

Die beiden optischen Distanzsensoren (GP2D12 von **[sharp??]**) wurden beim Elektronikhändler Conrad gekauft. Genauso wurden die meisten elektronischen Bauteile, zum Beispiel Widerstände, Relais, Sperrdioden, Kabel, Stecker und Schalter dort erstanden.

Als größtes Beschaffungsfeld sind die Pneumatik-Bauteile zu nennen, die auf Grund der eingeschränkten finanziellen Mittel nicht aus der Projektkasse hätten bezahlt werden können. Daher musste für diesen Bereich ein bereitwilliger Sponsor gefunden werden. Nach umfangreicher Suche und mehreren persönlichen Gesprächen wurde die Firma Ahlrich-Siemens GmbH aus Bremen gefunden. Da das vorgestellte bereits fertige Konzept der Pneumatik-Schaltung für eine „**[motionplatform??]**“ als Ein- und Ausgabegerät überzeugte, wurde dem Projekt umfassende Hilfe zugesagt. Die Firma ist regionaler Vertriebspartner der Firma Festo, von der ebenfalls die verwendete Software FluidSim und die Didactic-Bauteile stammen, die zur Planung und zum Test der ersten Bauvarianten verwendet wurden. Jedoch vertreibt die Firma hauptsächlich Pneumatikbauteile für den industriellen Einsatz, wodurch sich die Möglichkeit ergab, kostengünstig an gebrauchtes Material zu gelangen. So erhielten wir nicht nur vier Zylinder, die über unsere Anforderungen hinausgingen, sondern auch alle benötigten logischen Schaltungselemente, Schläuche, Steckverbindungen, Verteiler, Dämpfern und Druckregelventile in mehrfacherer Ausführung. Im Rahmen von Bauteilwechseln zeigte sich die Firma ebenfalls besonders kulant, da beispielsweise Druckregelventile auf Grund von Tests getauscht werden sollten. Insgesamt wurden auf diese Weise Bauteile mit einem Neuwert von über 2.000 Euro zur Verfügung gestellt, für die ein Gesamtpreis von ungefähr 225 Euro bezahlt wurde.

Um jeweils zwei Pneumatik-Zylinder miteinander verbinden zu können, wie es der Entwurf vorsah, musste eine Möglichkeit gefunden werden, diese stabil verknüpfen zu können. Hierfür wurden „L-Stahlstücke“ von der Firma Metallbau Finke für einen günstigen Betrag von acht Euro beschafft und zugesägt. Diese mussten anschließend noch mit Bohrlöchern versehen werden, die der Verschraubung

der Zylinderpaare dienten. Zusätzlich wurden die Metallstücke entgratet. Dies erledigte die Metallwerkstatt im Zentralbereich der Universität Bremen kostenlos.

Zuletzt wurde ein Teppich für den Projekttag beschafft, der der „**[motionplatform??]**“ endgültig das Aussehen eines fliegenden Teppichs verschaffen sollte. Hierzu wurden mehrere Händler von Orientteppichen aufgesucht, die sich grundsätzlich zumeist bereiterklärten, über ein Sponsoring nachzudenken. Letztendlich hat uns Orientteppiche Route aus Bremen das entgegenkommendste Angebot gemacht, da dieser neben einem originalen handgeknüpftem Orientteppich aus Nain/Iran im Wert von 350 Euro ebenfalls für den Projekttag Dekorationsmaterial zur Verfügung stellen wollte. Hierzu gehörten weitere Teppiche, Tonkrüge und Raumschmuck aus Edelmetallen. Hierfür wurde lediglich ein Preis von 50 Euro entrichtet.

Abbildung 14.18: Handgeknüpfter Orientteppich des Sponsors Route

Die **[bewegungsplattform??]** „Magic Carpet“ konnte insgesamt dank des Sponsorings für einen Gesamtbetrag von zirka 400,- Euro gebaut werden. Insgesamt wird jedoch der rein materielle Wert auf über 2.600 Euro geschätzt.

Das Projekt Micarpet dankt den großzügigen Sponsoren an dieser Stelle noch einmal rechtherzlich, ohne deren Zutun diese „**[motionplatform??]**“ niemals hätte realisiert werden können.

14.1.5 Sensorik

Um die **[bewegungsplattform??]** „Magic Carpet“ überhaupt als Eingabegerät verwenden zu können, sind diverse Sensoren notwendig, die in diesem Kapitel näher erläutert werden sollen.

Optische Distanzsensoren

Die wichtigsten Sensoren und Sensoren im eigentlichen Sinne sind die verwendeten optischen Distanzsensoren vom Typ GP2D12 von **[sharp??]**. Die Sensoren werden verwendet, um den Abstand zwischen der Oberplatte und einer fixen Platte zu verwenden. Dabei haben die Sensoren folgende Eckdaten:

möglicher minimaler Abstand	100mm
möglicher maximaler Abstand	800mm
Signalstärke V_{out} bei minimalem Abstand	2,6V
Signalstärke V_{out} bei maximalem Abstand	0,4V
Versorgungsspannung V_{CC}	5,0V
Stromstärke I_{CC}	33mA
Umgebungstemperatur T_{opr}	-10°C bis +60°C

Tabelle 14.2: Optische Distanzsensoren: Technische Eckdaten (siehe [**?**])

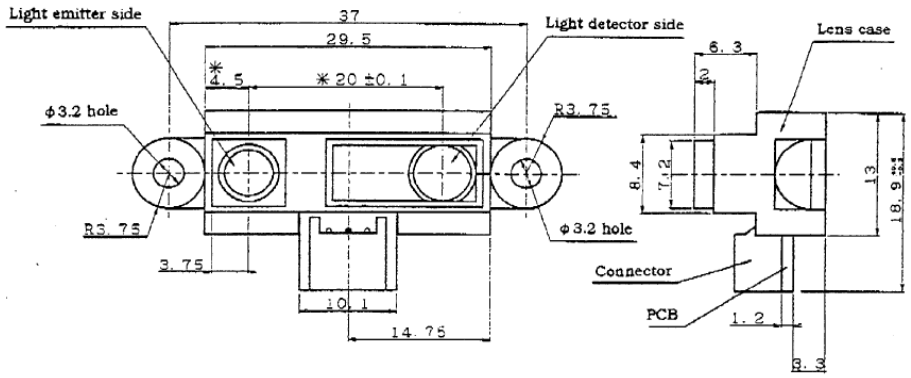


Abbildung 14.19: Optischer Distanzsensor (siehe [**?**])

Damit waren dies die geeigneten Sensoren für die Realisierung, da der mögliche Abstand zwischen Sensor und dem zu messenden Objekt innerhalb des Messbereichs der Sensoren lag. Die einfache Ansteuerbarkeit war bei diesen Sensoren auch optimal. Zum einen konnten die Sensoren auf einfache Art und Weise über das verwendete „**[SIOS??]**“ mit Spannung ($5V_{dc}$) versorgt werden. Zum anderen lieferten die Sensoren als Ausgabesignal stetig einen Wert, der über die Software an den „analogen Eingängen“ ständig abrufbar war. Hierauf wird im Kapitel „**[SIOS??]**“ eingegangen.

In der ersten Variante der Plattform waren die Sensoren auf der Unterplatte befestigt und dabei 45 cm vom Plattenmittelpunkt entfernt auf den Seitenhalbierenden angebracht. Da die optischen Distanzsensoren eine Messung zwischen zehn Zentimetern und 80 Zentimetern erlaubten, musste sichergestellt sein, dass die Oberplatte einen Mindestabstand von zehn Zentimetern nicht unterschreitet. Hierzu wurden die bereits oben erwähnten Holzsockel verwendet, auf denen die Dämpfungselemente montiert waren.

Gleichzeitig schützte diese Art der Konstruktion die optischen Distanzsensoren vor ungewollter Zerstörung durch die Oberplatte.

Abbildung 14.20: Optischer Distanzsensor

In der zweiten Variante der Plattform wurde auf Grund der verwendeten Bauteile eine geringe Neigung der Oberplatte spezifiziert. Auf diese Weise war es möglich, die Sensoren mittels zusätzlicher Seitenarme näher an der Oberplatte zu montieren, was den Vorteil hatte, dass genauere Werte geliefert wurden; genauer gesagt sind die Werte, die die optischen Sensoren liefern, im Wertebereich von geringeren gemessenenen Abständen genauer und besser untereinander differenzierbar als gemessene Werte größeren Abstands. Dies ergibt sich insbesondere aus dem folgenden Diagramm:

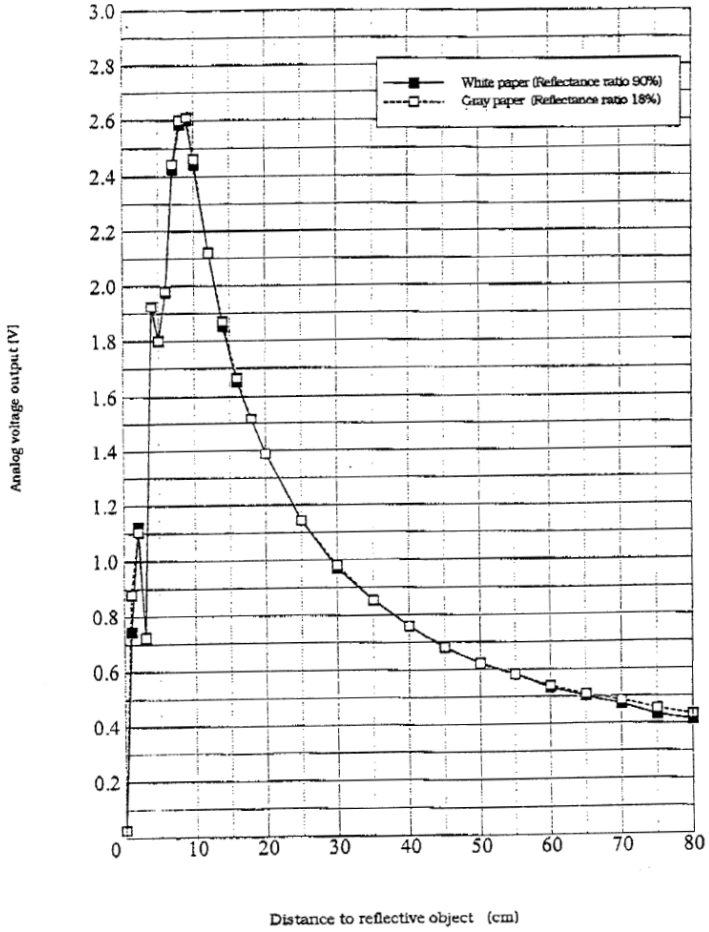


Abbildung 14.21: Diagramm der Ausgabewerte des optischen Distanzsensors (siehe [**?**])

Anmerkung: Mit in diesem Kapitel aufgenommen sind die Knöpfe zur Steuerung der Geschwindigkeit und die Schalter zur Steuerung der Plattform, die im eigentlichen Sinne keine Sensoren darstellen. Die freie Enzyklopädie Wikipedia definiert einen Sensor als „Bauteil, das neben bestimmten physikalischen oder chemischen auch die stoffliche Beschaffenheit seiner Umgebung qualitativ oder als Messgröße quantitativ erfassen kann“.(Wikipedia, www.wikipedia.org/wiki/Sensor, zuletzt gesehen am 03.09.2004)

Knöpfe zur Steuerung der Geschwindigkeit

Zur Steuerung der Fluggeschwindigkeit des virtuellen Teppichs, genauer gesagt zur Veränderung der Länge des Richtungsvektors, wurden von der „Interface-Gruppe“ zwei Taster verwendet. Einer davon war zur Inkrementierung der Länge zuständig einer zur Dekrementierung. Für die erste Variante der Plattform wurden die Taster provisorisch in handgeformte Pappe-Kunststoff-Verbindungen integriert. Angeschlossen wurden diese an die „digitalen Eingänge“ des „**[SIOS??]**s“, auf dessen softwareseitige Implementierung im Kapitel „**[SIOS??]**“ eingegangen wird.

In der zweiten Plattform-Variante wurden die Taster in zwei Fahrradlenker-Griffe integriert, die so an der Plattform von unten verschraubt wurden, dass der Benutzer in sitzender Haltung diese problemlos erreichen und sich an diesen in extremen Neigungssituationen Halt suchen konnte. Dabei wurde der rechte Taster als solcher für die Beschleunigung ausgewählt, da die rechte Hand vermutlich auf Grund der Analogie zum Automobil intuitiv dafür verwendet wird. Mit dem linken Taster wurde die negative

Beschleunigung erhöht. Dies bedeutet zum einen, dass eine positive Beschleunigung vermindert wird, was zur Folge hat, dass der virtuelle Teppich langsamer wird, zum anderen aber auch, dass eine bestehende negative Beschleunigung erhöht wird. Hierdurch erhöht sich die Geschwindigkeit gegen die Sichtrichtung.

Schalter zur Steuerung der Plattform

Um unabhängig vom PC bzw. der Engine in das Verhalten der „**[motionplatform??]**“ eingreifen zu können, wurden von der „Plattform-Gruppe“ zwei rot bzw. grün beleuchtete Schalter verwendet.
Der rote Schalter diente dabei als „**[forcefeedback??]**“-NotAus und unterbrach den kompletten Schaltkreis, der es der Engine ermöglicht, „**[forcefeedback??]**“-Signale an die Plattform zu schicken. Bereits im Vorfeld wurde bei der Planung diskutiert, welche Maßnahmen zu ergreifen wären, um ein Höchstmaß an Sicherheit mit der Plattform zu ermöglichen. Ein ungewolltes Abrutschen des Benutzers von der Oberplatte war als eine wahrscheinliche Variante zu betrachten, beispielsweise wenn zuerst ein „**[forcefeedback??]**“-Impuls vorn und unmittelbar folgend von hinten erfolgte. Um weitere „**[forcefeedback??]**“-Stöße auf den Benutzer vermeiden zu können, konnte dieser Schalter eingesetzt werden.
Der grüne Schalter wurde zusätzlich integriert, um die Plattform in eine stabile, waagerechte Position zu bringen. Zu diesem Zweck wurden die Pneumatik-Zylinder derart mit Druck beaufschlagt, dass sie ihre volle Tragkraft entfalten konnten. Dies geschah, um dem Benutzer einerseits ein leichtes Auf- und Absteigen von der Plattform zu ermöglichen, andererseits aber auch die Möglichkeit bieten, in Extremsituationen dem Benutzer eine stabile Grundlage ohne Neigung zu bieten, damit dieser sich wieder auf der Plattform in gewünschter Haltung ohne Störungen positionieren kann.

Es wurden zwei farblich unterschiedliche Schalter verwendet, um die Funktionen dieser besser voneinander unterscheiden zu können. Dabei stellt die Farbwahl mit rot und grün unter Umständen ein Problem für „Rot-Grün-Blinde“-Menschen dar, doch wurde eine Anordung gewählt, die einer Fussgängerampel entspricht. Dennoch ist darauf zu achten, dass allen Personen, die zur Bedienung eingeteilt werden, der Funktionsumfang geläufig ist.

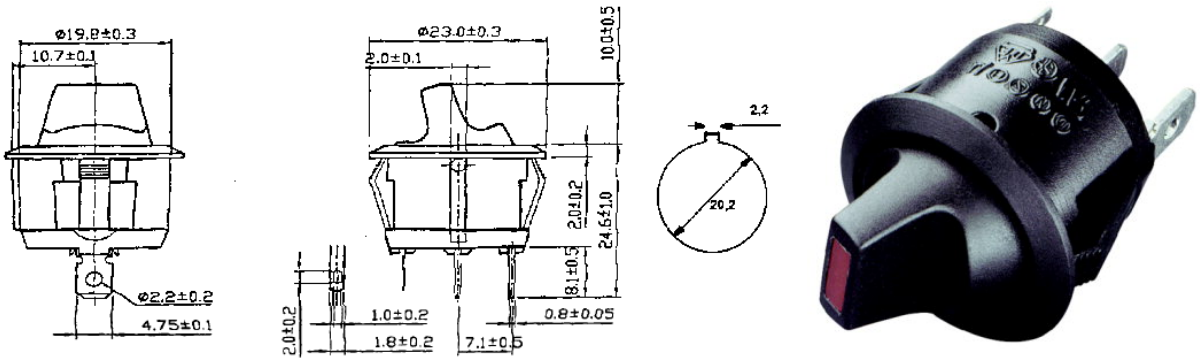


Abbildung 14.22: Skizze und Foto des Kippschalters (siehe [**?**])

Die Schalter wurden erst im Zuge der zweiten Plattform-Variante integriert und waren Bestandteil des Caves. Sie waren im Rahmen eingelassen und ermöglichten einer Person, die zur Aufsicht dort positioniert war, in einer Gefahrensituation durch Umlegen des entsprechenden Schalters, dem Benutzer auf der Plattform zu helfen.

Genauer wird im Kapitel 14.1.6 auf die Schaltkreise eingegangen.

14.1.6 Aktorik

Im Rahmen der zweiten Umsetzungsvariante des Eingabegerätes zu einer **[bewegungsplattform??]** wurden Aktorikelemente integriert.
In diesem Kapitel wird die finale Umsetzung erläutert, die eine konsequente Fortführung der ersten Pneumatikkonzepte darstellt. Hierfür wurde ein zweites Zylinderpaar und die entsprechende Schaltungslogik erweitert.
Der Schaltplan sieht wie folgt aus:

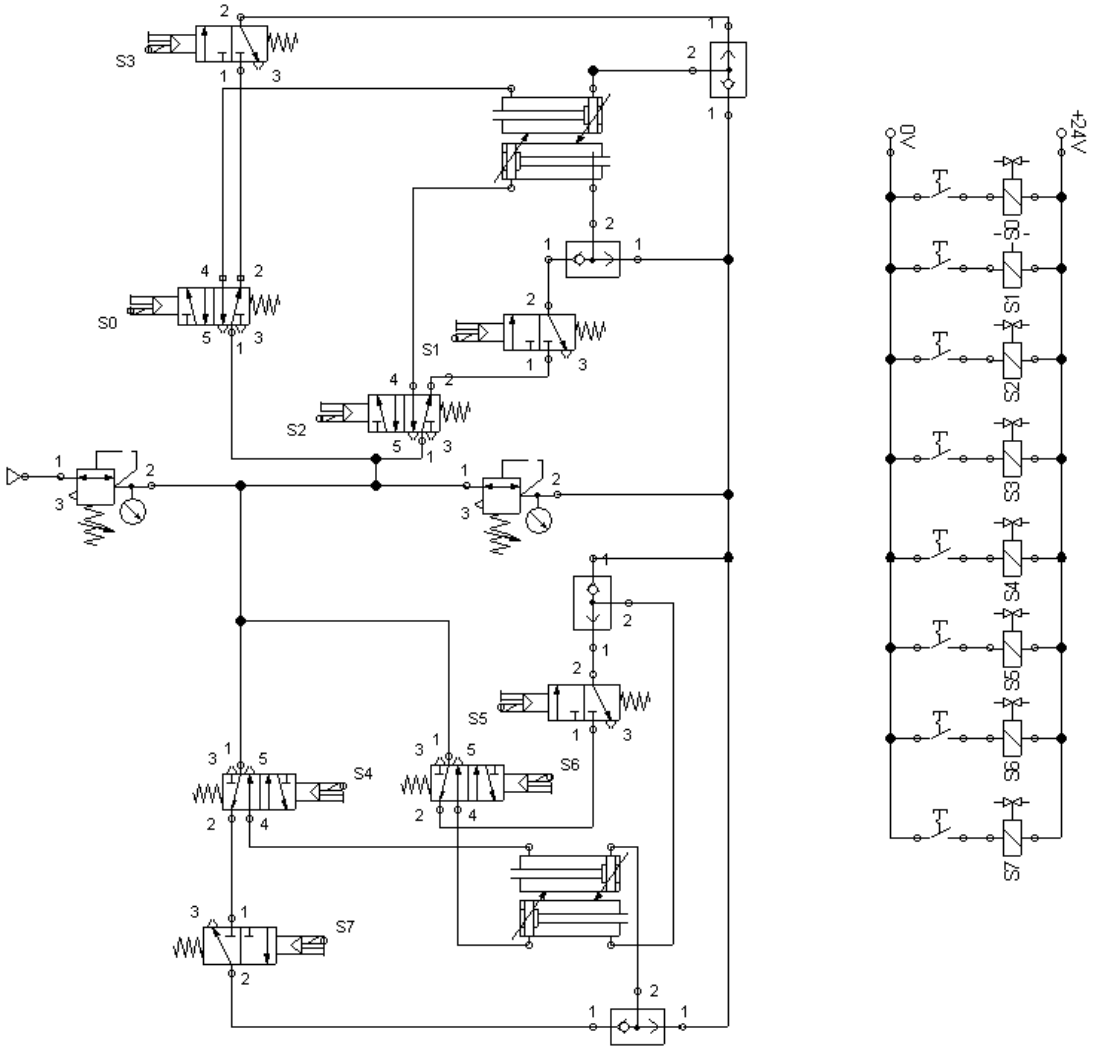


Abbildung 14.23: Finaler Schaltplan

Die einzelnen Elemente der Pneumatikschaltung werden im Folgenden näher beschrieben.

Zylinder

Wie schon in den ersten Entwürfen zu einer **[motionplatform??]** angedacht, wurden zwei Zylinderpaare zu Dämpfung und zur Ausgabe von „**[forcefeedback??]**“ verwendet.
Die verwendeten doppeltwirkenden Zylinder wurden produziert von der Firma Festo und tragen die Produktbezeichnung DN-63-200-PPV. Die Zylinder haben folgende technische Kennzahlen:

entspricht Norm	ISO 6431
Kolbennenngröße	63mm
Hub	200mm
Kolbenstangendurchmesser	20mm
Kolbenstangenende	Außengewinde (M16 x 1,5)
Nutzkraft (theoretisch) bei 6 bar Vorlauf	1761 N
Nutzkraft (theoretisch) bei 6 bar Rücklauf	1601 N
Luftverbrauch bei 6 bar Vorlauf/Hub	4,22 l
Luftverbrauch bei 6 bar Rücklauf/Hub	3,8 l

Tabelle 14.3: Zylinder: Technische Eckdaten (siehe [**?**])

Auf Grund der Eckdaten sind diese Zylinder ausreichend für den Einsatz in der „**[motionplatform??]**“ „Magic Carpet“. Spezifiziert wurde, wie in Kapitel 14.1.1 beschrieben, dass die Plattform stark genug sein muss, um einen männlichen, „gut gebauten“ Erwachsenen zu tragen. Auch der Hub ist ausreichend, um eine Neigung an den Ecken von $\pm 17{,}5°$ und $\pm 21{,}5°$ an den Seitenhalbierenden zu erreichen, da jeder Zylinder 20 Zentimeter Hub aufweist und die Zylinder auf den Diagonalen zirka 63 Zentimeter vom Mittelpunkt entfernt montiert sind. Dabei erreicht die Plattform eine maximale Beschleunigung von 0,4 m/s.
Angebracht wurden die Zylinder auf den Diagonalen, da dort die auftretenden Winkelgrößen auf die Befestigungselemente, die weiter unten beschrieben werden, geringer sind als auf den Seitenhalbierenden. Die Zylinder stellen dabei frei schwebende Elemente dar, die sich selbst tragen und mit der Ober- beziehungsweise Unterplatte nur durch die Befestigungselemente verbunden sind.

Zylinder-Verbindungselemente

Zum paarweisen Verbinden der Zylinder mussten Vorrichtungen gefunden werden, die den auftretenden Kräften gewachsen waren. Dazu wurden die Winkelstahle verwendet, die bereits im Kapitel 14.1.4 erwähnt wurden.
Hierzu mussten handelsübliche, gleichschenklige L-Eisen auf exakt die Länge zugeschnitten werden, die zwei Zylinder nebeneinander aufweisen. Da die Zylinder eine Montage mittels Schrauben auf den Ober- und Unterseiten vorsehen, mussten Löcher mit entsprechenden Abständen in die Winkeleisen gebohrt werden. Anschließend konnten dann zwei Zylinder miteinander verschraubt werden.

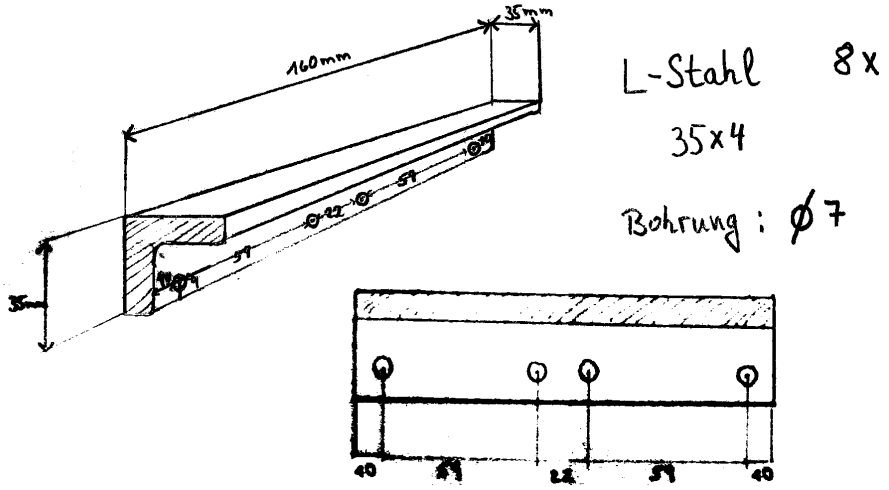


Abbildung 14.24: Skizze zum Zylinder-Verbindungselement aus L-Stahl

Die Skizze wurde als Vorlage in der Metallwerkstatt im Zentralbereich der Universität Bremen verwendet.

Zylinder-Befestigungselemente

Um die Zylinder in geeigneter Weise mit der Ober- beziehungsweise der Unterplatte zu verbinden und ihnen den nötigen Spielraum im Bereich der Winkelgrößen zu geben, wurde nach einem passenden Befestigungssystem gesucht. Hierbei wurden Kugelgelenke als die geeignetste Lösung erachtet und entsprechende Bauteile aus dem Programm der Firma Festo durch Sponsoring beschafft.

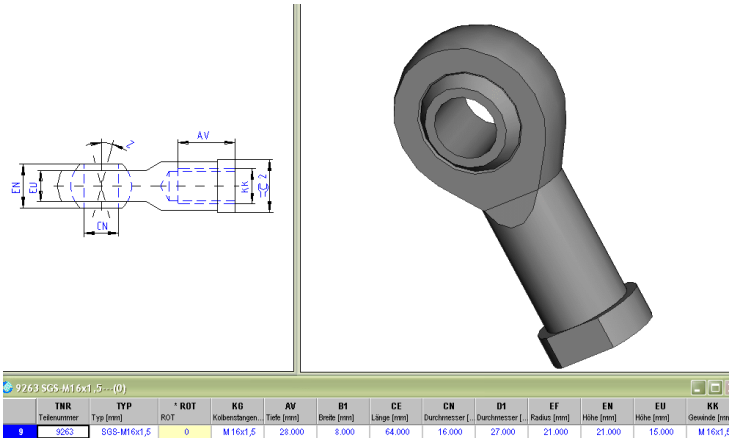


Abbildung 14.25: Technische Daten, Skizze und 3D-Modell des Gelenkkopfs (siehe [**?**])

Das Befestigungssystem umfasst sowohl Kugelköpfe, die direkt auf die Kolbenstangenenden aufgeschraubt werden können, als auch Lagerböcke, die entsprechend zu den Kugelköpfen passen und mit diesen mittels eines Bolzens verbunden werden können. Die Bolzen werden durch Gegenstücke vor dem Herausrutschen gesichert.

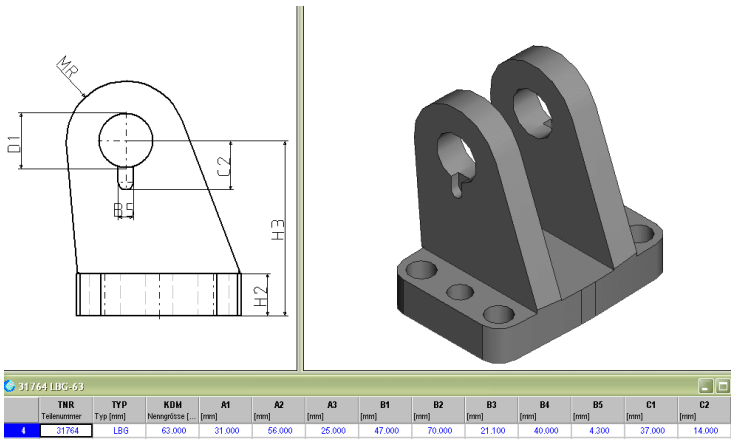


Abbildung 14.26: Technische Daten, Skizze und 3D-Modell des Lagerbocks (siehe [**?**])

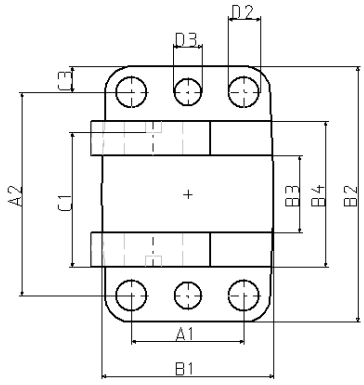


Abbildung 14.27: Draufsicht auf den Lagerbock (Skizze) (siehe [**?**])

Dank dieses Befestigungssystems bleibt die Plattform trotz ihrer gesteigerten Komplexität im Vergleich zur ersten Variante portabel, da die Zylinder auf einfacher Weise durch Herausnehmen der Bolzen von der Plattform zu trennen sind.
Die Lagerböcke wurden auf den Diagonalen der Ober- beziehungsweise der Unterplatte montiert, was den Vorteil hatte, dass die Kugelköpfe nicht so starken Winkelgrößen ausgesetzt sind.

Schaltlogik-Elemente

Um die Zylinder in der gewünschten Form, wie unter Kapitel 14.1.3 erläutert, einerseits zur Dämpfung andererseits für „**[forcefeedback??]**“-Events nutzen zu können, wurden Magnetventile und „Oder-Glieder“ verwendet.
Die verwendeten Magnetventile sind alle baugleich, auch wenn jeweils vier „**[5-2-wegeventil??]**e“ und „**[3-2-wegeventil??]**e“ benötigt wurden. Dies ist zurückzuführen auf die Tatsache, dass sämtliche Bauteile durch Sponsoring beschafft wurden, stellt aber kein Problem dar, da „**[5-2-wegeventil??]**e“ durch Sperrung eines Anschlusses leicht zu der anderen Variante umrüstbar sind.
Die für den Anschluss der Zylinder benötigten 6er-Schläuche wurden mittels „QuickStar“-Schnellverbindungsanschlüssen an die Zylinder bzw. die Magnetventile angeschlossen. Zusätzlich wurden die Magnetventile mit speziellen Schalldämpfern versehen, um den Geräuschpegel möglichst gering zu halten.
Die Magnetventile mit der Typbezeichnung L0502AAWR stammen von der Firma AV Pneumatik Germany und werden mit 24V$_{dc}$ geschaltet. Sie werden durch das verwendete „**[SIOS??]**“ durch entspre-

chende Verschaltung angesteuert. Ein entsprechender Schaltplan wird später erläutert.

Die verwendeten „**[oderglied??]**er“ OS-1/4B der Firma **[festogls??]** wurden ebenfalls mittels „QuickStar“-Schnellverbindungsanschlüssen für 6er-Schläuche in den Schaltkreis integriert und sorgen selbständig für einen zügigen Wechsel zwischen Dämpfungs- und „**[forcefeedback??]**“-Modus.

Druckregelventile

Um den Druck innerhalb der beiden Schaltkreise regeln zu können, werden zwei Druckregelventile entsprechend des obigen Schaltplans benötigt, um sicherzustellen, dass ein Kreislauf mit einem hohen Druck („**[forcefeedback??]**“-Kreis) und ein zweiter mit einem möglichst sehr niedrigen Druck (Dämpfungs-Kreis) arbeiten können. Bereits in der ersten Version der zweiten Realisierung der Plattform wurden Druckregelventile verwendet. Jedoch war mit diesen der Druck nicht genau einstellbar.
In Kapitel 14.1.3 wurde zudem bereits auf das Problem eingegangen, dass der Schaltkreisdruck bei Belastung durch den Benutzer ansteigt. Zu diesem Zweck wurden verschiedene Überlegungen beleuchtet. Nachdem eine Überlegung verworfen wurde, einen Gaskanister als zusätzlichen Drucktank anzuschließen, der Druckschwankungen hätte vermindern können, wurde ein spezielles Druckregelventil mit Rückentlüftung über die Firma Ahlrich Siemens beschafft. Dieses Ventil entlüftet nicht nur einen Schaltkreis bei Überdruck, sondern bot zudem die Möglichkeit, die Druckstärke sehr fein einzustellen. Damit war es möglich, für den Dämpfungskreis auf 0,3-0,4 bar einzustellen, für den „**[forcefeedback??]**“-Kreis wurde ein Druck von 5 bar gewählt. Dies stellt eine Besonderheit dar, da die Druckregelventile normalerweise für hohe Drücke (bis zu 16 bar) ausgelegt sind, dafür aber nicht so genau regulierbar sind.

Für die beiden Druckregelventile wurden ebenfalls „QuickStar“-Schnellverbindungsanschlüsse verwendet. Jedoch wurden an diese 8er-Schläuche angeschlossen, also solche, die einen höheren Durchfluss ermöglichen. Dies geschah, damit sichergestellt ist, dass eine hohe Luftmenge in das System aufgenommen werden kann.
Bei den beiden Druckregelventilen handelt es sich um die **[festogls??]**-Komponenten LFRN-1/4B (DämpfungsKreis) und LR-D-7-I-MINI („**[forcefeedback??]**“Kreis) (siehe [**?**]).

Elektronikschaltplan

Um die Magnetventile und damit die Pneumatik-Zylinder schalten zu können, wurde eine Schaltelektronik erarbeitet, auf die im Folgenden näher eingegangen wird.

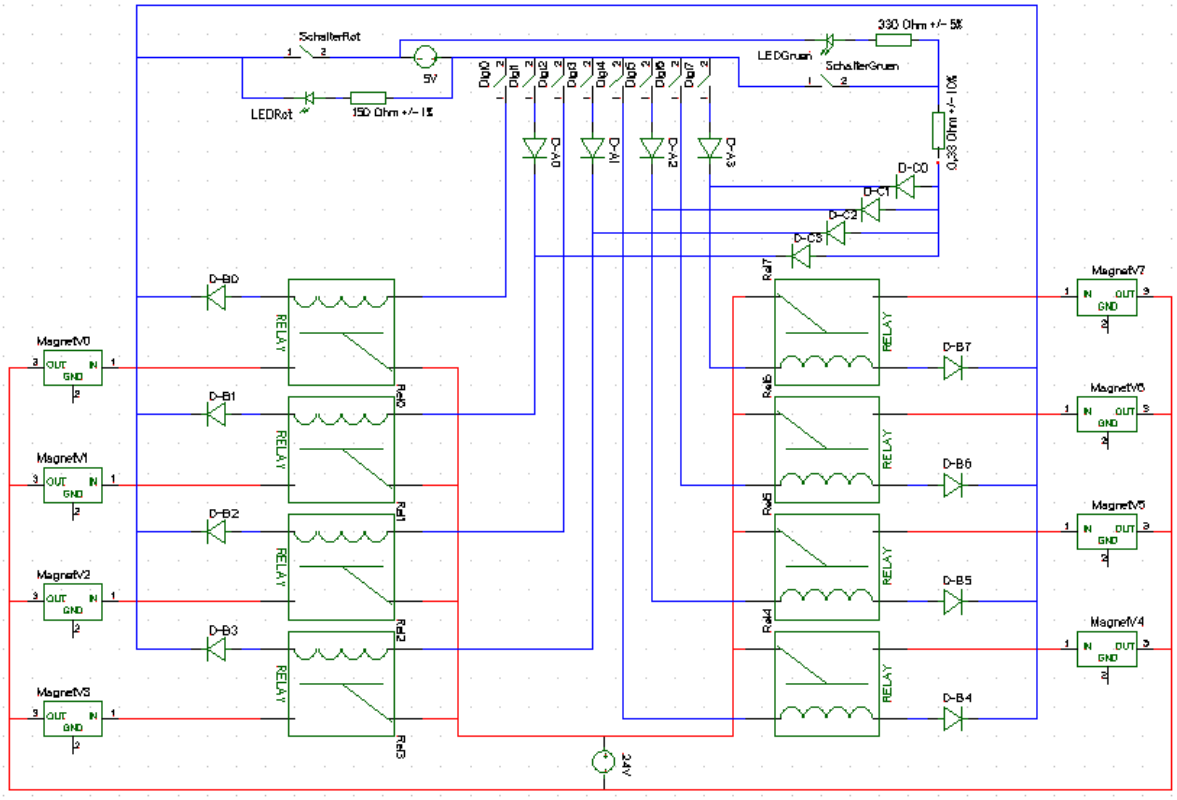


Abbildung 14.28: Elektronikschaltplan

Zunächst sind mit *Dig0* bis *Dig7* die Digitalausgänge des „**[SIOS??]**s“ gekennzeichnet. Diese haben eine Ausgangsspannung V_{out} von $5V_{dc}$ und sind zunächst an Relais angeschlossen. Die Relais schalten, sobald der Strom aus den Digitalausgängen entsprechend anliegt (blauer Schaltkreis). Diese schalten dann $24V_{dc}$, wodurch die Magnetventile *MagnetV0* bis *MagnetV7* mit ihrer Schaltspannung versorgt werden (roter Schaltkreis).
Der $5V_{dc}$-Gleichstrom-Kreis ist hinter allen Relais mit Sperrdioden *D-B0* bis *D-B7* gesichert, um im Falle des Schließens des Schalters *SchalterRot* keine ungewünschte Rückkopplung bzw. der Stromversorgung aller Relais zu verhindern. Zudem sind die Digitalausgänge mit ungeraden Kennziffern *Dig1* bis *Dig7* ebenfalls mit Sperrdioden *D-A0* bis *D-A3* und *D-C0* bis *D-C3* gesichert, um im Fall des Schließens des Schalters *SchalterGrün* eine ebensolche Rückkopplung zu vermeiden.
Mittels der Schalter SchalterRot und SchalterGrün können die Verhalten erzwungen werden, die in Kapitel 14.1.5 erläutert wurden.
Um zudem das Leuchten der LEDs im aktivierten Zustand der Schalter zu gewährleisten, mussten entsprechende Widerstände eingeführt werden, damit die LEDs nicht überlastet werden (max. $2V_{dc}$).

Somit war sichergestellt, dass beim Schließen eines Schalters nur die gewünschten Relais schalten bzw. nicht schalten. Gewollt war in diesem Zusammenhang das Zusammenspiel zwischen dem Schaltkreis für die roten und grünen Schalter. So sollte aus Sicherheitsgründen die „stabile Mittelposition“ der Oberplatte nicht anfahrbar sein, wenn durch den roten Schalter die „**[forcefeedback??]**“-Funktionalität deak-

tiviert ist.

Das folgende Bild zeigt die eigens erstellte Platine mit den Relais, den Sperrdioden und der 24V$_{dc}$ Spannungsquelle.

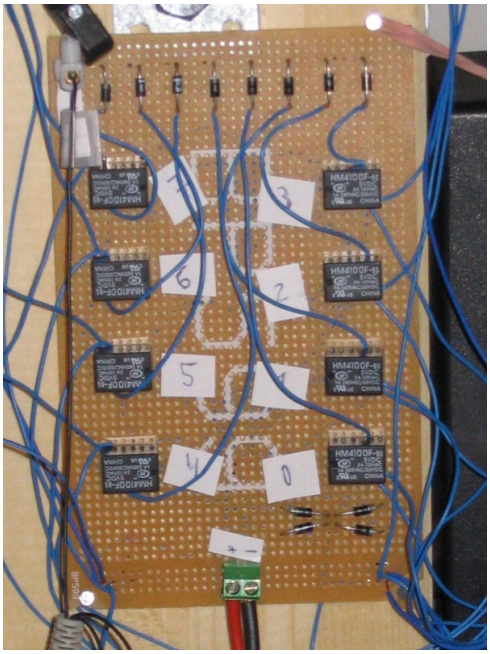


Abbildung 14.29: Eigens erstelle Platine

Simulation des Systems mit FluidSim

Um die Funktionsweise des Systems näher zu bestimmen, wurden umfangreiche Simulationen durch FluidSim (siehe [**?**]) durchgeführt. An dieser Stelle sollen aufbauend auf dem finalen Schaltplan für die Pneumatik alle definierten Zustände beispielhaft dargelegt werden.

Dämpfungsmodus: Der Dämpfungsmodus ist der Zustand, in dem dem Benutzer die Möglichkeit der eigenen Benutzereingabe an das System ermöglicht wird. Dabei ist die durch die Zylinder erzeugte Kraft so gering, dass durch geeignete Verlagerung der Körpermasse des Benutzers eine Neigungsänderung der Plattform-Oberplatte hervorgerufen werden kann.

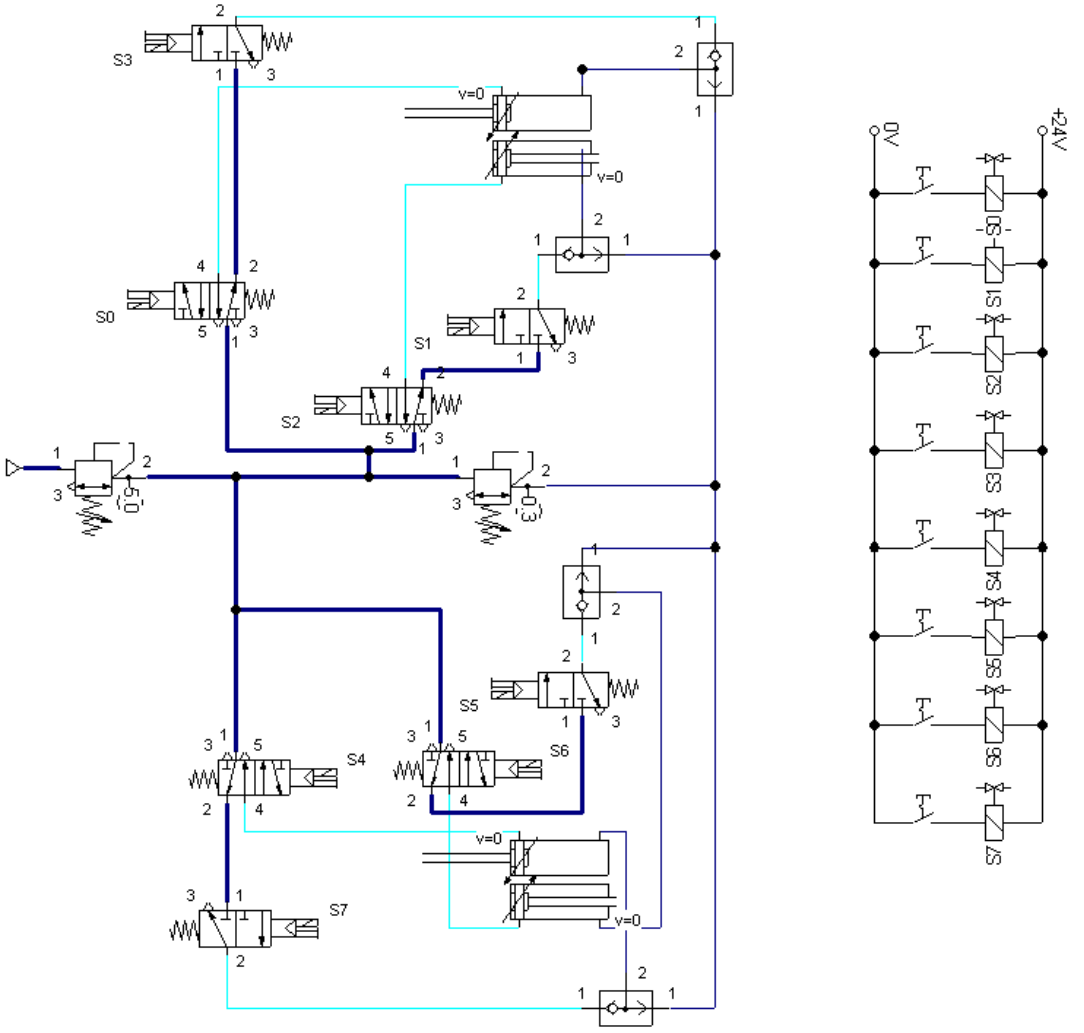


Abbildung 14.30: Dämpfungsmodus

Zu sehen ist fett und dunkelblau dargestellt der hohe Druck, dünn und dunkelblau der schwache Dämpfungsdruck und hellblau die nicht mit Druck beaufschlagten Pneumatik-Leitungen. Die „**[oderglied??]**er“ schalten dabei so, dass die Zuleitungen zu den Zylindern mit dem schwachen Druck beschaltet werden. Die Magnetventile selbst sind alle so geschaltet, dass kein Zylinder mit hohem Druck versorgt wird. Die Schalter an der Seite stehen für die Digitalausgänge des „**[SIOS??]**s“ und sind im Dämpfungsmodus nicht geschlossen.

Stabile Mittelposition: Die stabile Mittelposition wird beispielsweise durch Umlegen des grünen Kippschalters (siehe Kapitel 14.1.5) aktiviert und ermöglicht es dem Benutzer, die Plattform zu besteigen, ohne dass diese bereits seinem Gewicht nachgibt.

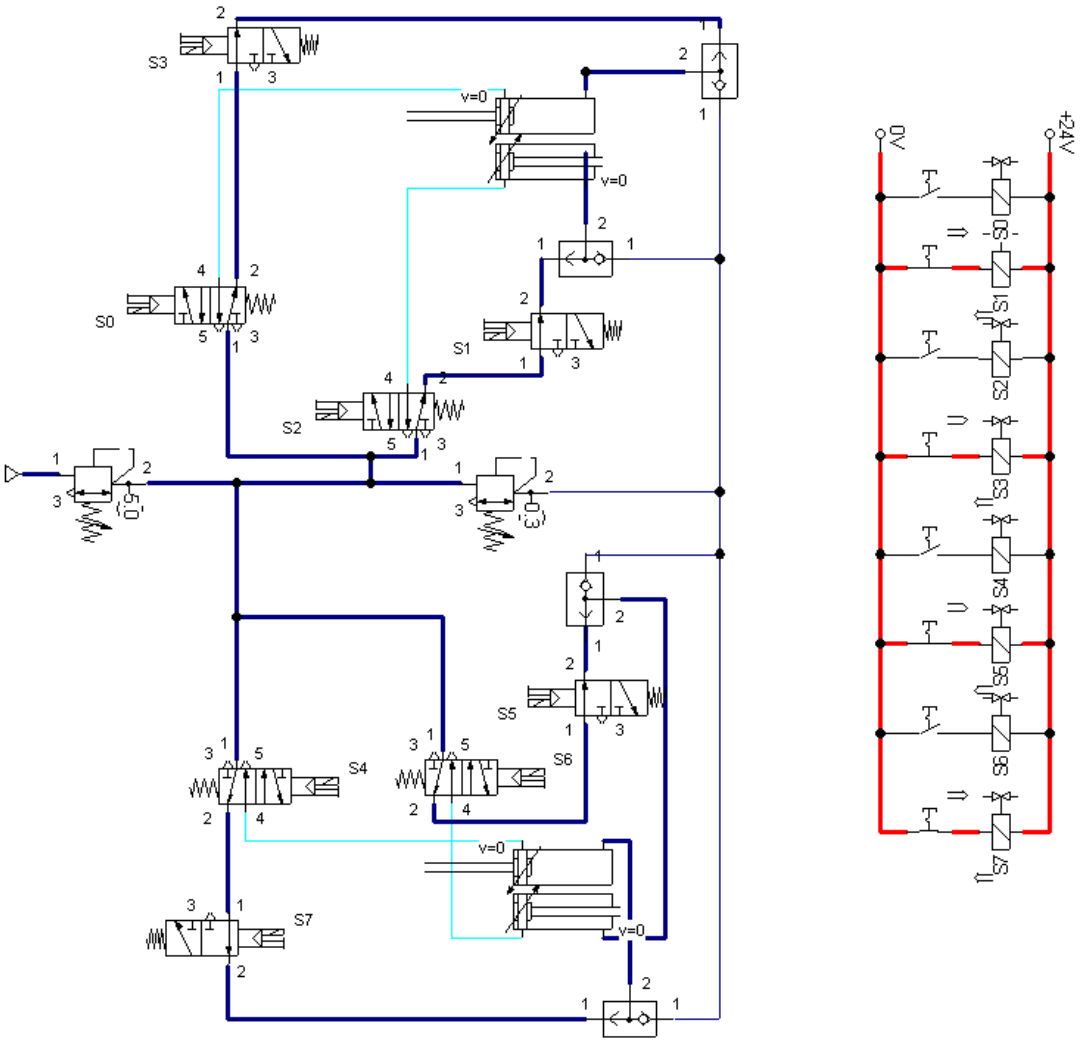


Abbildung 14.31: Stabile Mittelposition

Hierbei werden die digitalen Ausgänge eins, drei, fünf und sieben und die dazugehörigen Magnetventile geschaltet. Dies bewirkt, dass bei jedem Zylinderpaar einer von beiden Zylindern mit hohem Druck eingefahren und der andere ausgefahren wird.

Neigungen nach Vorne, nach Hinten, zur Seite und in die Ecke: Entsprechend ergeben sich folgende Schaltzustände, wenn Neigungen nach Vorne, nach Hinten, zur Seite und in die Ecke gewünscht werden.

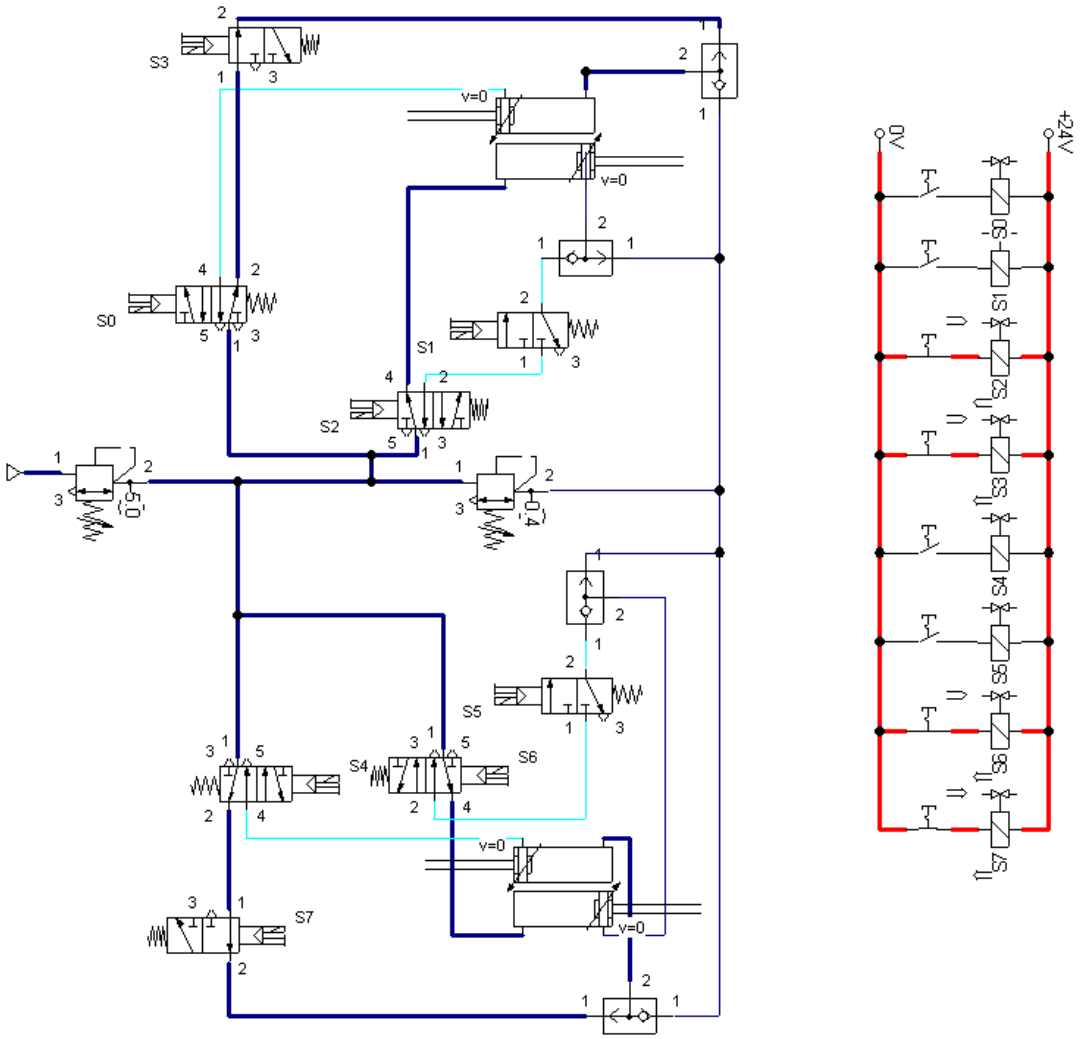


Abbildung 14.32: Neigung nach Vorne

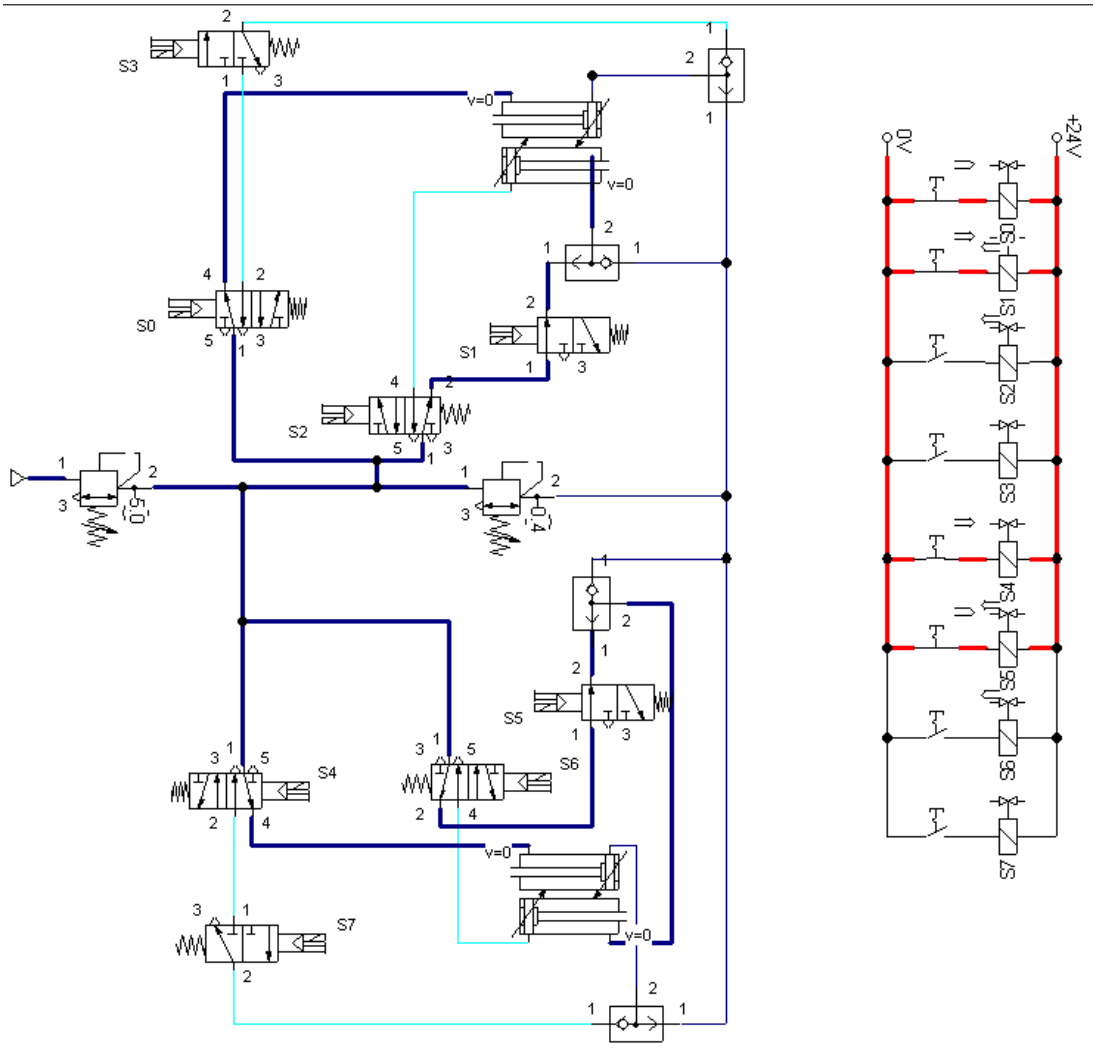


Abbildung 14.33: Neigung nach Hinten

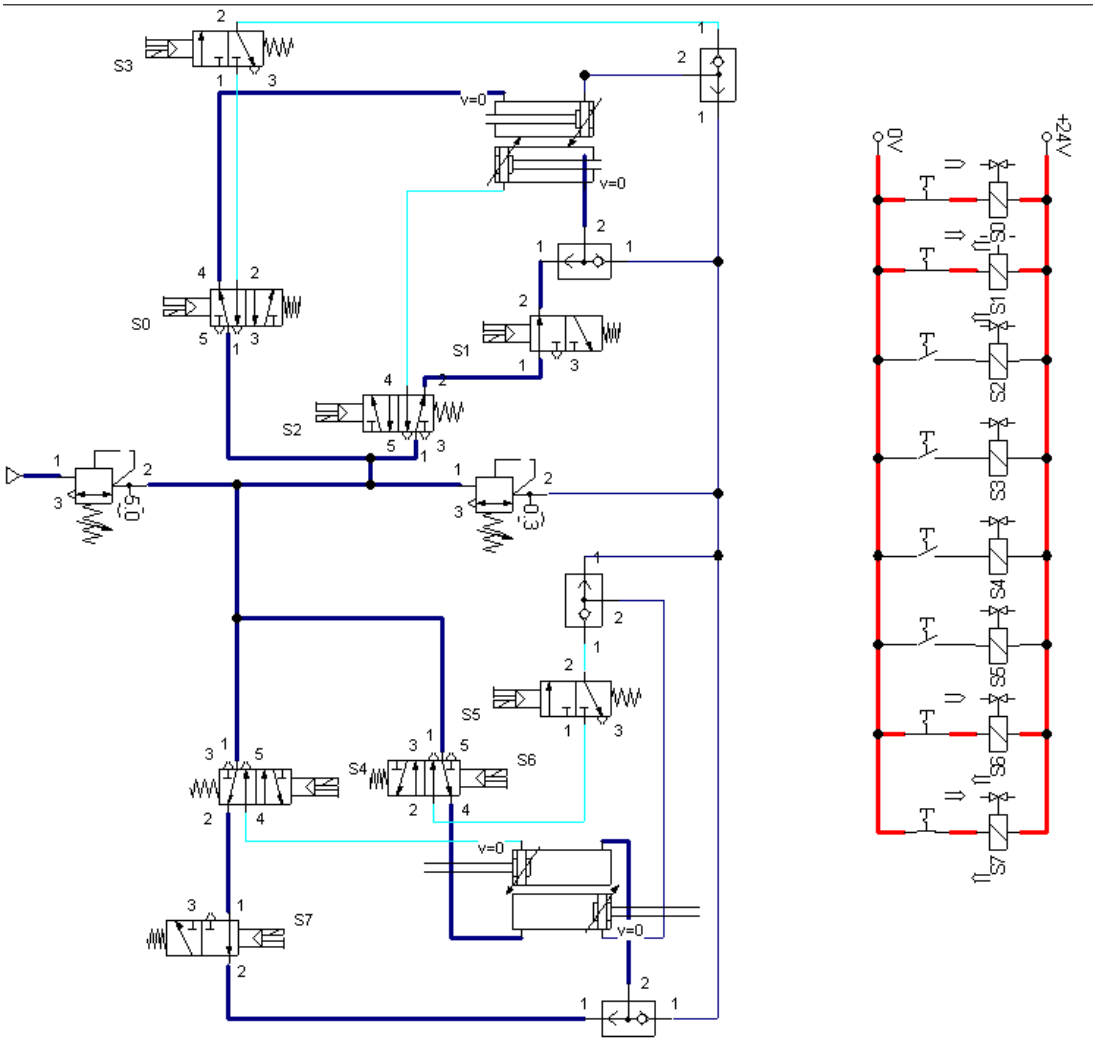


Abbildung 14.34: Neigung zu einer Seite

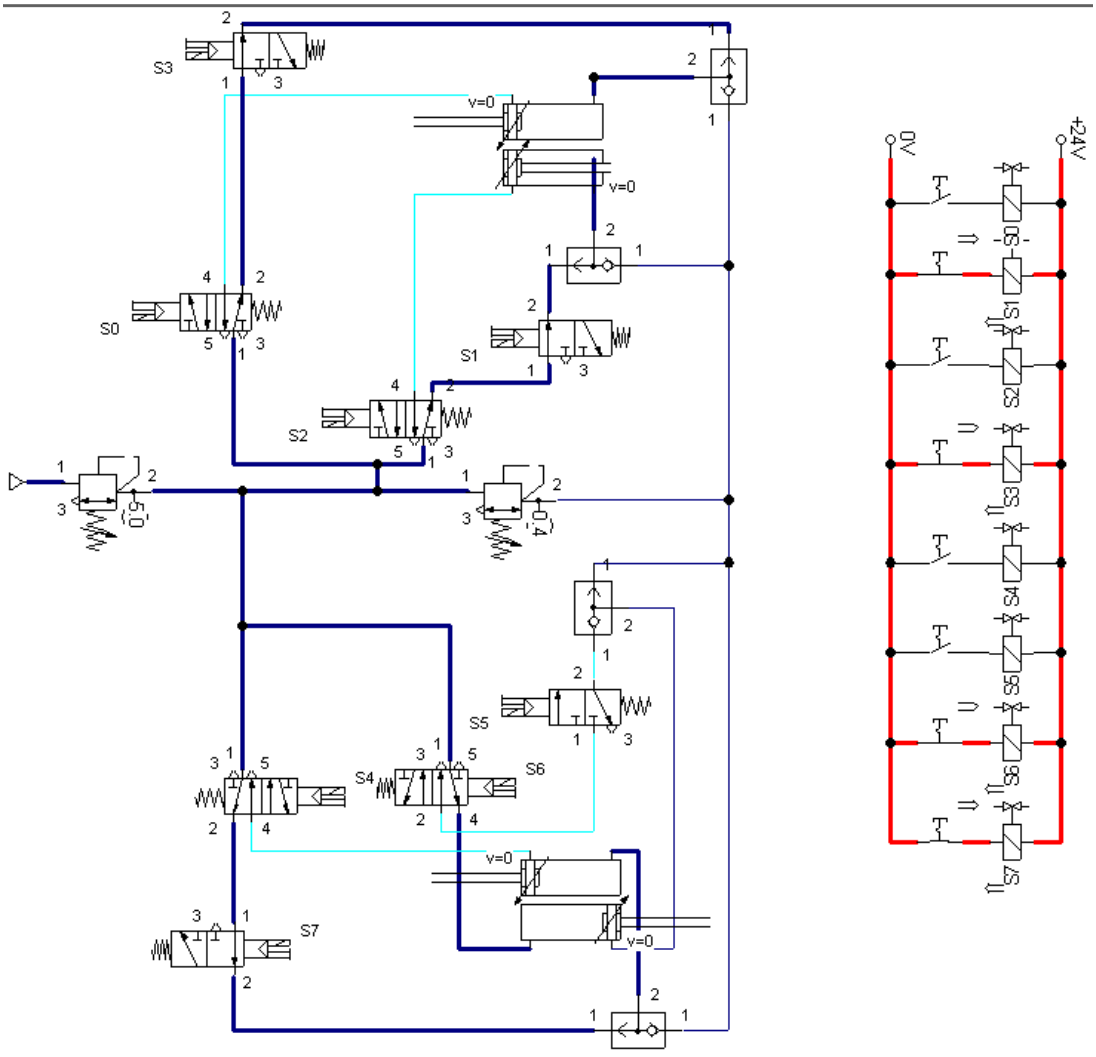


Abbildung 14.35: Neigung in eine Ecke

Simulation des Systems mit Petrinetzen

Zur Visualisierung der Funktionsweise wurde zudem ein Petrinetz erstellt, das das System in seinen verschiedenen Zuständen beschreibt.
Hierbei wurden die Stellen und Transitionen so angeordnet, dass sie die Hardwarekomponenten möglichst real nachbilden.
Simuliert wurden unter anderem die verschiedenen Neigungen der Plattform, der „Notaus“-Schalter, der Dämpfungszustand und das „**[SIOS??]**“ mit seinen digitalen Ausgängen.

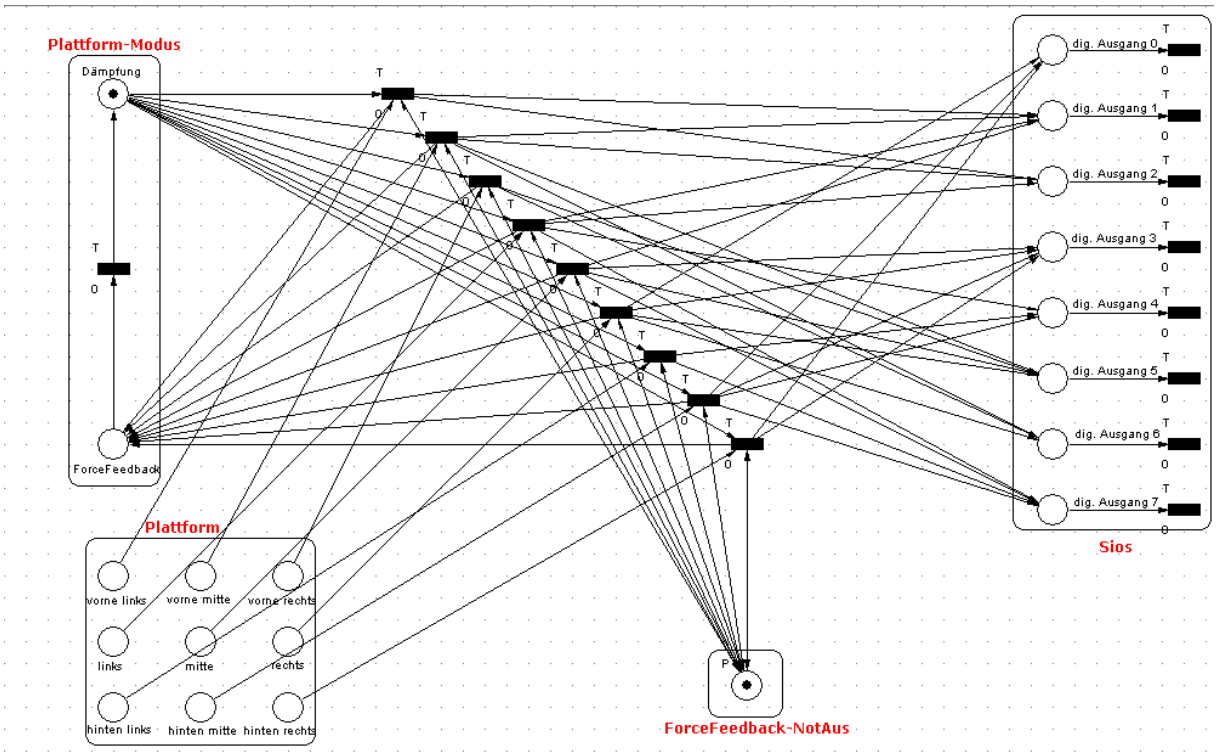


Abbildung 14.36: Simulation des Systems mittels eines Petrinetzes

Hilfestellung für ungeübte Benutzer beim Aufbau

Um Personen, die keinerlei Erfahrung mit der Plattform-Montage haben, jedoch diese einmal aufzubauen wünschen, wurden eigens dafür entworfene Farbcodierungen an Zylindern und den dazugehörigen Druckschläuchen angebracht. Zudem wurden Identifierungsziffern an die Magnetventile und die zu ihnen gehörigen Relais auf der Elektronikplatine aufgeklebt. Auf diese Weise war eine schnelle Zuordnung beim Anschließen der Zylinder bzw. bei einem ungewollten Abstöpseln der Verkabelung möglich.

Außerdem wurden an die Kabelverbindungen geeignete DIN-Stecker angebracht, um einen Fehlanschluss und damit die Zerstörung von Bauteilen zu verhindern. So wurden beispielsweise für die optische Distanzsensoren (siehe Kapitel 14.1.5) dreipolige Stecker verwendet und für die Kippschalter mit LEDs (siehe Kapitel 14.1.5) vierpolige. Zudem wurden für die Taster, die zur Steuerung der Geschwindigkeit dienen (siehe Kapitel 14.1.5), einfache zweipolige Steckverbindungen gewählt.

Im Rahmen der Montage der **[motionplatform??]** „Magic Carpet“ wurden am Projektwochenende, welches Ende März 2004 stattfand, sämtliche Elemente zum Gesamtsystem zusammengefügt. Dazu war es bereits im Vorfeld notwendig, sämtliche Druckverbindungen mit speziellem Material abzuisolieren, die Elektronikplatine gemäß des Schaltplans zu löten, die Zylinder-Befestigungselemente zu bearbeiten, um einen größeren Neigungswinkel zu erhalten und die Zylinder-Verbindungselemente anfertigen zu lassen. So war es möglich, an diesem Wochenende innerhalb zweier Tage einen kompletten Umbau von der Ausbaustufe eins auf Ausbaustufe zwei vorzunehmen (siehe Kapitel 14.1.3 und 14.1.3).

14.1.7 Baupläne des Gesamtsystems

In diesem Kapitel werden detaillierte Informationen zur technischen und physischen Umsetzung der Endrealisierung der **[motionplatform??]** „Magic Carpet“ gegeben. Im Kapitel 14.1.3 wurde bereits auf erste Umsetzungsideen eingegangen, die hier nicht mehr beleuchtet werden, da es sich dabei um frühere Versionen der Plattform handelt. Des Weiteren werden technische Informationen der Sensorik und Aktorik nicht noch einmal aufgegriffen, sondern lediglich die Art und Weise, wie sie in das Gesamtsystem verbaut wurden.

In der folgenden Abbildung 14.37 wird das Gesamtsystem mit seinen Maßen genauer gezeigt.

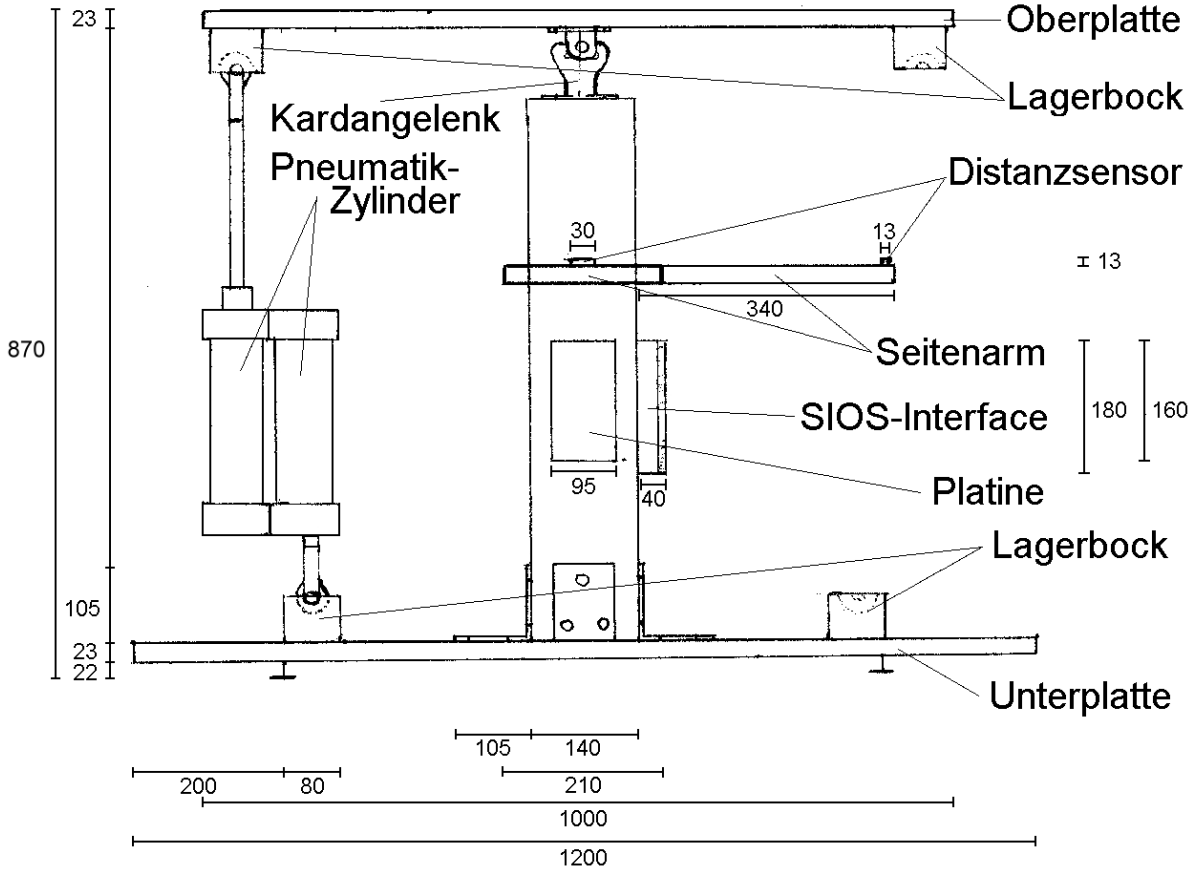


Abbildung 14.37: Technische Zeichnung der **[motionplatform??]** „Magic Carpet“

Zu sehen ist die größere Unterplatte, auf der der stützende Mittelpfosten mittig angebracht ist. Er wurde mit vier Holzschrauben und zusätzlich mit vier Winkeleisen auf seiner Position verschraubt und fixiert. An diesem Mittelpfosten sind das „**[SIOS??]**“ (beschrieben in Kapitel 15.2), die gelötete Elektronikplatine auf Basis des Schaltplans aus Abbildung 14.28, sowieso die beiden Druck-Regelventile (beschrieben in Kapitel 14.1.6). Außerdem sind in geeigneter Höhe die beiden Seitenarme mittels Winkeleisen befestigt, die auf der Oberseite jeweils einen optischen Distanzsensor (siehe Kapitel 14.1.5) und einen Teil der Pneumatik-Schaltlogik (siehe Kapitel 14.1.6) für jeweils ein Zylinderpaar tragen. Der Rest der Pneumatik-Schaltlogik ist jeweils an der Unterseite der beiden Seitenarme angebracht. Dabei sind die Elemente so befestigt, dass sie einen möglichst guten, kurzen Anschluss an den jeweiligen Zylinder ermöglichen.

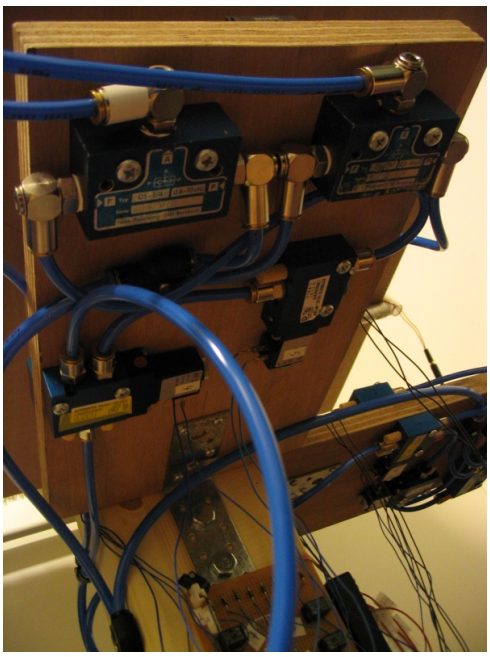


Abbildung 14.38: Seitenarm mit pneumatischen Schaltlogikelementen

Auf die Oberseite des Mittelpfostens ist mittig das Kardangelenk (siehe Kapitel 14.1.3) aufgesetzt und verschraubt, dass wiederum die mit ihm verbundene Oberplatte trägt.

Die Zylinderpaare ihrerseits sind in dieser technischen Zeichnung nur auf einer Seite beispielhaft dargestellt und befinden sich in der Realität jedoch auch auf der rechten Seite der Zeichnung. Wie die Zylinder angebracht und miteinander verbunden wurden, wurde im Kapitel 14.1.6 bereits näher beleuchtet.

Die gesamte Konstruktion ist auf vier Standfüßen gelagert, um einen sicheren Stand während des Plattform-Betriebs zu gewährleisten. Gleichzeitig ermöglicht dieses Standsystem einen einfacheren Transport und hat sich bereits in diversen Realtests bewährt.

Alle verwendeten Komponenten und Schaltpläne wurden bereits in obigen Kapiteln näher beschrieben und sollen an dieser Stelle nicht noch einmal aufgeführt werden.

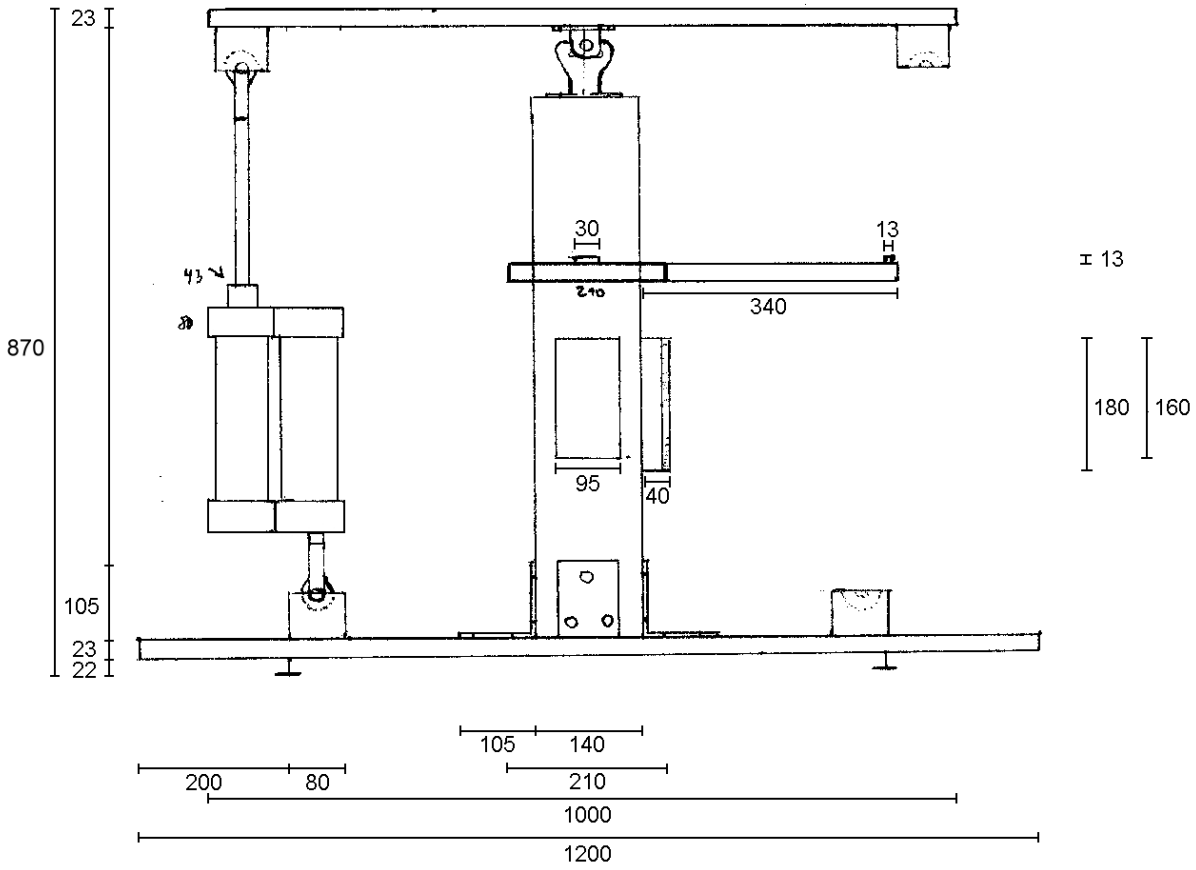


Abbildung 14.39: Technische Zeichnung der **[motionplatform??]** ohne Erläuterungen

14.2 SIOS-Interface

Von allen Komponenten, die wir benötigten, ist das **[SIOS??]** ein sehr bedeutender Teil des Systems, der eine große Rolle in der Kopplung zwischen der “Motion Platform“ und der virtuellen Welt spielt. In einer engen Zusammenarbeit mit einer programmierten Schnittstelle, die auf dem PC lief und mit dem das **[SIOS??]** Daten austauscht, steht das Gerät als Kommunikationsschnittstelle zwischen der Plattform und dem PC. Das Gerät erwartet passende Kommandos, um die Aktivität an seinen Analog- und Digitaleingängen bzw. Digitalausgängen auszuführen und somit die Informationen zur Verfügung zu stellen. Die Informationen beschreiben die Aktivitäten auf der Platform. Die Kommandos sind seine eigentlichen Steuerbefehle, die vom Hersteller AK Modul-Bus Computer GmbH für die Softwareentwickler im Handbuch veröffentlicht wurden (siehe [**?**], die folgenden Abschnitte fassen Teile des Handbuchs zusammen). Unser Zielsetzung war die Eigenschaften des Geräts kennen zu lernen und das Gerät gezielt nach unserem Anforderungen zum Laufen zu bringen. Wir mussten die Funktionalität des Gerätes kennen lernen, bevor wir mit der Entwicklung der Software beginnen konnten.

14.2.1 Beschreibung des SIOS-Interfaces

Es ist ein vielseitiges PC-Interface mit universellen Ein -und Ausgängen. Es besteht aus:

- **8 Digitale Eingängen** haben TTL-Pegel (Logik Transistor-Transistor Pegel) und sind gegen Beschädigungen durch Verpolung und Überspannung geschützt (+/-30V)

- **8 Digitale Hochstrom-Ausgänge**, jeder davon liefert einen Ausgangsstrom von bis zu 600 mA, diese Ausgangsspannung kann sich im Bereich von 5 V bis 12 V einstellen lassen. Mit einem zusätzlichen Schalter kann diese Ausgangspannung als Versorgungsspannung durchgeschaltet werden. Sollte während des Betriebs der Gesamtstrom aller Ausgänge einen Wert von ca. 1,5 A übersteigen, so schaltet die Überlastungsschutz die Ausgänge ab.
- **4 Analoge Eingänge**, von denen haben zwei einen Messbereich von 0 V bis 5 V, wie beim CompuLAB-Interface und bei der miniRS-Box. Zwei andere mit einem Messbereich von 0 V bis 2,5 V stehen für den Anschluss von Sensoren zur Verfügung. Die Standardauflösung beträgt 8 Bit. Wenn die Messungen mit größerer Genauigkeit durchgeführt werden sollen, wird eine Auflösung von 10 Bit gewählt.
- **2 Analoge Leistungs-Ausgänge**. Dieser Teil des **[SIOS??]** wandelt die analogen Werte in digitale, und liefert sie als eine Ausgangspannung zwischen 0 V und 5 V mit einer Auflösung von 8 Bit zurrück. Sind sie mit Leistungsverstärkern ausgerüstet worden, so können sie einen Strom von bis zu 300 mA liefern. Ein kleiner Gleichstrommotor oder auch ein Lautsprecher lassen sich damit benutzen bei einer maximalen Ausgangsleistung des Lautsprechers von ca. 1,5 W.
- **1 Internes Daten-RAM Modul** ist ein 32 Kilobyte großer Datenspeicher, der eine Möglichkeit bietet, eine schnelle Messung durchzuführen und auf ihm abzulegen. Pro Sekunde können bis zu etwa 10.000 Messungen erreicht werden. Diese hohe Frequenz vermeidet eine gleichzeitige Abfrage von PC am **[SIOS??]** und **[SIOS??]** an seinem Port, wobei es passieren könnte, dass der PC die Daten nicht rechtzeitig oder im schlimmsten Fall überhaupt nicht bekommt. So könnte er für immer in einem Pollingverhalten bleiben.
- **Ein Betriebssystem**. Das SIOS-Interface enthält den alten vollständigen Befehlssatz der RS-Box, um auch alte Programme nutzen zu können. Zusätzlich wurde ein neuer Befehlssatz implementiert, der auch schnellere Datenübertragungen erlaubt. Während des Betriebs kann das Betriebssystem eine Messserie von bis zu 5000 Messungen pro Sekunde erfassen und im RAM zwischenspeichern. Die Standardbaudrate beträgt 19200 Baud, kann aber auf bis zu 57600 Baud eingestellt werden.
- **1 Netzteil** zur Stromversorgung (12V/2A).

Außer diesen internen und physikalischen Beschreibungen, gibt es sehr wichtige Teile, die für die Steuerung des **[SIOS??]** und die Kommunikation mit anderen externen Geräten zuständig sind. Das sind die Kommandos. Sie sind die einzige Sprache, welche das **[SIOS??]** versteht, so kann ohne sie keine Kommunikation sauber stattfinden. Nicht sauber, weil man z.B. ohne den Schreibbefehl Werte der Digitalausgänge schalten kann und so das **[SIOS??]** in ein unkontrollierbares Verhalten bringen kann. Solche Zustände können vermieden werden, wenn im Programm zur Kommunikation mit der **[SIOS??]** immer ein entsprechender Befehl der Aktion folgt. Das hat den Grund, dass beim Schreiben auf den Digitalausgang ein solches Verhalten öfter auftritt, wenn zwischen dem Senden vom Schreibbefehl und den Daten ein anderer Befehl gesendet wird. Dieses Problem hatten wir am Anfang und in der Mitte der Projektphase. Wir haben bis zur Projektpräsentation unseren Code so verbessert, dass eine akzeptable Sicherheit der Kommandos gefolgt mit der Aktion gewährleistet ist.

Die Kommandos

Kommandos sind Befehle, die das **[SIOS??]** in ein bestimmtes kontrolliertes Verhalten bringen und es auffordern seine Aufgaben präzise zu erledigen. Nach dem Anschalten befindet sich das **[SIOS??]** im Basismodus und wartet auf gültige Kommandos. Sobald ein solches gesendet wird, ruft das **[SIOS??]** das entsprechende Programm in EPROM auf, und führt es aus. Das ausgeführte Programm unterscheidet anhand der verteilten Rolle jedes Kommandos, die folgendemassen aussehen:

- **1-51h Byte für die Digitalausgabe(81)**. Dieser Befehl schaltet die 8 Bit Digitalausgabe bereit und ermöglicht es alle gesendeten Daten sofort in digitale Werte als Ausgangstrom (600mA) zu interpretieren. In diesem Modus können auch manchmal andere gesendete Befehle als Daten übernommen werden. Als mögliche Lösung dafür sollte man beim Senden des Befehls 0 sicherstellen, dass dieser Modus nicht mehr aktiv ist, weil ein "0-Befehl" alle Aktivitäten des **[SIOS??]** ausschaltet. Ein kleines Beispiel dafür wird später in der programmierten Kollisionsschnittstelle gezeigt.
- **2-B0h Byte für die Eingabe von analogen Werten(176)**. Mit diesem Befehl werden die Analogeingänge als Empfänger von Daten bereit geschaltet, die von einem daran angeschlossenem externen Gerät als Messdaten bereit gestellt werden. Wenn nichts daran angeschlossen ist, liefert die SIOS Standardwerte 2 und 3 für Analogeingang A bzw. B. Alle Messungen werden hier als analoge Werte geliefert.
- **3-D2h Byte für die Messungen, abwechselnd (210)** Dieser Befehl sorgt dafür, dass die Messungen von z.B. den Analogeingängen A und B sich abwechselnd miteinander vertauschen, d.h. die Werte von A werden als Werte von B geliefert und die von B als Werte von A mit ständiger Abwechselung.
- **4-D3h Byte für die Eingabe von digitalen Werten(211)**. Dieser Befehl schaltet die acht Digitaleingänge bereit, wobei die Messdaten nur als digitaler Wert erfasst werden. Jeder von den acht Ports kann nur die Bitwerte 0 oder 1 als Messdaten liefern. Der Wert 0 entspricht keinem Strom, dabei bleiben die grünen LEDs aus. Beim Wert 1 liegt Strom an und die LEDs schalten ein.

Die folgenden Befehle haben dieselben Rollen wie die entsprechenden obigen.

- **5-48h Byte für die Digitale Ausgabe (72)**
- **6-3ch Byte Messung, Kanäle 0 (60)**
- **7-38h Byte Messung, Kanäle 1 (58)**
- **8-01h Byte 201 Identifikation: CompuLAB**
- **9-00h schaltet alle Aktivitäten am [SIOS??] aus.**

Diese Befehlsätze sind kompatibel zur alten Kompaktbox (1 bis 4) bzw. RS und zu CompuLAB-Emulation (von 1 bis 9)

Die meisten Ports der jeweiligen Digital- oder Analogeingänge bzw. Digital- oder Analogausgänge können durch die Verwendung von bestimmten Befehlsätzen auch ohne Einschalten der entsprechenden Kompaktbox-Kommandos angesprochen werden, um Informationen (besonders bei Messungskanälen wie Digital- oder Analogeingänge) zu bekommen. In der folgenden Tabelle ist die Funktionsweise dargestellt:

Senden	Empfangen	Funktion
00h		Unterbrechung für Messungen und laufenden Programmen
01h	10	Versionsnummer 1.0 Abfrage
02h	15	Datum : Tag
03h	8	Datum : Monate
04h	96	Datum : Jahr
05h	Byte	Baudrate ändern: FFh=57, 6kBd, FDh=19,2kBd, FAh=9,6kBd usw.
08h	Byte	Messungsbereich Kanal 0 ändern: 00h=5v, 80h=2,5h , 40h=1,25h
0Bh	Byte	Messbereich Kanal 3

Portausgaben :

Senden	Empfangen	Funktion
10h		Ausgabe Port 4

Porteingaben :

Senden	Empfangen	Funktion
20h	Byte	Lesen Port 1

Analogeingaben , 8 Bit:

Senden	Empfangen	Funktion
30h	Byte	Messen Kanal 0 (A)
31h	Byte	Messen Kanal 1 (B)
32h	Byte	Messen Kanal 2 (C)
33h	Byte	Messen Kanal 3 (D)

Analogeingaben , 10 Bit:

Senden	Empfangen	Funktion
38h	1 höhere Byte niedrige Byte	Messen Kanal 0 (A)
3Bh	1 höhere Byte niedrige Byte	Messen Kanal 3 (D)

Analogausgaben :

Senden	Empfangen	Funktion
40h	Byte	Analogausgabe Kanal 0 (A)
41h	Byte	Analogausgabe Kanal 1 (B)
42h	Byte	Referenzspannung Kanal 2 (C)
43h	Byte	Referenzspannung Kanal 3 (D)

Es gibt noch weitere Befehlssätze, über welche im Internet im Handbuch “Modul-Bus“ nachgeschlagen werden können. Es genügt mit der Suchmaschine “Google“ nach dem Namen “**[SIOS??]**“ zu suchen.

Um für jedes gesendete Kommando das entsprechende Programm nachzuladen und auszuführen, benötigt das **[SIOS??]** Speicherplatz, in dem alle Aktivitäten des Programms gelagert werden. Auf diesem Speicher sind auch Messungen zwischengespeichert, so dass bei der Abfrage diese vom Betriebssystem geholt und verarbeitet werden können, ohne die ordentliche Arbeitweise des Speichers zu stören.

Speicherbereich und Betriebsmodus

Das **[SIOS??]** verfügt über vier unterschiedliche Betriebsmodi mit jeweils eigenen Speichermodellen. Im Modus 0, das generell dem ersten Modus nach dem Einschalten entspricht, sind das EPROM und das RAM im Adressspeicher 0000h bis 7FFFh bzw. 8000h bis FFFFh gespiegelt als Datenspeicher im Bereich 0000F bis 7FFFh. Alle Prozessorunterbrechungen werden im RAM-Bereich ab Adresse 8000h gelenkt. Das Systeminterface verwendet den Timer 0 800Bh und die serielle Schnittstelle 8023h, wobei die Vektoren im EPROM-Bereich zurückgelenkt werden. Deshalb darf der User die RAM-Bereiche von 000Bh-000Ch und 0023h-0025h nicht ändern, es sei denn, dass er die Unterbrechungen für eigenen Zweck umschreiben möchte. Dieser Modus ist kompatibel zum Entwicklungssystem ES535, deshalb können Programme in jeweils passende Speicheraufteilung nachgeladen werden. Die Umschaltung im höheren Betriebsmodus erfolgt durch Programmsprünge zu besonderen Adressen: 7400h, 7800h und 7C00h ordnungsgemäß für Modus 1, Modus 2 bzw. Modus 3. Die Umschaltung bleibt gültig bis die SIOS neu eingeschaltet wird. Eine Rückkehr ist nicht vorgesehen. Die Modi 1 und 2 dienen dazu, Programme in Adressbereich ab 0000h zu laden und zu starten. Der RAM liegt als Datenspeicher jeweils bei Adresse 0, während im Modus 1 RAM als Daten- und Programmspeicher im Bereich 0000h bis 7FFFh angesprochen werden kann. Im Modus2 ist der RAM in zwei 16k-Bereiche aufgeteilt, die als Daten-

und Programmspeicher im Bereich 0000h bis 3FFFh parallel liegen, und jeweils in Bereich 4000h bis 7FFFh gespiegelt werden. Im Modus 3 schaltet ein alternatives Betriebssystem im oberen 16k-Bereich des EPROMs. Dennoch ist der Bereich unbenutzt und für zukünftige weitere Entwicklungen der SIOS vorgesehen.

Die 4 verschiedenen Modi:

- **Modus 0**:

Programmspeicher	Datenspeicher	I/O-Bereich
EPROM	RAM	
0...32k	0...32k	
gespiegeltes RAM: 32k...64k.		

- **Modus 1**: (Umschaltung in Adresse 7400h)

Programmspeicher	Datenspeicher	I/O-Bereich
1/2 RAM	1/2 RAM	32...64k
0...16k	0...16k	
jeweils gespiegeltes RAM: 16k...32k.		

- **Modus 2**: (Umschaltung in Adresse 7800h)

Programmspeicher	Datenspeicher	I/O-Bereich
RAM	RAM	
0...32k	0...32k	
		32k...64k

- **Modus 3**: (Umschaltung in Adresse 7c00h)

Programmspeicher	Datenspeicher	I/O-Bereich
EPROM	RAM	
0...16k	0...32k	
16...32k	0...32k	
		32...64k

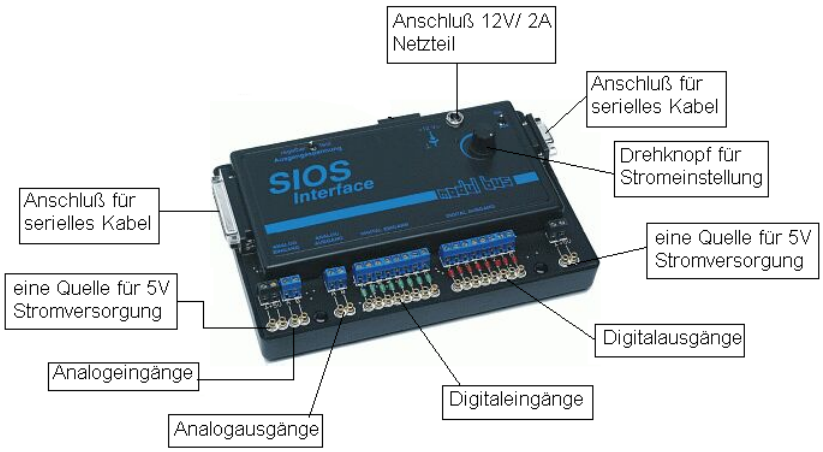


Abbildung 14.40: Das SIOS-Interface

Diese theoretische Beschreibung hat das Gerät mit allen seinen Werkzeugen für die Programmierung vorgestellt. Jedoch nicht konkret gezeigt, wie man überhaupt einen Befehl in einer bestimmten Programmiersprache senden, oder eine Antwort einer Abfrage an SIOS bekommen kann. Nichtdestotrotz genügten für unser Projekt die Informationen über die Befehlsätze und die Konfigurationselemente, um die SIOS mit unserer selbst geschriebenen Software zu steuern. Wir haben mit diesen Informationen ein Programm in C/C++ entwickelt, um das Gerät anzusteuern und werden es in folgenden Zeilen in dem Abschnitt Programmierung im Detail vorstellen.

14.2.2 Programmierung

Für unser Projekt war es die Aufgabe der SIOS-Schnittstelle, einen Informationsaustauchen zwischen dem virtuellen Teppich in der Welt und der Motion-Platform zu ermöglichen. Aus den Aktivitäten der Komponenten sowohl in der realen Welt (wie die Sensoren, die Geschwindigkeitsschalter, die Lüfter und die Relais der Plattform) als auch in der virtuellen Welt (wie den virtuellen Teppich durch die Engine) ergeben sich die Parameter für die Kommunikation, die über das Interface abläuft. Das Programm sollte folgendes tun: über die SIOS die Werte der Sensoren lesen und dadurch die Winkelneigung des Teppichs bestimmen, anschließend die Kollisionseffekte und die Windsimulation in die virtuellen Welt übertragen.

1. Zugriff auf die SIOS:

   Der Zugriff erfolgte durch den seriellen Port des PCs. Jeder angesprochene Port muss mit allen Informationen über die Konfiguration der SIOS, dem Namen des PC-Ports und den auszuführenden Aktionen geöffnet werden. Dafür wird die Variable HANDLE-File unter Windows (unter Linux eine Variable Filedescriptor) hergestellt, indem alle diese Informationen aufgebaut werden. Damit der Zugriff auf mindestens zwei Ports möglich ist, existieren die drei Methoden `OpenPort`, `setConfigurationPort` und `setCommunicatioTimeouts`, deren Parameter eben diesen Informationen entsprechen.

   (a) `openPort()`

   Diese Methode versucht einen PC-PORT zu öffnen. Wenn die Aktion erfolgreich abschließt, wird `“der PC-PORT ist erfolgreich geöffnet”` auf die Konsole geschrieben und `TRUE` als Rückgabewerte der Methode zurückgeliefert; ansonsten `FALSE` mit der Fehlermeldung `“FEHLER: Portöffnung fehlgeschlagen”`.

   Das Erzeugen einer File-Descriptor-Struktur vom Typ `HANDLE` (Windows API) wird für den seriellen Port und die auszuführenden Aktionen mittels `CreateFile()` konfiguriert.

```
HANDLE CreateFile(
   LPCTSTR lpFileName,          // pointer to name of the file
   DWORD dwDesiredAccess,       // access (read-write) mode
   DWORD dwShareMode,           // share mode
   LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                                // pointer to security attributes
   DWORD dwCreationDisposition,  // how to create
   DWORD dwFlagsAndAttributes,  // file attributes
   HANDLE hTemplateFile         // handle to file with attributes to
                                // copy
);
```

   Parametererklärungen:

   `lpFileName`: ist ein Pointer auf einen leeren String, der von Objekten wie Datei, Pipe, Mailhost, Kommunikationsressourcen, Diskgeräte, Konsole oder Verzeichnisse spezifiziert wird. Für das Programm zeigt es auf Kommunikationsressourcen (COM1 für die erste SIOS, COM2 für die zweite SIOS).

`dwDesiredAccess`: spezifiziert die Art von auszuführenden Aktionen auf dem Objekt. Es wurde in `GENERIC_READ` oder `GENERIC_WRITE` gesetzt.

`dwShareMode`: setzt den entsprechenden `sharing mode`, welcher festlegt ob ein weiteres `HANDLE` erzeugt werden darf, wenn ein anderer Prozess bereits Zugriff auf diese Datei hat bzw. wie die Zugriffsrechte dann aussehen. Es wurde auf `0` gesetzt und damit der weitere Zugriff verboten.

`lpSecurityAttributes`: ist ein Zeiger auf eine `SECURITY_ATTRIBUTES` Struktur, über den feststellt wird, ob die HANDLE-Datei an einen Kindprozess vererbt werden kann oder nicht. Es wurde auf NULL gesetzt.

`dwCreationDisposition`: definiert die Aktion, die unternommen wird, wenn die Datei `HANDLE` existiert oder nicht. Dafür gibt 5 Stati: `CREATE_VIEW`, `CREATE_ALWAYS`, `OPEN_EXISTING`, `OPEN_ALWAYS` und `TRUNCATE_EXISTING`. Es wurde in unserem Programm `OPEN_EXISTING` gesetzt.

`dwFlagsAndAttributes`: definiert die Dateiattribute und bezeichnet die Datei. Es wurde auf 0 gesetzt.

`hTemplateFile`: spezifizierte das `HANDLE` mit Lesenzugriff auf die Datei. Es wurde auf 0 gesetzt.

Zusammenzufasst sieht die Portöffnungs- und Herstellungsmethode, wie wir sie konfiguriert haben, mit allen Sicherheitsabfragen, ob der Port geöffnet wurde, folgendenmaßen aus:

```
bool CSiosCom::openPort(char* portname){
  hcom=CreateFile(portname,
    GENERIC_READ | GENERIC_WRITE,
    0,
    0,
    OPEN_EXISTING,
    0,
    0);

  if(hcom==INVALID_HANDLE_VALUE)
  {
    printf("FEHLER: SIOS-Port %s: Öffnen fehlgeschlagen.\n", portname);
    return false;
  }
  printf("SIOS-Port ist erfolgreich geoeffnet: %s\n", portname);
  return true;
}
```

Obwohl der Port geöffnet ist, genügt dies nicht, um eine Kommunikation mit den SIOS durchzuführen. Ein Grund dafür ist, dass die Konfiguration der SIOS in dem Gerätkontrollbock fehlt. Deshalb wurde eine Konfigurationsmethode geschrieben.

(b) `setConfigurationPort(DWORD BaudRate, BYTE ByteSize, DWORD fParity, BYTE Parity, BYTE StopBits)`

Diese Funktion konfiguriert den seriellen Port, auf welchem die Kommunikation mit der SIOS abläuft. An Parametern werden die Baudrate, die Bytegröße, die Bitparität, die Parität und den Stoppbit der SIOS benötigt. Mit diesen Eingaben werden die Informationen über das Gerät mit der Struktur `DCB` mittels der Funktionen `GetCommstate` und `SetCommState` an den seriellen Port übermittelt.

Mit ihnen wird geprüft, ob die Struktur in dem Gerätkontrollblock für die HANDLE-datei tatsächlich gepackt ist oder nicht. (2do)

Dazu werden die Rückgabenwerte von `GetCommstate` und `SetCommState` abgefragt; sind diese `FALSE`, dann liefert die gesamte Konfigurationsfunktion `FALSE` zurück mit der Fehlermeldung, dass die Portkonfiguration fehlgeschlagen ist und die HANDLE-Datei wird geschlossen. Ansonsten wird `TRUE` mit der entsprechender Meldung zurückgegeben.

```
bool CSiosCom::setConfigurationPort(DWORD BaudRate,BYTE byteSize,DWORD
fparity,BYTE parity,BYTE StopBits)
{
  DCB m_dcb;
  if(( GetCommState(hcom, &m_dcb))==0)
  {
    CloseHandle(hcom);
    printf("FEHLER: Verbindung mit Port fehlgeschlagen.\n");
    return false;
  }

  ZeroMemory(&m_dcb, sizeof(m_dcb));
  m_dcb.DCBlength           = sizeof(m_dcb);
  m_dcb.BaudRate            = BaudRate;
  m_dcb.fParity             = fparity;
  m_dcb.fOutxCtsFlow        = false;
  m_dcb.fOutxDsrFlow        = false;
  m_dcb.fDtrControl         = DTR_CONTROL_DISABLE;
  m_dcb.fDsrSensitivity     = false;
  m_dcb.fTXContinueOnXoff   = FALSE;
  m_dcb.fOutX               = false;
  m_dcb.fInX                = false;
  m_dcb.fNull               = FALSE;
  m_dcb.fRtsControl         = RTS_CONTROL_DISABLE;
  m_dcb.fAbortOnError       = false;
  m_dcb.ByteSize            = byteSize;
  m_dcb.Parity              = parity;
  m_dcb.StopBits            = StopBits;

  if(SetCommState(hcom,&m_dcb)==0){
    printf("FEHLER: Portkonfiguration fehlgeschlagen.\n");
    closePort();
    return false;
  }

  printf("Port-Konfiguration abgeschlossen.\n");
  return true;
}
```

Im Handbuch der SIOS wurden alle Konfigurationselemente für die serielle Schnittstelle vorgegeben. Es sind die Folgenden: Baudrate: 19200Bd, Bytegröße: 8 Bit, Stoppbit: 2 Mit den beiden Funktionen `OpenPort` und `setConfigurationPort` kann die serielle Schnittstelle mit der SIOS kommunizieren.

(c) `setCommunicationTimeout(DWORD ReadIntervalTimeout, DWORD ReadTotalTimeou DWORD ReadTotalTimeoutConstant, DWORD WriteTotalTimeoutMultiplier, DWORD WriteTotalTimeoutConstant)`

Diese Funktion legt die gewünschte Antwortzeit für die serielle Schnittstelle fest. Die maximale Antwortzeit wird auf `0` gesetzt, damit die entsprechenden Zugriffe auf die Schnittstelle nicht blockieren.

Es wird mit den vordefinierten Funktionen `GetCommTimeouts` und `SetCommTimeouts` geprüft, ob die Kommunikationszeit an die HANDLE-Datei gebunden ist. Wenn ja, liefert die Kommunikationszeitmethode TRUE zurück, gefolg von der Meldung, dass die Zeit gesetzt ist. Sonst FALSE mit entsprechender Meldung, worauf die HANDLE-Datei geschlossen wird.

```
bool CSiosCom::setCommunicationTimeout(DWORD ReadIntervalTimeout,DWORD
ReadTotalTimeoutMutiplier,DWORD ReadTotalTimeoutConstant,DWORD
WriteTotalTimeoutMultiplier,DWORD WriteTotalTimeoutConstant)
{
  COMMTIMEOUTS comTimeouts;

  if(GetCommTimeouts(hcom,&comTimeouts)==0) return false;

  comTimeouts.ReadIntervalTimeout        =  ReadIntervalTimeout;
```

```
  comTimeouts.ReadTotalTimeoutConstant   =  ReadTotalTimeoutConstant;
  comTimeouts.ReadTotalTimeoutMultiplier =  ReadTotalTimeoutMutiplier;
  comTimeouts.WriteTotalTimeoutConstant  =  WriteTotalTimeoutConstant;
  comTimeouts.WriteTotalTimeoutMultiplier=  WriteTotalTimeoutMultiplier;

  if(SetCommTimeouts(hcom,&comTimeouts)==0){
    printf("FEHLER: Setzen der Timeouts fehlgeschlagen.\n");
    closePort();
    return false;
  }

  printf("CommunicationTimeout gesetzt.\n");
  return true;
}
```

Mit den drei beschriebenen Funktionen konnten die Werte der an die SIOS angeschlossenen Sensoren abgefragt werden.

2. Sensorenwerte lesen und Winkelneigung des Teppichs bestimmen

   Die Laser Distanz Sensoren werden unter der Plattform angebracht. Sie ermitteln die Höhe des Teppichs und leiteten das Ergebnis auf die Analogeingabe der SIOS weiter. Das Programm liest die Analogeingabe der SIOS aus, um den Neigungswinkel des Teppichs zu berechnen.

   Im Programm werden drei Methoden verwendet, um den Informationsaustausch zwischen der SIOS und PC zu realisieren (`writeOnport()`, `readOnportSens()` und `input_sensor()`):

   (a) `writeOnport()` Diese Methode bekommt als Übergabeparameter die gewählten Befehle und schreibt diese in die HANDLE-Datei.

   Im Fehlerfall liefert die `WriteOnport`-Methode den definierten Wert der Variable `WRITE_SUCCESSFULL` (4) zurück. Sonst den Wert `WRITE_OFF` (9) und schreibt eine Meldung in die Konsole mit Ursachenbeschreibung.

```
int CSiosCom::writeOnport(int port)
{
  DWORD wdword=0;
  if(!WriteFile(hcom,&port,1,&wdword,NULL)){
    printf ("FEHLER: SIOS-Port: ID-%d\n", GetLastError());
    return WRITE_OFF;
  }
  return WRITE_SUCCESSFULL;
}
```

   Sobald die SIOS den Befehl ausführt, schreibt sie auf der HANDLE-Datei ihre zu lesende Information zurück.

   (b) `readOnportSens()`

   Mit dieser Methode lassen sich die Werte der Sensoren, die Geschwindigkeitsänderungen und auch der Anschlußstatus der SIOS ermitteln.

   Es wird überprüft, ob eine gültige Verbindung besteht und eine Textmeldung: `“Sios is Connected”` auf die Konsole geschrieben, wenn es sich ums erste Lesen oder den Neuanschluß der SIOS handelt. Im anderen Fall bleibt der Rückgabewert der Lesemethode 0, und auf der Konsole erscheint die Fehlermeldung `“Sios is not Connected”`.

```
int CSiosCom::readOnportSens()
{
  DWORD rdword=0;
  int value=0;
  int readF=0;
  double counter=0.0;
  do{
    readF=ReadFile(hcom,&value,1,&rdword,NULL);
    counter+=0.01;
```

```
  }while((readF==0 || value<=3)&& counter<10.0);
  if(rdword!=0){
    siosStatus=300;
    if(flag1){
      printf("Sios is Connected \n");
      flag1=false;
      flag2=true;
    }
  }else{
    if(flag2 && siosStatus==0){
      printf("Sios is not Connected \n");
      flag2=false;
      flag1=true;
    }
    if(siosStatus>0){
      siosStatus--;
    }
  }
  return value;
}
```

Damit keine anderen Schreib -und Lesenprozesse die zu lesenden Sensorenwerte stören, hatten wir eine zusätzliche Methode `input-sensor` mit einem Port als Übergabeparameter des Analogeingangs eingebaut, deren Aufgabe das Sicherstellen des Zugriffs auf die Sensorenwerten nur für diesen Prozess war.

(c) `input_sensor()`

Mit dieser Methode wird nur auf den Wert jedes einzelnen Port des Analogeingangs bzw. jedes einzelnen Sensors zugegriffen. Da es nur eine HANDLE-Datei gibt, wird auch darauf jede Abfrage und jede Antwort überschrieben.

```
int CSiosCom::input_sensor(int port)
{
  writeOnport(0xB0);
  writeOnport(port);
  return readOnportSens();
}
```

Leider war der Rückgabewert nicht immer die erwartete Sensorenmessung, weil es manchmal vorkam, dass dieser von den Werten der Beschleunigung in der HANDLE-Datei überschrieben wurde. Aber das Problem wurde durch die Methoden `fillringPufferLikeFIFO_A()` und `fillringPufferLikeFIFO_B()` bei der Ausfüllung des Ringpuffers mit der Plattformhöhe beseitigt.

(d) `fillringPufferLikeFIFO_A()` und `fillringPufferLikeFIFO_B()`.

Wir waren auf die Idee angekommen, einen Ringpuffer herzustellen, weil wir das Problem vom Ruckeln des virtuellen Teppichs lösen und auch eine weichstufige Neigungssimulation der Plattform in der virtuellen Welt haben wollten. Das Ruckeln stammte von der sofortigen Weiterleitung des berechneten Winkels von jeder, also auch unstabilen Sensormessung an die Engine. Für jeden Sensor wurde ein Ringpuffer hergestellt, weil sie bei der gleichen Höhe leider unterschiedliche Messung zeigten. Für die Simulation des Ringpuffers hatten wir ein Feldelement mit einer Länge von 40 genommen. Der Zeiger zeigte die aktuelle Position im Feld, die ausgefüllt werden sollte. Für die Berechnung der nächsten Position wurde der Zeiger um 1 inkrementiert und Modulo Länge des Felds neuberechnet.

Jeder Ringpuffer wurde mit den Werten der Höhen der geneigten Plattform ausgefüllt, die die Sensoren gemessen hatten. Aber leider konnten wir keine Standardformel finden, die je nach Sensorenwert die Messung der Höhe generalisieren konnte. Der Grund dafür war, dass die Spurverfolgung der Sensorenmessung in einem 2-Koordinatensystem eine Kurve zeigte. Also war es nötig, je nach Sensorenwert eine Höhe zu bestimmen, die wir in eine Datei geschrieben haben. Da die zwei Sensoren unterschiedliche Messungen zeigten, wurden zwei solcher

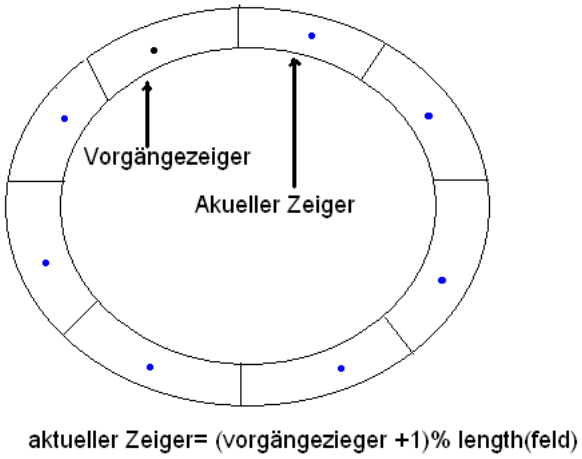


Abbildung 14.41: Schema eines Ringpuffers

Dateien erstellt. Bei dem Aufruf der Methoden `fillringPufferLikeFIFO_A()` und `fillringPufferLikeFIFO_B()` wird in ihren Rümpfen geprüft, ob der eingegebene Sensorenwert größer als 3 ist. Wenn dies der Fall ist, wird für die entsprechende Sensormessung auf das Feldelement dieses Sensors ein Wert geholt und im Ringpuffer geschrieben, der einer bestimmter Plattformhöhe entsprach; im anderen Fall das Gegenteil.

```
void CSiosCom::fillringPufferLikeFIFO_A()
{
  int sens=input_sensor(PORTA);
  if(sens>3){
    valx = sens;
    ringPufferX[posx] = arrays1[valx];
    posx = (posx+1) % MAX_RING_PUFFER;
  }

}

void CSiosCom::fillringPufferLikeFIFO_B()
{

  int sens=input_sensor(PORTB);
  if(sens>3){
    valy=sens;
    ringPufferY[posy] = arrays2[valy];
    posy = (posy+1) % MAX_RING_PUFFER;
  }
}
```

(e) Berechnung des Winkels

Dies ist mit den Methoden `averageValueRingpufferA()`, `averageValueRingpufferB()` und `readDegree()` mit dem Übergabeparameter `averageValue` realisiert. Jede der Methoden `averageValueRingpufferA()` und `averageValueRingpufferB()` holt immer den aktuellen Ringpufferzustand, addiert alle seine Inhalte zusammen, teilt sie durch die Länge des Ringpuffers und gibt das Ergebnis als Rückgabewert weiter an die Methode `readDegree()`. Damit berechnet `readDegree()` den Neigungswinkel der Plattform. Darauf wurde die `cmath` Funktion `atan()` des $Arcus - Tangens$ verwendet, die den tatsächlichen Winkel wie folgt berechnete:

$winkel = (atan((hoehe - (double)averageValue)/abstand) * 180.0)/M_PI;$

wobei

$abstand$ = gesamte Plattformhöhe;

$averageValue$ = (aktueller Ringpufferzustand)/ Größe des Ringpuffers;
$hoehe$ = gemessene Plattformhöhe von Sensoren, wenn die Plattform horizontal ist.

```
double CSiosCom::readDegree(int averageValue)
{
  double winkel=0.0;
  winkel= (atan((hoehe - (double)averageValue) / abstand)*180.0)/M_PI;

  return winkel;
}

void CSiosCom::fillringPufferLikeFIFO_A()
{
  int sens=input_sensor(PORTA);
  if(sens>3){
    valx = sens;
    ringPufferX[posx] = arrays1[valx];
    posx = (posx+1) % MAX_RING_PUFFER;
  }
}

void CSiosCom::fillringPufferLikeFIFO_B()
{

  int sens=input_sensor(PORTB);
  if(sens>3){
    valy=sens;
    ringPufferY[posy] = arrays2[valy];
    posy = (posy+1) % MAX_RING_PUFFER;
  }
}

int CSiosCom::averageValueRingpufferA()
{
  int averagePufferA=0;

  for(int i=0; i<MAX_RING_PUFFER; i++){
    averagePufferA += ringPufferX[i];
  }

  averagePufferA /= MAX_RING_PUFFER;

  return averagePufferA;
}
int CSiosCom::averageValueRingpufferB()
{
  int averagePufferB=0;

  for(int i=0; i<MAX_RING_PUFFER; i++){
    averagePufferB += ringPufferY[i];
  }
  averagePufferB /= MAX_RING_PUFFER;

  return averagePufferB;
}
```

Mit diesen Methoden kann die Schnittstelle bereits eine Aufgabe lösen: den Winkel der Plattform in der realen Welt messen bzw. berechnen und diesen anderen Komponenten zur Verfügung stellen. Aber sie kann noch nicht die Werte der Geschwindigkeitsschalter getrennt von denen der Sensoren liefern, und die Kollision in der virtuellen Welt an die Plattform senden.

3. Werte der Geschwindigkeitsschalter auslesen

   Um die Werte der Geschwindigkeitsknöpfe abzufangen, wurde eine Methode `inputDigit()` mit einer Parameterübergabe in Form des Digitaleingabeports geschrieben. Beim Aufruf sendete sie mittels der Methode `writeOnport()` die Befehle für den Zugriff des digitalen Eingangs und den gewählten Digitaleingabeport an das SIOS. Mit `readOnportSens()` holte sie sofort die Antwort auf die HANDLE-Datei ab und gibt sie als Rückgabewert zurück.

```
int CSiosCom::inputDigit(int port)
{
  writeOnport(0xD3);
  writeOnport(port);
  return readOnportSens();
}
```

4. Übertragung der Kollisionseffekte und der Windsimulation in die virtuelle Welt

   Diese zwei Aktionen sind auf die Schnittstelle durch die Methode `Setdigitoutput` simuliert, die als Übergabeparameter eine Digitalausgabe erwartete. Sobald sie von einem digitalen Ausgabeport aufgerufen werden, erzeugt sie mittels `writeOnport()` einen Ausgabestrom an diesem Port auf die SIOS. Aber dafür sendet sie an den Port den Befehl `0x51`, der den Zugriff auf die Digitalausgänge erlaubt.

```
void CSiosCom::setDigitOutput(int siosOutPort)
{
  writeOnport(0x51);
  writeOnport(siosOutPort);
}
```

Mit allen diesen Methoden ist es möglich, über die Schnittselle auf die SIOS zuzugreifen, und mit allen Komponenten, die angeschlossen sind, Informationen auszutauschen. Aber die Schnittstelle konnte nicht die Koordinationen der Kollisionen aus der virtuellen Welt auf die Plattform simulieren oder den Teppich steuern. Um diese Probleme zu lösen, wurden zwei Komponenten `SensorikInput` und `CollisionPlugin` geschrieben, welche die Fähigkeiten der Schnittstelle nutzen.

`SensorikInput` In dieser Komponente werden die Informationen über den aktuellen Winkel der Plattform und die Geschwindigkeitsschalter verwendet, um das aktuelle Verhalten des Virtuellen Teppichs zu definieren und weiter an die Engine zu leiten. Dies wird erfolgreich durch die Methoden `updateAcceleration`, `updateInput`, `init` und `calculateOrientation` realisiert.

1. `init()`

   Mit dieser Methode wird die SIOS-Schnittsetlle für den Informationsaustausch geöffnet. Damit können andere Komponenten auf die Informationen der an der SIOS angeschlossenen Komponenten zugreifen.

```
bool  CSensorikInput::init()
{
  bool rval = m_Sios.openPort("COM1");
  if( !rval )    return false;
  rval = m_Sios.setConfigurationPort(19200,8,TRUE,NOPARITY,TWOSTOPBITS);
  if( !rval )    return false;
  rval = m_Sios.setCommunicationTimeout(MAXDWORD,0,0,0,0); // Timeout setzen
  if( !rval )    return false;
  return true;
}
```

2. `updateInput()`

   Diese Methode muß von jedem PlugIn, welches von `IInput` ableitet, implementiert werden, und es sorgte dafür, dass ein neues Event des aktuellen Plattformzustands für die Engine in der `event_list` eingehängt wird. Dabei wird die Methode `calculateOrientation()` aufgerufen, die das neue Event erzeugt.

```
void CSensorikInput::updateInput(tEventList& event_list, float timeSinceLastUpdate ){

  // neuer event wird in die Eventliste gepackt
  //CSettingContainer::getSetting( eSENSORIK_EXE )
  if(CSettingContainer::getSetting( eSENSORIK_EXE ) )
  {
    cAbs = false;
    event_list.push_back(new CEventAcc( updateAcceleration( timeSinceLastUpdate), cAbs ));
    CEvent* pEvToAdd = calculateOrientation();
    if( pEvToAdd != 0 )
    {
      event_list.push_back( pEvToAdd );
    }
  }
}
```

3. `calculateOrientation()`

   Diese Methode berechnet den neuen Plattformzustand und gibt diesen als Event gekapselt zurück. Dazu wird die Methode der Schnittstelle `readDegree()` mit Parameterübergaben die Rückgabewerte der Methoden `averageValueRingpufferA()` und `averageValueRingpufferB()` aufgerufen, die die vorderen oder hinteren und linken oder rechten Winkeln der Plattform berechnet und sie an die Methode `AngleUnitsToRadians()` weiter gibt. So kann die Methode `CEventOrientationAxis()` mit den Winkeln aus dem Rückgabewert von `AngleUnitsToRadians()` die Flugrichtung auf dem virtuellen Teppich bestimmen und `calculateOrientation` diese Flugrichtung als neues Event zurückgeben.

```
CEvent*  CSensorikInput::calculateOrientation(){
  static float min_alpha = 100.0,
               min_beta = 100.0,
               max_alpha = -100.0f,
               max_beta = -100.0f;

  CEvent* pRValEv = 0;
  float alpha= 0.0;
  float beta = 0.0; //alpha - positiver Wert entspricht der rechten Neigung,
                    //bewirkt, das Teppich nach rechts fliegt. Falls alpha negativ
                    //ist, fliegt Teppich nach links.
                    //beta - positiver Wert entspricht der vorderem Neigung,
                    //bewirkt, das Teppich runter fliegt. Negativer Wert bewirkt
                    //dass Teppich nach oben fliegt.

  m_Sios.fillringPufferLikeFIFO_A();
  m_Sios.fillringPufferLikeFIFO_B();
  alpha = m_Sios.readDegree(m_Sios.averageValueRingpufferA());
  beta = m_Sios.readDegree(m_Sios.averageValueRingpufferB());
  if(flagA){
    const float bremsfaktor = 0.1f;

    alpha = alpha * bremsfaktor * 2;
    beta = -beta * bremsfaktor;

    alpha = Math::AngleUnitsToRadians(alpha);
    beta = Math::AngleUnitsToRadians(beta);

    if (max_alpha < alpha)
      max_alpha = alpha;
    if (min_alpha > alpha)
      min_alpha = alpha;
    if (max_beta < beta)
      max_beta = beta;
    if (min_beta > beta)
      min_beta = beta;

    pRValEv = new CEventOrientationAxis( abs( alpha ) > 0.01f ? alpha : 0.0,
                      abs( beta ) > 0.01f ? beta : 0.0 );
    flagA=false;
```

```
    flagB=true;
  }

  return pRValEv;
}
```

4. `updateAcceleration()`

   Diese Methode verarbeitet alle Informationen über die Geschwindigkeitsschalter, um die Beschleunigung des virtuellen Teppichs in der Vorwärts- oder Rückwärtsrichtung zu bestimmen. Der Wert, der aus der Digitaleingabe `0x20` kam, bestimmt den Vorwärtsflug und derjenigen aus der Digitaleingabe `0x21` den Rückwärtsflug. Die Methode `inputDigit` mit `0x20` lieferte 1 (Vorwärtsflug) zurück, wenn der entsprechende Knopf gedrückt wurde; `inputDigit` mit `0x21` 2 (Rückwärtsflug), ansonsten jeweils 0.

```
float  CSensorikInput::updateAcceleration( float timeSinceLastFrame){

int val0=0,val1=0;
float acc = 0;
const float AccPerSec = 2.0f;
if(((m_Sios.inputDigit(DIGITPORT_1)>0 && m_Sios.inputDigit(DIGITPORT_1)<=3) &&
  (m_Sios.inputDigit(DIGITPORT_2)>0 && m_Sios.inputDigit(DIGITPORT_2)<=3)) &&
  flagB==true)
{
  val0 = m_Sios.inputDigit(DIGITPORT_1);
  val1 = m_Sios.inputDigit(DIGITPORT_2);

  if(  val0 == 1 ) { //Falls Wert ist 1, dann ist digital 0 angeschaltet,
    acc = AccPerSec; //das heisst, dass die Beschleunigung erhöht werden muss
  }
  if( val1 == 2 )    //Falls Wert ist 2, dann ist digital 1 angeschaltet,
  {
    acc = -AccPerSec;          //das heisst, dass die Beschleunigung erhöht werden muss
  }
  if( val0 == 3 || val1==3)
  {
    cAbs = true;
  }
  flagB=false;
  flagA=true;
}
else
{
  flagA=true;
}
return acc;
}
```

Es werden die bool-Variablen `flagB` und `flagA` bei den Methoden `updateInput()` und `calculateOrientatio` verwendet. Ein wichtiger Grund ist, dass beide Methoden auf dieselbe globale Variable `ReadOnPortSens` immer unkontrolliert zugegriffen haben, so dass manchmal eine sehr große Verzögerung bei der neuen Winkelberechnung vorkam. Durch ihre Anwendungen ist das Problem beseitigt, so dass kein neuer Winkel berechnet wird, wenn die Beschleunigung geupdatet wird - ebenso und umgekehrt.

`CollisionPlugin` Mit dieser Komponente werden alle relevanten Kollisionen der virtuellen Welt an die Plattform übertragen. Der Spieler bekommt einen kurzen Anstoß in der entsprechenden Stelle auf die Plattform, wenn er in der virtuellen Welt mit einem anderen Objekt während des Flugs kollidiert ist. Zur Realisierung wurden drei Methoden implementiert: `structTimeCounter()`, `positionToCommand()` und `updateOutput()`.

1. `updateOutput()`

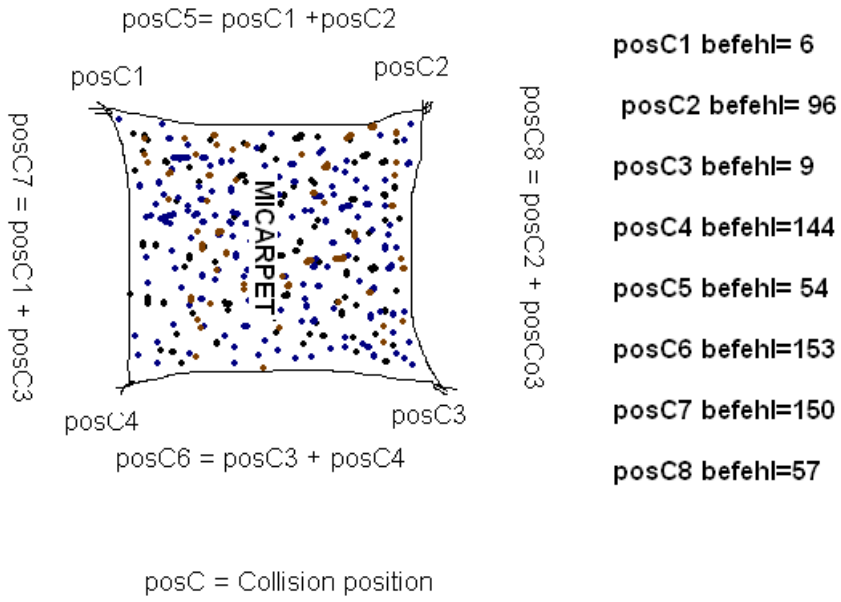


Abbildung 14.42: Kollisions-Position und resultierendes SIOS-Kommando

Die Methode bekommt als Übergabeparameter eine Referenz auf die Eventliste `event_list` und die Zeitdifferenz `timeSinceLastUpdate`. Aus der Eventliste wird die aktuell aufgetretene Kollision geholt und die entsprechende Kollisionsseite auf der Plattform berechnet. Dann gibt sie es an die Methode `positionToCommand()` weiter, wo der Befehl für das Anschalten der Digitalausgaben gewählt wird. Insgesamt gibt es acht Kollisionspositionen. Für jede Position gab es einen bestimmten Wert und einen bestimmten Befehl für das Anschalten des entsprechenden Ausgabestrom auf der SIOS. An jeden Ausgabestromport ist ein Relais angeschlossen, wodurch den Strom für das Anschalten der Druckluft fließt (vgl. 2do).

Es wird die Methode `setDigitouput()` der SIOS-Schnittstelle aufgerufen, die dafür sorgt, dass der Rückgabewert (Befehl) der Methode `positionToCommand()` an die Digitalausgabeporte als Strom gesendet wurde, um den Anstoß des virtuellen Teppichs an der Plattform zu simulieren. Für jede erzeugte Stromausgabe wurde eine Dauerzeit 5ms (`CollisionTime`) gesetzt, auf welche die Zeit `timeSinceLastUpdate` bei jedem Aufruf mittels der `structTimeCounter`-Methode abgezogen wird. Sobald die CollisionTime sich zwischen 0.0f und -0.1f befindete, werden die angeschalteten Ports wieder ausgeschaltet.

```
void  CCollisionPlugin::updateOutput( tEventList& event_list, float timeSinceLastUpdate )
{
  float xKoord, zKoord;
  int positionCollision=-1;
  //std::cout << timeSinceLastUpdate << "\n";
  for( tEventList::iterator i = event_list.begin(); i != event_list.end(); ++i )
  {
    if( (*i)->m_eventID == eEVENT_COLLISION)
    {
      CEventCollision& ec= (CEventCollision&) *(*i);
      xKoord = ec.m_CollisionPos.x;
      zKoord = ec.m_CollisionPos.z;
      // berechnen der position auf dem teppich (also des dig outputs)
      if(zKoord < -PLATTFORM_BREITE/2)
      {
        if(xKoord < -PLATTFORM_BREITE/2)
          positionCollision= 1;
        if(xKoord >= -PLATTFORM_BREITE/2 && xKoord < PLATTFORM_BREITE/2)
```

```
          positionCollision= 5;
        if(xKoord >= PLATTFORM_BREITE/2)
          positionCollision= 2;
      }
      if(zKoord >= -PLATTFORM_BREITE/2 && zKoord < PLATTFORM_BREITE/2)
      {
        if(xKoord < 0)
          positionCollision= 7;
        if(xKoord > 0)
          positionCollision= 8;
      }
      if(zKoord >= PLATTFORM_BREITE/2)
      {
        if(xKoord < -PLATTFORM_BREITE/2)
          positionCollision= 4;
        if(xKoord >= -PLATTFORM_BREITE/2 && xKoord < PLATTFORM_BREITE/2)
          positionCollision= 6;
        if(xKoord >= PLATTFORM_BREITE/2)
          positionCollision= 3;
      }
      csios->m_Sios.setDigitOutput(positionToCommand(0));
      CollisionTime = COLL_TIME;

      csios->m_Sios.setDigitOutput(positionToCommand(positionCollision));

    } // ENDE: if( (*i)->m_eventID == eEVENT_COLLISION)
  }

  structTimeCounter(timeSinceLastUpdate);

  if(CollisionTime < 0.0f && CollisionTime> -0.1f)
  {
    csios->m_Sios.setDigitOutput(positionToCommand(0));
    CollisionTime =0.0f;
  }
}
```

2. `positionToCommand()`

   Diese Methode wählt für eine bestimmte Kollisionsposition den passenden Befehl für das Anschalten des Digitalausgabeports auf das SIOS-Interface und gibt diesen zurück.

```
int CCollisionPlugin::positionToCommand(int collisionEvent)
{
  int befehl;
  if(collisionEvent==1){//Kollision vordere spitze links
    befehl=6;
  }else
  if(collisionEvent==2){//Kollision vordere spitze rechts
    befehl=96;
  }else
  if(collisionEvent==3){//Kollision hintere spitze rechts
    befehl=9;
  }else
    if(collisionEvent==4){//Kollision hintere spitze links
    befehl=144;
  }else
    if(collisionEvent==5){//Kollision auf ganze vordere seite des Teppiches
    befehl=54;

  }else
    if(collisionEvent==6){//Kollision Kollision auf ganzen hintere Seite des  Teppiches
    befehl=153;

  }else
    if(collisionEvent==7){//Kollision Kollision auf ganzen linke Seite des Teppiches
    befehl=150;
  }else
    if(collisionEvent==8){//Kollision Kollision auf ganzen rechte Seite des Teppiches
    befehl=57;
```

```
  }else{
    befehl=0;
  }

  return befehl;
}
```

3. `structTimeCounter()`

   Diese Methode zieht auf die gesetzte `CollisionTime` die Zeit seit der letzten Update `timeSinceLastUpdat` ab und prüft, ob `CollisionTime` größer als 0 war.

```
void CCollisionPlugin::structTimeCounter(float timeSinceLastUpdate )
{
  if(CollisionTime >= 0){
    CollisionTime -= timeSinceLastUpdate;
  }
}
```

14.3 Geschwindigkeitssteuerung und Windsimulation

Unmittelbar nach der Bildung der Teilgruppen haben wir uns Gedanken darüber gemacht, was unsere Gruppe leisten soll. Als unsere Hauptaufgabe galt die Realisierung der Steuerung des Systems (Fliegender Teppich) mittels Plattform und anderen Eingabegeräten. Außerdem sollten wir verschiedene Spezialeffekte (z.B. Wind, Nebel u.s.w.), die die Simulation realitätsnäher erscheinen lassen, einbauen. Um unsere Aufgaben nach Schwierigkeitsgrad zu differenzieren und damit unsere Kräfte auf die verbliebenen drei Semester zu verteilen, haben wir einen 3-Stufen-Plan aufgestellt. Im Sommersemester 2003 sollten wir die Stufe eins erfüllen. Dazu zählt folgendes Vorhaben: Geschwindigkeitssteuerung an die SPS anschließen und die daraus resultierenden Signale interpretieren. Dies soll dem Benutzer mit zwei Druckknöpfen einfaches Gasgeben und Abbremsen möglich gemacht werden.

Zur Geschwindigkeitssteuerung wollten wir zuerst einen Telegraphen einsetzen, der aus dem maritimen Umfeld stammt. Grundidee ist in diesem Falle, einen mechanischen Hebel so zu manipulieren, dass dieser Signale zur Beschleunigung an die SIOS sendet.
Diese Variante der Geschwindigkeitssteuerung konnte an die Plattform nicht angepasst werden. Dann wollten wir die Beschleunigung mittels einfacher Gaspedale der Firma Thrustmaster umsetzen. Später haben wir uns jedoch aus dem Grund, dass die Gaspedale zum fliegenden Teppich einfach nicht passen, anders entschieden und zwei einfache Druckschalter zur Geschwindigkeitssteuerung benutzt.

Im Wintersemester 2003/2004 wurden durch die Sensorik-Gruppe die Geschwindigkeitsreglung, wie bereits erwähnt, über zwei digitale Schalter, die die Fluggeschwindigkeit erhöhen bzw. erniedrigen, realisiert. Diese wurden in Fahrradlenkradhörner eingebaut (Abbildung 14.43). Letztere wurden an der Plattform angebracht und dienen nebenbei auch als Haltegriffe.

Die Software zur Geschwindigkeitsmanipulation ist als ein eigenes Modul umgesetzt worden, was der leichteren Wartung und besseren Übersicht dient. Die Realisierung des Flugwindes war komplexer, sowohl bei der Planung als auch bei der Realisierung. Als Erstes ist die Suche bzw. Auswahl des geeigneten Luftstromerzeugers zu nennen. Anfangs entstand eine Idee, die Druckluft nutzen zu wollen. Pneumatische Winderzeugung haben wir aber wegen enormer Geräuschentwicklung abgelehnt. Axiallüfter, die zur Kühlung in PCs genutzt werden, waren unsere nächste Alternative. Da die Lüfter relativ klein sind (12 cm x 12 cm), hätten wir eine hohe Flexibilität bei der Installation im Cave gehabt. Auf den Abbildungen 14.44 und 14.45 kann man eine Skizze sehen, die eine von uns entworfene Ventilatorenaufstellung schildert. Diese Lüfter erzeugten jedoch nicht genügend Wind und führten zu kostspielige Realisierungsmöglichkeiten. Weiterhin haben wir nach anderen Windsimulationsmöglichkeiten recherchiert.

Abbildung 14.43: Lenkradhörner

Zum Beginn des Sommersemesters 2004 haben wir uns entschlossen, den im Projektraum vorhandenen Standventilator zu benutzen und noch einen Tischventilator für Rückenwinderzeugung zu besorgen. Die Skizzen, die darstellen, wie dies realisiert wurde, kann man auf den Abbildungen 14.46 und 14.47 sehen.

Für die Schaltung der Ventilatoren haben wir einen Schaltkreis (siehe Abbildung 14.48) entworfen und nötige Relais und weitere Bauteile beschafft. Der Schaltkreis ist notwendig, da die Ventilatoren mit 220 V betrieben werden und Ausgänge der Sios lediglich 5 V bieten. Um die Ventilatoren mit Sios-Signalen zu schalten, haben wir 5 V zu 220 V Relais verwendet. Außerdem wurde noch ein zweites Sios-Interface gebraucht, da die Ausgänge der ersten Sios bereits komplett belegt waren. Die Ventilatoren wurden mittels der Schaltung an die erwähnte Sios angeschlossen. So konnten sie durch die Simulation angesprochen werden.
Nun kamen wir zu der nächsten Problematik: der Cave bekam einen Stoffhimmel. Dadurch hatten wir keine Möglichkeit mehr, den vorderen Ventilator direkt über der vorderen Wand des Caves anzubringen. Das Problem wurde wie folgt gelöst: wir haben ein dehnbares, flexibles und sehr leichtes Alu-Rohr besorgt, welches einen passenden Durchmesser hat, um nicht ein allzu großes Loch im Himmel zu schneiden zu müssen und dennoch genügend Luftstrom durchzulassen. Um das Rohr mit dem Ventilator zu verbinden, haben wir eine Stoffmuffe aus einem windfesten Stoff genäht. Sie dient als Übergang von 45 cm (Durchmesser des Ventilators) auf 15 cm (Durchmesser des Rohres) bei einer Länge von 50 cm (Abbildung 14.49).

Der vordere Ventilator wurde dann am rechten Pfosten angebracht. Der hintere Ventilator wurde am hinteren oberen Träger zentral angebracht (Abbildungen 14.46 und 14.47). Zur Steuerung der Ventilatoren haben wir ein Modul `VentilatorPlugin` implementiert. Sie werden digital ein- und ausgeschaltet. Somit haben wir das „Muss-Kriterium“ erfüllt. Das Vorhaben, die Ventilatoren analog gemäß der Geschwindigkeit zu schalten, mussten wir aus Zeitmangel aufgeben, ebenso die Realisierung der Nebeleffekte.

Obwohl sie rudimentären Charakter hat, betrachten wir die Windsimulation als gelungenes Mittel, die

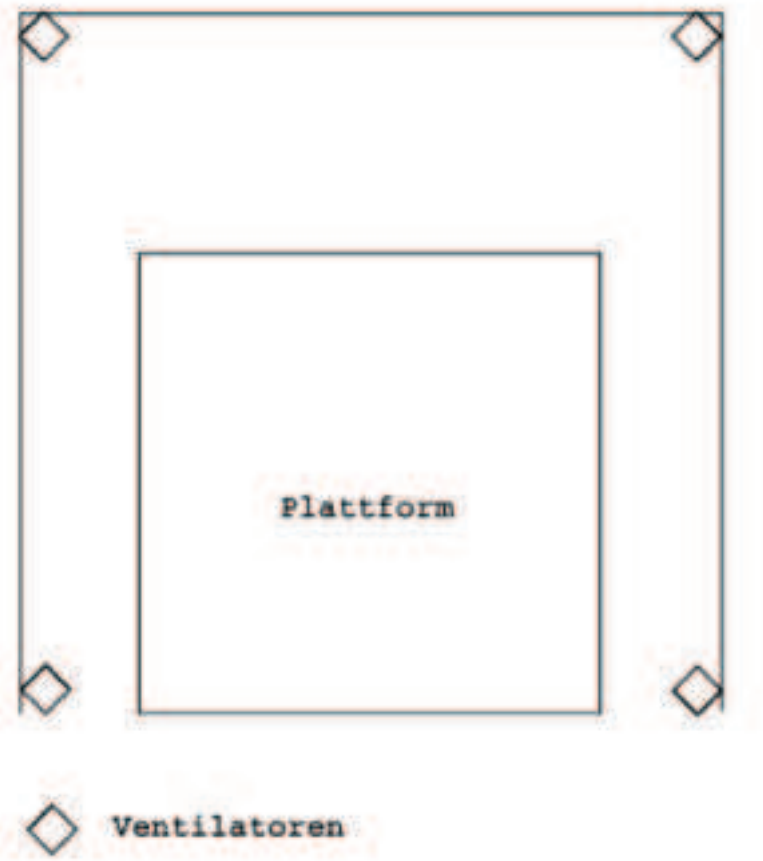


Abbildung 14.44: Entwurf mit Lüfter. Ansicht von oben

CAVE um eine weitere aktorische Komponenten zu ergänzen und somit die Simulation realitätsnäher zu gestalten. Beide Ventilatoren liefern genügend Luftverwirbelung, um den Wind zu spüren. Von der Stärke des Windes sowie der Geräuschentwicklung beider Ventilatoren wird der Benutzer nicht negativ beeinträchtigt.

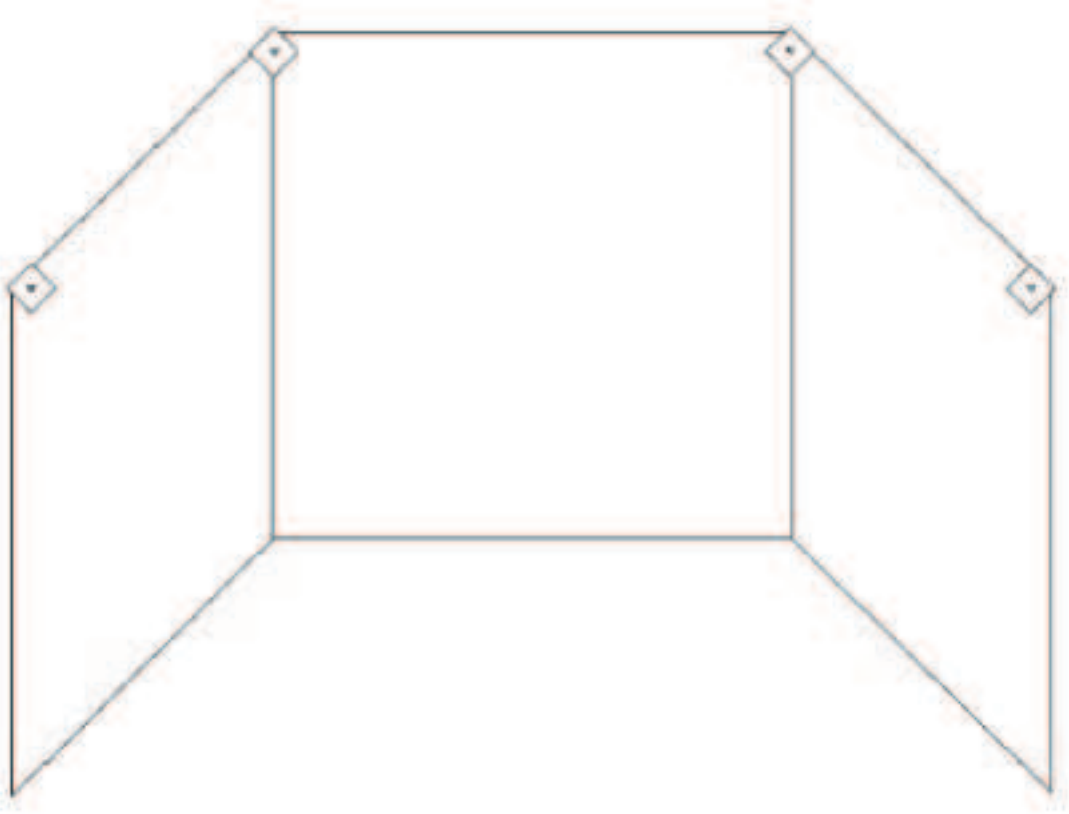


Abbildung 14.45: Entwurf mit Lüfter. Frontale Ansicht. (V steht für Ventilator.)

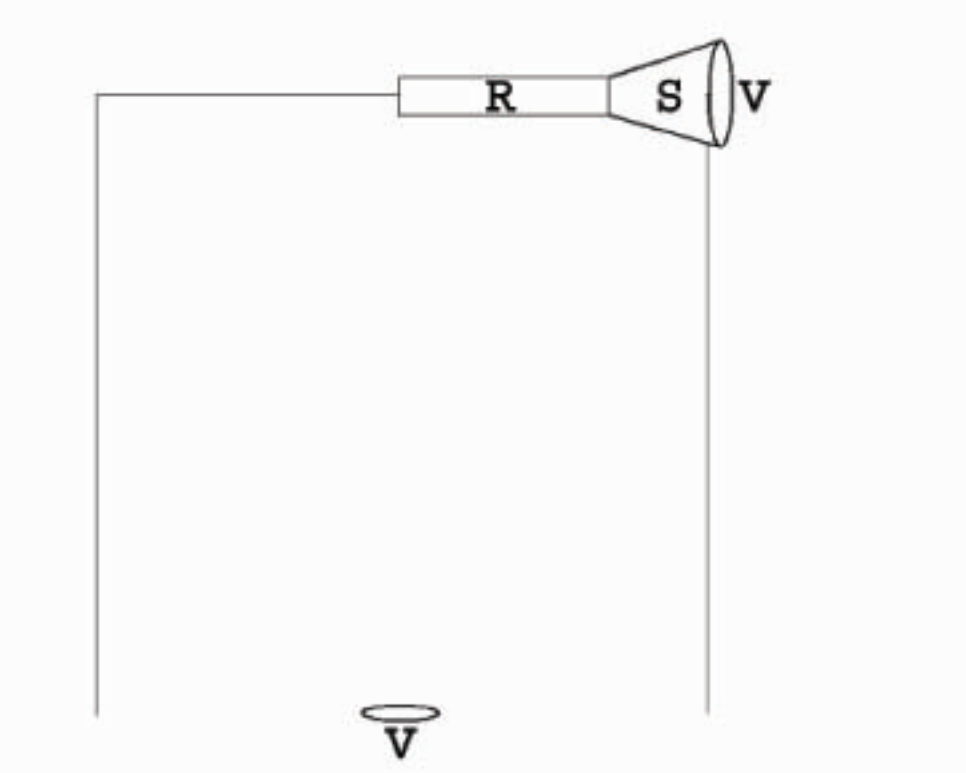


Abbildung 14.46: Endgütige Version. Ansicht von oben. (V steht für Ventilator, R für Alurohr, S für Stoffmuffe)

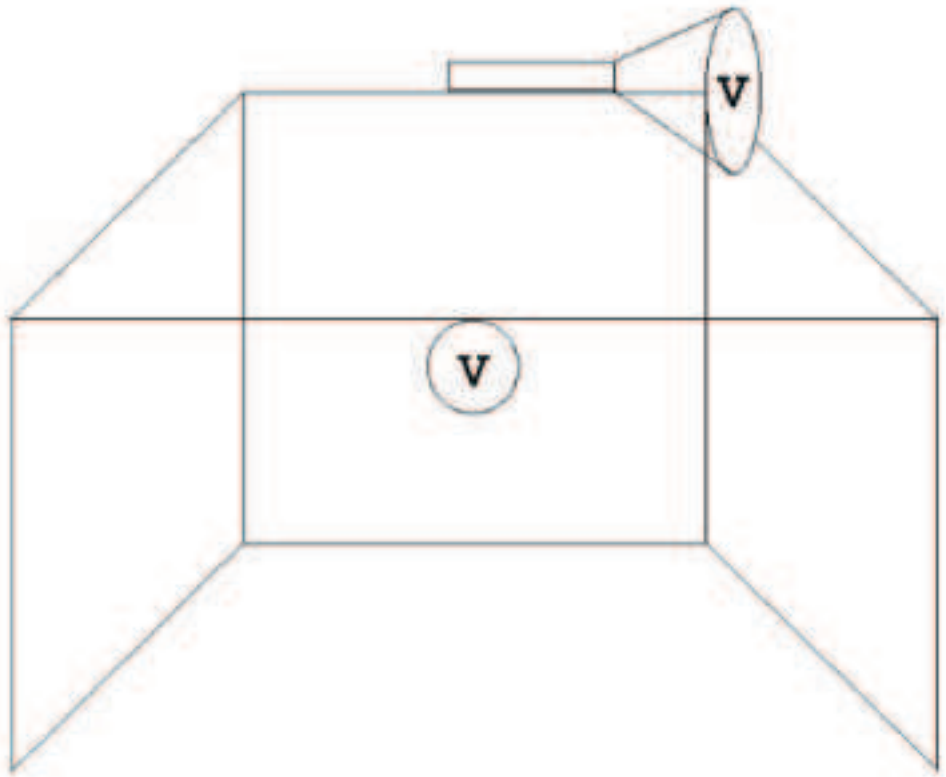


Abbildung 14.47: Endgültige Version. Frontale Ansicht. (V steht für Ventilator.)

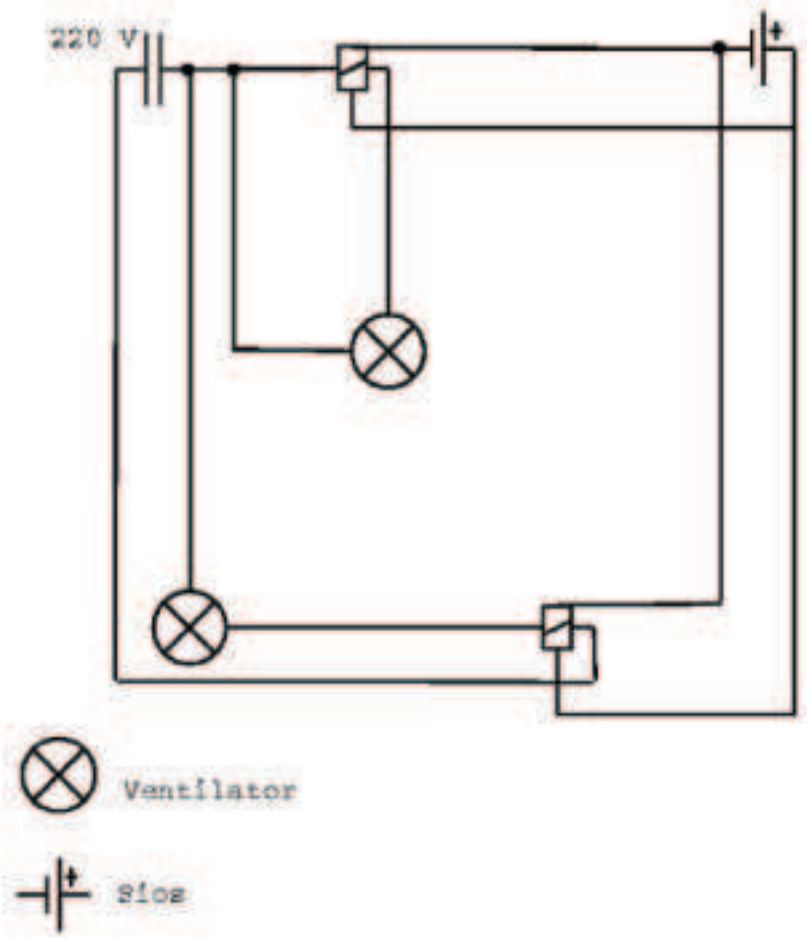


Abbildung 14.48: Schaltkreis.

Abbildung 14.49: Stoffmuffe.

Kapitel 15

Sound

Neben der in einem vorigen Teil beschriebenen Grafikdarstellung im System ist der Sound die zweite Komponente, die zum „virtuellen Teil“ der Mixed Reality beiträgt – daher soll natürlich die Soundausgabe ebenfalls möglichst realistisch wirken. Dieser Teil beschreibt sowohl die verwendete Hardware als auch die Software-Anbindung des von uns verwendeten Systems.

15.1 Hardware

Um unsere Entscheidung für die Sound-Hardware zu begründen, sollen im folgenden Text zuerst einige Verfahren beschrieben werden, welche die Positionierung von Sounds im Raum und somit die Herstellung eines Raumklangs ermöglichen.

15.1.1 Prinzipien

In diesem Abschnitt soll zuerst die Basis aller Soundverfahren für Raumklang kurz beschrieben werden. Das einfachste denkbare Wiedergabeverfahren ist natürlich die Wiedergabe über einen einzigen Lautsprecher (*mono*). Da es nur eine Quelle gibt, ist keine Lokalisierung des Sounds im Raum möglich, für den Zuhörer kommen die Geräusche immer direkt vom Lautsprecher selbst.

Um dies zu verbessern, gibt es eine Vielzahl von Verfahren, die dem Zuhörer eine Ortung der Klänge im Raum ermöglichen. All diesen Verfahren ist gemein, dass sie das psycho-akustische Phänomen ausnutzen, dass das menschliche Ohr bei Unterschieden in Lautstärke (Pegel-Unterschiede) und Verzögerung (Laufzeit-Unterschiede, das Schallsignal erreicht das eine Ohr früher als das andere) eine Ortung des Ursprungs von Schall vornimmt. Alle Verfahren simulieren also diese Unterschiede künstlich und rufen somit beim Hörer eine Lokalisation des Schalls hervor.

Das einfachste und bekannteste Verfahren ist *Stereo*, bei dem zwei Lautsprecher benutzt werden. Durch Abspielen mit unterschiedlichen Pegelstärken oder kleiner Verzögerung kann so die Soundquelle für den Zuhörer im Raum zwischen den beiden Lautsprechern positioniert werden. Ein wichtiger Aspekt ist hier und auch in anderen Bereichen oft die Abwärtskompatibilität, d.h. Stereo-Signale sollen möglichst auch trotzdem noch auf einem Mono-System wiedergegeben werden können, ohne dass dieses eine Konvertierung vornehmen muss.

Stereo erlaubt nur eine Lokalisierung des Schalls auf einer virtuellen Linie, für den wirklichen Raumklang ist aber zumindest eine freie Positionierung auf einer Ebene wünschenswert (etwa für Schallquellen hinter dem Hörer) – am besten natürlich ganz frei im dreidimensionalen Raum (z.B. auch von oben und unten). Dies kann durch das Hinzufügen von weiteren Lautsprechern erreicht werden. Die Wiedergabe

erfolgt dann nach den selben Prinzipien wie bei Stereo, d.h. mit Pegel- oder Laufzeit-Unterschieden, nur dass sich mit einer steigenden Anzahl von Lautsprechern die Gewichtung immer komplizierter gestaltet. Die beliebteste Anordnung ist im Moment die *Quadraphonie*, d.h. die Verwendung von vier Lautsprechern, die jeweils vorne links und rechts sowie hinten rechts und links des Hörers aufgestellt sind und durch entsprechende Gewichtung immerhin eine Lokalisierung des Sounds in der Ebene ermöglichen.

15.1.2 Standards für Surround- und Mehrkanalsound

Die oben beschriebenen Verfahren setzen alle auf mehrere Lautsprecher mit unterschiedlichen Signalen. Um das also realisieren zu können, muss auf jeden Lautsprecher ein eigenes Ausgabesignal ausgegeben werden; somit gibt es einen Kanal für jeden Lautsprecher. Wie man sich denken kann, führt das gerade bei komplexeren Aufstellungen zu einem sehr großen Datenaufwand. Gleichzeitig gilt wie oben beschrieben auch oft die Vorgabe, dass möglichst die Signale auch noch auf einfacheren Systemen abspielbar sein sollten, ohne dass es zu einer großem Beeinträchtigung des entstehenden Sounds kommt (etwa die Wiedergabe von vier Lautsprechersignalen auf einem Stereo-System).

Um zu zeigen, wie dies praktisch ermöglicht wird, wollen wir hier kurz ein paar existierende Sound-Standards vorstellen, die vieles davon verwirklichen.

Dolby Sound Dolby ist ein System, das zur Rauschverminderung im Audiobereich und Mehrkanal-Tonformat eingesetzt wird und mittlerweile ein Standard geworden ist. Dieses Verfahren kennt jeder schon von seiner heimischen Stereoanlage, die z.B. bei der analogen Aufzeichnung von Kassetten spezielle Frequenzen automatisch anhebt, deren Signal ggf. verstärkt und beim Abspielen wiederum um denselben Wert absenkt.

Dolby Surround Hierbei handelt es sich um ein analoges Mehrkanal-Tonsystem. Vier Tonkanäle werden in zwei Tonspuren komprimiert, die anderen zwei freien Kanäle dienen für Effekte. Beim Dolby Surround kommt schon eine höhere Anzahl von Lautsprechern zum Einsatz. Vier Boxen erzeugen den Raumklang, deshalb wird der Surroundsound auch oftmals „4.0“ genannt, was der Anzahl der Boxen entspricht. Zum Einsatz kommen wie beim Stereo jeweils ein Lautsprecher, links- und einer rechts vom Hörer. Im Hintergrund werden dann noch zusätzlich zwei weitere Lautsprecher - die Surroundboxen - links und rechts aufgestellt. Alle angesprochenen Boxen werden standardmäßig mit dem Surroundsound angesprochen, d.h. der Surroundsound wird auf das Stereosignal aufgelegt. Somit ist eine Abwärtskompatibilität zum Stereo-Sound jederzeit möglich. Ein Verstärker teilt das ausgehende Signal auf die entsprechenden Kanäle (positionierten Boxen im Raum) auf. Der qualitative Hörvorteil gegenüber dem Stereo-Klang ist enorm und von jedermann erkennbar. Der Hörende bekommt das Gefühl vermittelt, dass er sich inmitten der Soundkulisse befindet, allerdings erfordert das Verfahren einen Verstärker, welcher das Dolby-Surround-Signal dekodieren kann (Quelle: [**?**]).

Dolby 4.1 Surround Wie der Name schon andeutet, beinhaltet diese Variante den „normalen“ Dolby Surroundsound der schon oben erklärt wurde. Aber als weiteren Zusatz bietet der 4.1 Surroundsound im Gegensatz zum 4.0- eine weitere Komponente. Gemeint ist hierbei ein weiterer Lautsprecher, der externe Bass. Beim Surroundsound dienen die hinteren zwei Surroundboxen nur zur effektiven Unterstützung von Geräuschen und geben deshalb nicht permanent einen Ton von sich. Der Bass, ebenso wie das hintere Boxenpaar, sendet nur bei Bedarf bzw. Unterstützung Signale aus. Bei besonderen Frequenzbereichen soll dieser dienlich sein und somit für mehr Raumgefühl und Dramatik sorgen.

Abbildung 15.1: Verwendetes Boxensystem von der Firma Logitech

Dolby 5.1 Surround In diesem System kommen fünf Kanäle für den effektiven Ton, zuzüglich dem Basskanal wie bei 4.1 zum Einsatz. Eine Anordnung dieser Lautsprecher sieht wie folgend aus: Links - Mitte - Rechts - Hinten Links - Hinten Rechts - Effekt. Die vorderen drei Boxen, ebenso wie die zwei hinteren Surround-Boxen, sind zuständig für die permanente Tonwiedergabe. Sie stellen die Hauptkomponente dieses Surroundsystems dar. Im hinteren Bereich ist der Bass angeordnet, der die Beschallung von Niederfrequenzen und Effekten übernimmt.

Dolby Digital Dieses Verfahren (auch als AC-3 bekannt [**?**]) wird hauptsächlich im Kinobereich eingesetzt und ist ein datenreduziertes, d.h. verlustbehaftetes Mehrkanal-Tonsystem, welches neben den drei Frontkanälen auch noch zwei vollwertige Rückkanäle besitzt. Töne werden optimal im Raum verteilt und lassen die Ortung von Schallquellen wie im realen Leben erscheinen. Beim Dolby Digital Sound wird ein weiterer Lautsprecher, der Subwoofer, in das Raumgefühl integriert. Dieser spezielle Lautsprecher ist für die tiefen Töne zuständig und kann vom Hörer nicht positionell geortet werden. Egal, wo dieser im Raum aufgestellt wird, seine tiefen, lauten Schallquellen werden durch den Raum reflektiert und treffen von allen Seiten auf den Hörer. Ebenso kommt im hinteren Bereich eine weitere zentrierte Schallquelle hinzu. Drei Boxen werden im Vordergrund (Links, Center, Rechts) positioniert und einzeln angesprochen. Hinter dem Hörer werden die anderen drei Lautsprecher aufgestellt und wiederum einzeln angesprochen (Surround Links, -Hinten und Surround Rechts).

15.1.3 Auswahl der Hardware

Bei der Geräteanschaffung stand zuerst die Frage im Raum, wie viele Lautsprecher-Boxen in unserem Cave positioniert werden sollten. Da das Spielvergnügen mit dem unterstützenden Sound möglichst in 3D erreicht werden sollte, musste in jeder Ecke des Caves eine Box vorhanden sein, wodurch das Raumgefühl der Akustik in einer „realen Welt“ schon einmal halbwegs gegeben war.

Also brauchten wir minimal vier Boxen, die aus verschiedenen Komponentensystemen zusammengesetzt sein könnten. Ein 5.1-Dolby-System stand zuerst zur Debatte, wurde aber schnell wieder verworfen, da die Innenmaße des Caves zu minimal für eine sinnvolle Aufteilung eines solchen Soundsystem gewesen wären. Ein Center-Lautsprecher hätte in diesem Fall gar keinen Sinn gemacht und dem Spieler keinen Vorteil in Sachen Geräuschkulisse gebracht.

Wir haben uns schließlich für zwei 2.1-Boxen-Systeme der Marke Logitech entschieden (Logitech X-220). Dieses System enthält jeweils zwei Hochleistungslautsprecher und einen Hochleistungs-Subwoofer mit einer kompletten Gesamtleistung von 64 Watt PMPO. Eine Aufteilung bzw. Positionierung der Boxenpaare war dann im weiteren Verlauf problemlos. Jeweils ein Sound-System war für den vorderen und den hinteren Bereich des Caves zuständig. In jeder der vier Ecken des Caves wurde ein Lautsprecher auf den Boden gestellt, die zwei Subwoofer wurden ebenfalls auf dem Boden positioniert. Allerdings

wurden diese jeweils vor und hinter der Plattform mit der direkten Ausrichtung auf den Spieler aufgestellt. Somit war eine optimale Aufteilung der Boxen in dem Cave gegeben und die quadrophonische Anordnung erlaubte eine 3D-Positionierung von Sounds.

Damit jedoch eine sinnvolle Aufteilung der Geräuschkulisse überhaupt funktionieren kann, muss die benutzte Soundkarte über dementsprechende Fähigkeiten und Ausgangskanäle verfügen. Die Hardware, die uns zu Beginn des Projektes von der Uni zur Verfügung gestellt wurde, erfüllte diese Qualifikationskriterien, so dass eine Neuanschaffung einer Soundkarte nicht notwendig war. Somit war die Hardware für den Sound fast komplett und wurde nur noch durch zwei Verlängerungs-DIN-Kabel komplettiert, die die Boxen mit dem entfernten PC verbinden sollten.

15.2 Software

15.2.1 Entscheidungskriterien für Fmod

Nachdem sich das Projekt entschlossen hatte, sich der Simulation eines fliegenden Teppichs widmen und somit ein Spiel zu implementieren, welches in einer dreidimensionalen Welt spielen sollte, begann die Soundgruppe mit der Abwägung verschiedener Wege, den Benutzer in der CAVE zu beschallen. Wie man aus der Übersicht der verschiedenen Systeme leicht ersehen kann, ist eine Stereo-Beschallung zwar in jedem Fall besser, als völlig auf Sound zu verzichten; allerdings wird dem Benutzer durch im Raum ortbaren Klang, z.B. dem Geräusch eines Schusses hinter dem Benutzer, die Immersion in die Welt erleichtert. So stand beispielsweise recht früh fest, dass der Spieler beschossen werden sollte. Die genaue Art des Beschusses hingegen war noch unklar; doch ob es sich hierbei um schädliche Zauber oder um Kanonenkugeln handeln sollte: es stand fest, dass der Benutzer allein durch seine akkustische Wahrnehmung in der Lage sein sollte, zu wissen, ob der Schuss von vorne, hinten, links oder rechts kam. Dies ermöglicht neben der gesteigerten Atmosphäre eine wesentlich intuitivere Bedienung der Eingabegeräte. Um dies zu ermöglichen, ist mindestens 4.1-Sound erforderlich; d.h., dass es einen Subwoofer für den Bass gibt, und vier Satellitenboxen, die vorne links, vorne rechts, hinten links und hinten rechts vom Benutzer angebracht sind. Nach langer Überlegung entschlossen wir uns dazu, ein 4.1-System zu nehmen, da die Vorteile eines 5.1-Systems (die zusätzliche Center-Box) nicht die unverhältnismäßig hohen Kosten eines solchen Boxensystems rechtfertigten. Wiederum aus Kostengründen entschlossen wir uns aufgrund eines Sonderangebotes dazu, ein „4.2“-System aufzustellen, indem wir zwei baugleiche 2.1-Systeme kauften; d.h. je zwei Stereo-Boxen samt Subwoofer. Das eine Boxenpaar schlossen wir dann an dem Ausgang an, der für die vorderen Boxen zuständig ist, und das andere dementsprechend am „Hinten“-Ausgang der Soundkarte. Das System hatte sich in unseren zahlreichen Tests bewährt. Ursprünglich war geplant, eine Halterung für die Boxen zu bauen, die wir so direkt an die CAVE-Konstruktion hätten montieren können. Aus Zeitgründen hat sich dieses Unterfangen leider als nicht durchführbar herausgestellt.

Nicht ganz so trivial gestaltete sich die Wahl der Audio-Engine. Wir stießen hier auf eine ziemliche Vielfalt an (zumindest für den akademischen Gebrauch kostenfreien) verfügbaren Sound-Engines. Nach einiger Zeit des Recherchierens im Internet, was vor allem durch das Lesen Erfahrungsberichten der Benutzer geschah, standen zwei Audio-Engines zur Auswahl, die beide einen sehr guten Ruf bei den Anwendern hatten: OpenAL (Audio Library) und Fmod. Beide Engines sind bewährt und werden in vielen professionellen kommerziellen Produkten eingesetzt. Ebenso erfüllen beide das für uns sehr wichtige Kriterium der Plattformunabhängigkeit, da unsere Simulation ja auch unter Linux laufen sollte. Nach einigem weiteren Recherchieren einerseits und Beratungen mit erfahrenen Programmierern entschieden wir uns schließlich für Fmod. In einem direkten Vergleich der beiden API gefiel Fmod uns aufgrund der leicht verständlichen und intuitiven API besser als OpenAL. Ebenso stellt Fmod eine sehr große Bandbreite an Funktionalitäten zur Verfügung, die wir benötigten, wie etwa das Positionieren statischer Sounds in der dreidimensionalen Welt, oder das regelmäßige Aktualisieren dynamischer Soundquellen,

deren Bewegung in der Welt auf diese Weise simuliert wird. Auch war zu diesem Zeitpunkt bereits die Entscheidung gefallen, auf der Basis der OGRE-Engine zu arbeiten. Die Fmod-Audio-Engine war sehr gut für die Zusammenarbeit mit OGRE ausgelegt, und stellte entsprechende Schnittstellen bereit, damit „die Welt“ mit den „Sounds“ kommunizieren konnte, und beispielsweise im Falle einer Kollision (in der Welt) ein Geräusch (von der Sound-Engine) abgespielt werden konnte. Darüber hinaus gibt es eine durch Foren kommunizierende Community um Fmod, die auch vielfach mit Rat beiseite stand; auch die API war sehr gut und umfangreich erklärt; diese praktischen Gründe festigten unseren Beschluss, Fmod als Bibliothek zu verwenden.

15.2.2 Benutzung von FMod

Um mit der fmod-Engine einen 3D-Sound in der Welt zu platzieren, lädt man zunächst die entsprechende Wave-Datei mittels der Funktion

`FSOUND_SAMPLE_LOAD`.

In diesem Schritt hat man auch die Möglichkeit anzugeben, ob das Sample dreidimensional im Raum platziert werden soll oder ob es, wie im Falle der Hintergrundmusik beispielsweise, in Stereo aus den Lautsprechern ausgegeben werden soll.

In der Hauptschleife des Programms werden nun im Update unter anderem die Geräuschquellen bewegt; dies geschieht für fmod mit der Funktion

`FSOUND_3D_SetAttributes`.

Hier wird fmod mitgeteilt, an welcher Position sich das betreffende Objekt befinden soll; die Position, an die das Objekt bewegt wird, stammt aus der Engine.

Jetzt muss noch der Spieler selbst bewegt werden (sofern die Engine das vorsieht). Hier dient die Funktion

`FSOUND_3D_Listener_SetAttributes`.

Die Funktion nimmt als Parameter unter anderem die Position und Geschwindigkeit des Spielers sowie die Ausrichtung in der Welt.

Im Anschluss muss noch die Methode

`FSOUND_UPDATE`

aufgerufen werden, damit der Zustand der Audio-Engine intern aktualisiert wird.

Damit dieses System der dreidimensionalen Positionierung so funktioniert, musste die Soundgruppe eng mit der Enginegruppe zusammenarbeiten, da viele Aufrufe von Methoden, welche von der Soundgruppe implementiert wurden, von der Enginegruppe aufgerufen werden und umgekehrt.

Die Dreidimensionalität des Sounds wird über zwei Boxenpaare ausgegeben. Es handelt sich hierbei um zwei 2.1-Systeme, die jeweils mit einem Subwoofer ausgestattet sind; dies war eine verhältnismäßig kostengünstige Lösung im Vergleich zu einem 5.1-System (siehe nächsten Abschnitt).

15.3 Soundbibliothek

15.3.1 Auswahl und Bearbeitung von Soundsamples

Schon zu Beginn des ersten Semesters stand fest, dass der MiCarpet-Cave mit 3D-Sound ausgestattet werden soll. Aus diesem Grund wurde schon im Verlauf der weiteren Semester darauf geachtet, von verschiedensten Quellen Sounds zu sammeln. Zu beachten, welches Datenformat die jeweiligen Sound-/

Musikstücke haben, war anfangs nur nebensächlich. Im Allgemeinen lassen sich Datenformate nachträglich problemlos mit anderer Software in das gewünschte Format konvertieren. Fest stand nur, dass die Titelmelodie, die permanent im Hintergrund abgespielt wird, ein Stück im MP3-Format ist. Die einzelnen Hintergrundgeräusche der Umgebung und die verursachten Aktionen durch den Spieler sollten hingegen im WAV-Format vorliegen.

Alle Samples, die im Projekt MiCarpet verwendet wurden, stammen ausschließlich aus Internetquellen des World Wide Web. Dank des grossen Angebotes an verschiedensten Musiksamples konnte eine umfangreiche Soundbibliothek erstellt werden. So gab es genügend Quellen, die beispielweise Naturgeräusche oder Tierstimmen für den freien Download bereit stellten. Die Sounds, die nicht zu finden waren, wurden einfach aus diversen anders klingenden Geräuschen erstellt oder zusammengemixt. So wurde beispielsweise schlichtweg eine amerikanische Abfangrakete zu einem Kollisionssound einer Palme adaptiert.

Wie sich später herausstellte, bestand das einzige Problem darin, dass die ausgesuchten Samples eine minimale Frequenz besitzen mussten. Diese musste über dem Wert von 8 KHz liegen, da fmod (siehe [**?**]) ansonsten Probleme beim Abspielen aufweist. Mit der entsprechenden Software konnte zudem auch die Konvertierung erledigt werden. Als Konvertierungssoftware wurde das Programm *Goldwave* genutzt. Mit diesem Tool können Musikstücke in jeglicher Hinsicht bearbeitet und verändert werden. Die Veränderung von Höhen, Tiefen und Lautstärken sowie das Umkonvertieren von Formaten und das Zurechtschneiden der entsprechenden Musiksamples verlief hiermit reibungslos. So umfasste letztendlich unsere Soundbibliothek ein umfangreiches Sortiment von Wind-, Natur-, Tier- und Kollisionsgeräuschen, welche je nach Ereignis zu dem entsprechenden Event abgespielt wurden.

Im Folgenden werden Verweise zu diversen Internetadressen aufgelistet, aus denen Dateien für die Soundbibliothek extrahiert wurden:

1. “Platinum Pictures Multimedia” http://www.3dcafe.com/asp/soundsanimals1.asp 2004
2. “GRSites.com” http://www.grsites.com/sounds 2004
3. “Hooti” http://joelbramblett.tripod.com/sounds.html
4. “Partners In Rhyme” http://www.partnersinrhyme.com/pir/PIRsfx.html 2004
5. “Advances.com” http://www.audiosparx.com/sa/display/cat.cfm?soundamerica=true 2003
6. “Sounddogs.com, Inc.” http://www.sounddogs.com/subcategories.asp?Type=1&CategoryID=1060 2004
7. “Soundquellen im Internet”, D. Massow http://mitglied.lycos.de/dima1703/sound/soundquellen.htm 09/01/2001
8. “TheFreeSite.com” http://www.thefreesite.com/Free_Sounds/Free_WAVs/index.html 2004
9. “TNS Group, Inc.” http://www.freesoundfiles.tintagel.net/Audio/ 2004
10. “WavWeb” http://www.members.tripod.com/ buggerluggs/ie/wav-dir184.htm
11. “The Sound Page” http://members.tripod.com/ushk/sounds/sounds.html

15.3.2 Auswahl der Hintergrundmusik

Im folgenden möchten wir etwas genauer beschreiben, wie es zu der Auswahl der eingesetzten Musikstücke kam und welche Voraussetzungen und Kriterien hierbei eine entscheidende Rolle gespielt haben. Die Mitglieder der Soundgruppe hatten sich, nachdem schließlich klar war, in welchem Kontext sich die virtuelle Welt des Caves befinden würde, die Story also weitestgehend abgesteckt und formuliert war, überlegt, die Musik, die als Hintergrundmusik fungieren soll, selber zu komponieren. Um dies zu realisieren, wollten wir über ein an den Computer angeschlossenes MIDI- Keyboard mit entsprechender Software und erarbeitetem Hintergrundwissen über Tonalität und Rhythmus der orientalischen Musik kreativ zu Werke gehen. Wir recherchierten also über die Theorie der arabischen Musik, welche Instrumente dort primär zum Einsatz kommen, welche Tonabstände und Tonarten typisch sind, wie das Tempo gegeben ist und wie die Klangfarbe allgemein aussieht. Es wurde uns aber schnell bewusst, dass es leider aufgrund der vorgegebenen Zeit die wir zur Verfügung hatten, um den ersten Meilenstein zu erreichen, nicht realistisch war, unser Vorhaben zu realisieren. Die Zeit, welche die Programmierung der Soundengine erforderte, ließ uns leider nicht die Möglichkeit offen, uns der eigentlichen Komposition zu widmen. Da wir alle noch keine Erfahrung in der Soundprogrammierung besaßen, war eine intensive Einarbeitung in die Audio-Engine Fmod notwendig. Wir beschlossen also, auf die eigenständige Komposition zu verzichten und überlegten uns weiterhin, wie wir die Hintergrundmusik aus bereits bestehenden Musikstücken gestalten könnten. Wir durchforsteten unsere privaten CDs und das Internet nach arabischen Künstlern und entsprechenden MIDI Dateien, dachten an Mozarts Oper “Die Flucht aus dem Serail“ (Köchelverzeichnis 384) und hörten uns Musik aus uns bekannten und für unsere Zwecke vergleichbaren Computerspielen an. So zum Beispiel die Musik aus dem Spiel “Magic Carpet“, erschienen 1994 unter dem Label Bullfrog Productions, sowie ausgehend von unserer Piraten-Story, die karibischen Klänge aus den “Monkey Island“ Spielen von LucasArts. Auch der Kinofilm “Fluch der Karibik“ (Touchstone, 2003) bot in dieser Richtung einiges an Material. Trotz der massiven Recherche in diesem Bereich entschieden wir uns letztendlich allerdings dennoch weder für karibische noch arabische Klänge. Die für uns am meisten geeignete Musik, die als begleitende, unterstützende, aber dabei nicht aufdringliche Hintergrundkulisse diesen sollte, bot sich in sphärischen Soundstücken, das heisst in synthetischer Musik mit vielen Streichern und einem eingängigen Rhythmus. Dabei fanden wir zwei Musikstücke, die auch aufgrund ihrer Länge diesbezüglich sehr passend waren. Denn es stellte sich bei den meisten der vorher herausgesuchten Lieder heraus, dass sie zeitlich die Dreiminutenmarke nur selten überschritten, was für unsere Zwecke leider einfach zu kurz war. Es bestand natürlich die Möglichkeit, diese Lieder zu „loopen“, also immer wieder von vorne abzuspielen, wenn sie zu Ende waren, was bei der geringen Länge eines jeden Stückes aber doch etwas schnell ermüdend gewirkt hätte. Eine andere Möglichkeit wäre gewesen, mehrere Lieder für ein jeweiliges Level auszuwählen und zu arrangieren, diese Idee verwarfen wir aber nach einigen Tests wieder, da die Atmosphäre welche durch die Musik geschaffen wurde dadurch sehr schwankend und inhomogen war. Die Stücke die wir schließlich für die beiden Level ausgewählt haben, sind zum einen von der Gruppe “Vangelis“ das Lied “Islands of the orient“, erschienen 1996 auf dem Album “Oceanic“ und dem Label “Eastwest“ für das erste Level, sowie das Lied “Finished Symphony“ der Gruppe “Hybrid“, als Maxi CD erschienen 1999 auf dem Label “Distinctive Records“ für das zweite Level.

15.4 Probleme

Die Realisierung des Sounds im Cave war kein triviales Problem. Es traten trotz umfangreicher Einarbeitung (unter anderem in die Funktionen des fmod-Soundsystems) und theoretischer Vorbereitung an verschiedenen Stellen zwischendurch immer wieder neue Probleme auf, die aber gelöst werden konnten. Ein weiteres Problem bestand in der schwierigen Raum- und Rechnersituation, die für verschiedene Teilgruppen des Projekts gleichermaßen bestand. Um das von uns Programmierte testen zu können, war

es notwendig, an dem schnellsten der drei Projektrechner zu arbeiten. Es galt also, sich mit den Gruppen zu arrangieren, die diesen ebenfalls benötigten. Dies verursachte gelegentlich Probleme, ein bestimmtes Ziel zeitgerecht zu erreichen.

15.5 Finale Realisierung

Die finale Realisierung des Sounds orientierte sich am Drei-Stufen-Plan und setzt sich zusammen aus den beiden Teilen *Hardware* – dazu gehören die Nutzung von vorhandenen Ressourcen (Rechner, Soundkarte) sowie die Anschaffung der erforderlichen Geräte (Boxen, Kabel) und deren Positionierung im Cave – und *Software*, wozu das Programmieren inklusive des verwendeten Soundsystems fmod ebenso zählt wie die Recherche, Anpassung und Einbindung der Sounds.

Zum Teil Hardware wurde bereits im vorigen Abschnitt 15.1 das Meiste erwähnt. Nachdem eingangs von der Recherche der Sounds und dem Anlegen einer Soundbibliothek berichtet und oben bereits auf die grundlegenden Standards eingegangen wurde, soll die Programmierung konkreter beschreiben.

Der Drei-Stufen-Plan sah für den ersten Meilenstein eine einfache Hintergrundmusik und Fluggeräusche vor wie z.B. Wind und Flattern von Teppichfransen. Für den zweiten Meilenstein waren Soundreaktionen von Seiten der Objekte in der Welt zu realisieren; das heißt, dass diese einen spezifischen Sound von sich geben, wenn der Spieler sich ihnen nähert (zum Beispiel eine Wohngegend die häusliche Geräusche von sich gibt wie angeregte Unterhaltungen oder das Klappern von Geschirr). Zur Fertigstellung des dritten Meilensteins wurde entsprechend des Leveldesigns eine Soundbibliothek erstellt sowie ein Eventsystem realisiert, sodass bei Kollisionen des Spielers mit Objekten und bei dem Erreichen einer bestimmten Aufgabe ein dafür passender Sound abgespielt wird.

Die Aufgabe lag unter anderem darin, den Objekten, die in der Spielwelt auftauchen, einen geeigneten Sound zuzuordnen. Diese Objekte und Objekttypen in der Welt sind in den jeweiligen Leveldateien definiert. Die speziellen Objekttypen, die einen Sound besitzen sollen, bekommen in der selben Datei einen speziellen Sound-Marker, welcher den Namen der abzuspielenden Datei ohne explizite Angabe der Dateiendung und Verzeichnisangabe enthält. Ganz ähnlich verhält es sich mit der Hintergrundmusik, nur dass diese keinem speziellen Objekttypen zugeordnet, sondern der Hintergrundmusik-Marker seinerseits wieder im Ambient-Marker der Leveldatei zu finden ist (siehe auch Kapitel 12.1.3 (Seite 87)). Dazu gilt zu erwähnen, dass die jeweilige Hintergrundmusik eines Levels in einer Endlosschleife abgespielt und bei Beendigung des Levels gestoppt wird. Die dynamischen sowie statischen Objekte werden beim Laden der Leveldaten auf den Sound-Marker hin traversiert und zutreffende Objekte in eine Liste geschrieben. Für jedes Element dieser Liste wird schließlich ein Soundobjekt erzeugt, welches die entsprechende Zeichenkette für die Sounddatei als Parameter erhält.

Erzeugt werden diese Soundobjekte im Soundmanager, wo auch bei der Initialisierung der Welt das Initialisieren und Laden der fmod-Engine stattfindet. Bei der Erzeugung der Soundobjekte wird der Name der Sounddatei auf seine Dateiendung und seinen Pfad hin erweitert, sowie ein entsprechend freier Kanal bereitgestellt. Die Soundobjekte selber sind vom Typ `cSound`, was bedeutet, dass sie die Eigenschaften besitzen, die in der Klasse `Sound` definiert sind. Dazu zählen zum Beispiel die Lautstärke des abgespielten Sounds, die Loopeigenschaft und der Distanzfaktor. Letzterer ist zuständig für die Distanz (ausgehend vom jeweiligen Objekt), ab welcher der Sound zu hören ist. Entsprechend verringert oder erhöht sich die Lautstärke, wenn man sich vom Objekt fortbewegt oder nähert. Da durch die Soundkarte eine bestimmte Anzahl von Hardwarekanälen festgelegt war, musste ein spezielles Kanalhandling erarbeitet werden. Gelöst wurde dies, indem bei jedem Update in der World-Klasse abfragt wurde, ob auf den vergebenen Kanälen noch ein Sound abgespielt wird, um diesen gegebenenfalls wieder freizugeben. Neben den Objekten, die einen festen Platz und spezifischen Sound in der Spielwelt haben, war es weiterhin nötig, Geräusche abzuspielen, wenn ein bestimmtes Ereignis (Event) eintrat. Dies konnte durch

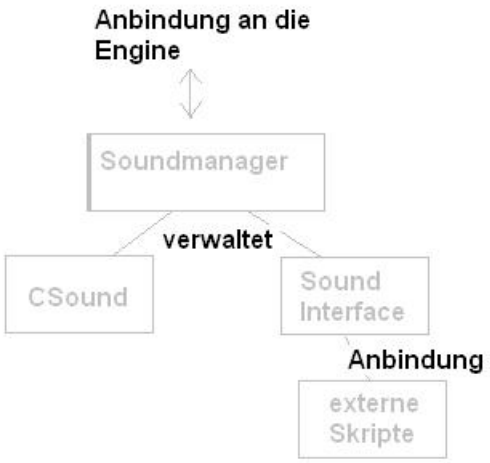


Abbildung 15.2: Schema Finale Realisierung Sound

eine Kollision des Spielers mit einem Objekt sowie durch das Erfüllen einer bestimmten Aufgabe gegeben sein. Dazu wurde ein Soundinterface zur Verfügung gestellt, welches Funktionen enthält, die diese Events behandelt. Die Funktionen werden dabei über die Scripte aufgerufen.

Teil VI

Schlusswort

Abschließend wollen wir uns noch einmal bei allen Personen bedanken, welche durch ihre Hilfe und Mitwirkung das Projekt erst möglich gemacht haben. Als erstes seien hier Professor Friedrich-Wilhelm Bruns als Initiator und Betreuer aller drei MiCa-Projekte sowie Dr. Dieter Müller als direkter Betreuer dieses Projektes genannt, welche uns über die zwei Jahre stets unterstützt und geholfen haben. Unser Dank gilt außerdem allen anderen Mitgliedern des artec-Instituts, die oft bei Sachfragen mit Rat und Tat zur Seite standen. Schließlich waren die Sponsoren des Projektes eine große Hilfe, da nur durch ihre Unterstützung etliche Teile des Systems so gebaut werden konnten, wie es vorgesehen war.

Zum Schluß soll angemerkt werden, dass zwar das Projekt beendet ist, aber damit nicht die Arbeit an dem MiCarpet-System: im Rahmen von Diplomarbeiten werden noch einzelne Aspekte des Systems weiterentwickelt; dies gilt vor allem für die in Kapitel 14 vorgestellte Motion-Platform. Hier ist das Projekt auch die Grundlage für eine weitergehende Beschäftigung und Entwicklung im Bereich Mixed Reality.

Teil VII

Anhang

Anhang A

Projekthandbuch

Folgender Anhang ist eine Kopie der Projekthandbuches, welches für das Projekt im März 2003 aufgestellt wurde. Das Projekthandbuch war ein Leitfaden für die organisatorischen Details und Informationen im Projekt (für die Teilnehmer selbst).

A.1 Organisation

Die Organisation des Projekts über die Laufzeit von zwei Jahren teilt sich in die Bereiche des Organisationsteams, des Kassenwarts und des Webmasters auf.

A.1.1 Aufgaben der einzelnen Gruppen

Der Inhalt des vorliegenden Handbuches ist für die Teilnehmer, welche für einen entsprechenden Zeitraum für einen Aufgabenbereich eingesetzt sind, verbindlich. In dem jeweiligen Abschnitt sind notwendige Regelungen festgeschrieben und sollen möglichst eingehalten werden.

Organisationsteam

Das Organisationsteam - im Weiteren als Orga abgekürzt - wird für eine Zeit von exakt der Dauer eines Semesters eingesetzt. Dies schließt die vorlesungsfreie Zeit im Anschluss an den normalen Semesterbetrieb ein. Die Orga besteht aus vier Teilnehmenden des Projektes. Die Wahl der Orga wird zum Ende der vorlesungsfreien Zeit, spätestens zu Beginn eines neuen Semesters, ausgeführt. Eine Wahl kommt zustande, sofern mehr als vier Teilnehmer freiwillig diese Aufgabe übernehmen wollen und im Vorfeld keine Einigung getroffen werden kann. Von dieser Regelung befreit ist lediglich der Kassenwart.

Zu den Aufgaben der Orga gehört in erster Linie die Organisation des wöchentlichen Projekttages. Dieser ist zunächst in Projekt-Plenum, Vorlesung, Arbeitsvorhaben und in eine gruppenübergreifende Kollektiv-Veranstaltung aufgeteilt. Zu der primären Aufgabe der Orga zählen ferner die Moderation sowie Protokollierung des Plenums einschließlich anschließender Aufbereitung und Bereitstellung der Mitschriften. Die Gestaltung der einzelnen Veranstaltungen (Projekt-Plenum, Arbeitsvorhaben, Projektwochenende und außerordentliche Treffen) ist ebenfalls Aufgabe der Orga. Hierzu ist ein Organisationstool zu verwenden, um den Ablauf des Projektes zu planen und zu dokumentieren. Während des Projekttages und bei außerordentlichen projektspezifischen Terminen obliegt der Orga die organisatorische Koordination von den zu erledigenden Aufgaben und deren Durchführung.
Außerdem ist die Orga verantwortlich für das Voranschreiten des Projektes. Dieses ist regulierbar durch die Planung der Arbeitsvorhaben, die Aufbereitung und das Informieren über erzielte Ergebnisse, sprich

Informationsquelle zu sein. Über die Aufgaben des wöchentlichen Turnus' hinaus ist die Orga Ansprechpartner für alle projektbezogenen Fragen und die Organisation von Projekttreffen innerhalb der vorlesungsfreien Zeit. Zum Schluss der eingesetzten Zeit muss die Orga einen Semesterbericht verfassen.

Kassenwart

Der Kassenwart wird zu Beginn des Projektes für die Gesamtdauer gewählt. Diese Aufgabe obliegt B. Vaudlet, der damit von anderen Aufgaben der Organisation des Projektes entlastet ist. Primäre Aufgabe des Kassenwarts ist die Buchhaltung. Das Eröffnen eines Projekt-Kontos gehört zu Beginn der Projektlaufzeit zu den Aufgaben. Die Buchhaltung schließt das Überwachen der Ein- und Ausgänge des Projekt-Kontos ein. Hierzu zählt auch das Auffordern zu den monatlichen Einzahlungen einzelner Teilnehmer auf das Projekt-Konto sowie das Einfordern der universitären Vergütung für das Projekt pro Person und Semester. Auslagen für projektbezogene Anschaffungen müssen mit einer Quittung belegt werden. Das Sammeln der entstandenen Quittungen sowie Auszahlungen ausgelegter Gelder für Ressourcen liegt bei dem Kassenwart. Bei Rückzahlungen monatlich geleisteter Überweisungen zu Projektende muss darauf geachtet werden, welche Eingänge von dem Teilnehmenden zu verbuchen sind. Rückzahlungen orientieren sich demnach an den geleisteten Anteilen.
Die Anschaffung von Ressourcen für das Projekt erfordert eine monatliche Einzahlung von fünf Euro pro Teilnehmer auf das Projekt-Konto. Die Daten für das Projekt-Konto sind wie folgt:

Kontonummer	1160199160
Bankleitzahl	29151700
Institut	KSK Syke
Empfänger	Mica
Verwendungszweck	Beitrag [Monatsname]
Summe	5,00 EUR

Für den Verwendungszweck ist [Monatsname] durch den jeweiligen Monat, für den der Beitrag gilt, zu ersetzen. Beispielsweise sieht eine Überweisung für den Monat Dezember an der Position Verwendungszweck wie folgt aus: Beitrag Dezember. Zu Beginn des Projektes (November 2002) ist eine einmalige Überweisung von zehn Euro vorgesehen. Auslagen für Ressourcen für das Projekt sind erst nach Rücksprache mit der Orga zu tätigen und können nur durch Vorlage einer Quittung zurückerstattet werden. Der monatliche Beitrag kann auch durch eine entsprechende Summe, Monate mal Faktor fünf, im Voraus getätigt werden. Dies ist dann auf der Überweisung entsprechend zu vermerken.
Eine pünktliche Einzahlung ist wünschenswert, sicherlich jedoch nicht verpflichtend. Sollten finanzielle Engpässe auftreten, so kann eine Überweisung ausfallen. Dies ist dem Kassenwart allerdings via Email (bvaudlet@tzi.de) zu melden.

Webmaster

Aufgaben des Webmasters sind die Pflege der dem Projekt zugeordneten Internet-Plattform, die Aktualisierung des dazugehörigen Forums und die Vergabe von persönlichen Zugängen (Accounts) zu diesen Medien. Hierzu sind die Teilnehmenden B. Breder, T. S.-H. Ahlbrecht, P. Rodacker und C. Lauterbach eingesetzt. Darüber hinaus obliegt dem Webmaster das Layout und Design der Webpräsenz zu dem Projekt.

A.1.2 Zugangs- / Schlüsselordnung

Der Schlüssel für den Projektraum befindet sich in einem Schlüsselkasten, welcher im MZH, Ebene 0 installiert ist. Schlüssel für den Schlüsselkasten sind in zehnfacher Kopie im Umlauf unter den Teilnehmenden. Zugangskarten für das GW2-Gebäude, in welchem sich der Projektraum befindet, sind ebenfalls in zehnfacher Ausgabe im Umlauf.

A.1.3 Projektraum

Der Projektraum befindet sich im GW2-Gebäude der Universität Bremen. Die Raumnummer ist B1810 B.

Nutzung, Verhaltensregeln

Die Nutzung des Projektraumes ist primär den Teilnehmenden des Projektes vorbehalten. Für projektexterne Personen ist die Nutzung frei, sofern eine teilnehmende Person des Projektes anwesend ist und der Raum nicht für Projektinterna benötigt wird. Der Projektraum ist nicht gedacht als Abstellplatz für persönliche Sachgegenstände. Ebenso ist die Entwendung von Ressourcen des Projektes aus dem Projektraum, welche aus Projektmitteln finanziert worden, untersagt. Für die Nutzung des Internetzugangs gelten die Richtlinien der Ebene 0. Auch ist darauf zu achten, entstandene Abfälle beim Verlassen des Raumes zu entsorgen.

A.1.4 Expertengruppen

Die Expertengruppen ergeben sich durch die Aufteilung der Aufgaben, die zur Realisierung des Caves anstehen, auf verschiedene Gebiete. Die Realisierung fordert die Unterteilung in die Gruppen: Cave, Sensorik/Aktorik/Interface, Plattform, Sound, Coding Guidelines und dreier temporärer Gruppenkonstellationen. Die Aufgaben der einzelnen Gruppen ergeben sich intuitiv aus deren Bezeichnung und werden innerhalb der einzelnen Gruppen festgelegt. Dabei ist auf eine Absprache mit den anderen Gruppen zu achten. Die Teilnehmenden sind wie folgt auf die Gruppen aufgeteilt:

Hardware (permanent): Christian Lauterbach,
Manfred Biess (Ansprechpartner)

Cave (Konstuktion) (permanent): Gunnar Niehuis,
Tina Ahlbrecht,
Patrick Breder (Ansprechpartner),
Manfred Meiss

Grafikimplementierung (permanent): Christian Lauterbach (Ansprechpartner),
Christopher Koschack,
Manfred Biess,
Patrick Rodacker

Grafikmodellierung (permanent): Tobias Scheele (Ansprechpartner),
Björn Breder

Soundkompositionen -installation (permanent): Gunnar Niehus,
Tina Ahlbrecht,
Manfred Meiss,

Benjamin Vaudlet (Ansprechpartner),
Patrick Breder

Interfacetechnik (Sensorik/Aktorik) (permanent): Markus Emde,
Elmar Berger (Ansprechpartner),
Franck Ngueuleu,
Volker Weinert,
Tobias Scheele,
Tatiana Kotas,
Alexander Janot,
Björn Breder (Ansprechpartner)

Anforderungsdefinition (temporär): Christian Lauterbach,
Christopher Koschack,
Patrick Rodacker

Softwarespezifikationen (temporär): Christian Lauterbach,
Christopher Koschack,
Patrick Rodacker

Sponsoring-Mappe / Public Relations (temporär): Tobias Scheele,
Patrick Rodacker,
Björn Breder

Webmaster (permanent): Christian Lauterbach,
Patrick Rodacker (Ansprechpartner),
Björn Breder,
Tina Ahlbrecht,
Manus Meiss

Finanzen: Benjamin Vaudlet (Ansprechpartner)

Arbeitspakete

Die Bezeichnungen für die einzelnen Arbeitspakete ergeben sich innerhalb der einzelnen Expertengruppen.

A.2 Kommunikation

Die Kommunikation innerhalb des Projektes wird neben dem Projekttag vorwiegend über das Medium Internet abgewickelt. Hierzu ist das regelmäßige Abrufen von Emails und der regelmäßigen Besuch des Forums während des Semesterbetriebs verbindlich. Innerhalb der vorlesungsfreien Zeit kann der Besuch des Forums und der Abruf von Emails reduziert werden, sollte aber bei einem bevorstehenden Treffen noch unmittelbar davor genutzt werden. Ferner liegt jedem/jeder Teilnehmenden eine Kontaktliste mit Telefonnummern jedes Mitglieds vor.

A.2.1 Medium Internet

Die Internetpräsenz des Projektes hat die Funktion, Informationen unter den Teilnehmenden zu verbreiten. Zu erreichen ist die Webpräsenz unter `http://www.micarpet.de`. Jede/r Teilnehmende hat

einen persönlichen Zugang zu der Internetpräsenz (Account) bestehend aus Benutzername und Passwort. Außerdem besteht ein weiterer öffentlicher Zugang zu den Seiten, der eine eingeschränkte Funktionalität bereithält. Dieser Zugang ist vorwiegend für potenzielle Sponsoren des Projektes und interessierte Personen gedacht. Der Inhalt der Websites ist daher seriös zu sehen und entsprechend zu pflegen. Bei der Einspeisung von Informationen ist daher darauf zu achten, dass diese keine Klamauknamen oder -bezeichnungen enthalten. Die sprachliche Form der Inhalte sollte möglichst klar und unmissverständlich sein. Das Medium Internet stellt für die Kommunikation innerhalb des Projektes zwei Plattformen zur Verfügung. Das Forum, zu erreichen unter `http://www.micarpet.de/forum/`, ist primäre Informationsquelle. Der Emailverkehr ist nur als Ausweichmöglichkeit zu erachten.

Forums-Richtlinien

Als primäres Kommunikationsmedium dient das Forum der Internetpräsenz. Der Zugang zu dem Forum ist ebenfalls passwortgeschützt. Die Zugangsdaten sind für jede/n Teilnehmende/n die selben wie die der Internetpräsenz. Diese Zugangsdaten werden zunächst von dem Webmaster ausgegeben, können aber noch modifiziert werden. Hierbei ist darauf zu achten, den Internetauftritt sowie das Forum durch eine vernünftige Kombination des Passwortes aus Buchstabe, Zahlen und Sonderzeichen zu schützen. Beispielsweise sollte das Wort mica o.ä. als Passwort vermieden werden.
Im Rahmen des Projektes werden verpflichtende Umfragen im Forum bereitgestellt. Diese sind entsprechend gekennzeichnet und am Thread zu erkennen. Die Teilnahme an solchen Umfragen ist verpflichtend. Um die Übersichtlichkeit zu behalten und das Einsetzen eines Moderators zu verhindern, sollten in dem Forum keine unnötigen Postings gemacht werden. Wenn dem Autor eines Postings im Nachhinein Tippfehler auffallen, so kann eine Nachricht editiert werden. Von der Praxis, mittels einer zweiten Nachricht den Fehler der ersten Nachricht zu revidieren, sollte Abstand genommen werden.
Wichtige Postings, die besonders zur Kenntnis genommen werden sollen, werden im Thread (im Topic) entsprechend gekennzeichnet. Diese Option bleibt der Orga vorbehalten. Wichtige Themen für das Forum können an die Orga gereicht werden. Die Subjects (Topics, Threads) müssen eindeutige Bezeichnungen enthalten. In speziellen Fällen enthalten diese zudem noch Schlüsselwörter, wie poll = Umfrage oder urgent = wichtig.

Email-Richtlinien

Der Emailverkehr ist nur als Ausweichmöglichkeit zu sehen, wenn es darum geht, alle Teilnehmenden des Projektes zu erreichen. In solchen Fällen ist auf folgende Richtlinien zu achten:
Der Betreff einer Email sollte eindeutig sein, zudem sollte dieser das Schlüsselwort mica enthalten. Während des normalen Semesterbetriebs ist von einem nahezu täglichen Abrufen der Emails aller Teilnehmenden auszugehen. Der Kommunikationsweg via Email ist vor allem bei kurzfristigen Benachrichtigungen zu benutzen. Beispielsweise sollten kurzfristige Abmeldungen von Projekttreffen wegen Krankheit über Email mitgeteilt werden. Der Emailverkehr findet ohne Verschlüsselung statt.

A.2.2 Weitere Kommunikationswege

Neben dem Internet als vorwiegendem Kommunikationsweg sind andere Möglichkeiten zur Kommunikation unter den Teilnehmenden willkommen; diese sind der in 2. erwähnten Kontaktliste zu entnehmen.

Plenum

Während des Projekt-Plenums obliegt die Moderation der Orga. Diese ist auch Wortführer und verteilt das Rederecht. Während der Diskussion ist auf die Reihenfolge der Wortmeldungen zu achten. Gesprä-

che, die nicht das inhaltliche Thema betreffen, sollen möglichst außerhalb des Plenums gehalten werden. Das inhaltliche Verfolgen des Plenums wird vorausgesetzt.

Arbeitsvorhaben

Während des Arbeitsvorhabens agiert jeder Teilnehmer autark oder in der Expertengruppe, sofern in dem Projekt-Plenum kein spezielles Vorhaben für das Arbeitsvorhaben vereinbart wurde. In der Regel findet im Arbeitsvorhaben keine Moderation statt.

A.3 Dokumentation

Die ständige Dokumentation des Projektfortschritts ist Basis für alle anstehenden Berichte. Die Dokumentation umfasst die Handhabung von Dateien, die für das Projekt erstellt werden und die Mitschriften aus dem Projekt-Plenum, sowie aus den außerordentlichen Treffen.

A.3.1 Dateibenennung

Die Benennung von Dateien sollte einheitlich in folgendem Format erfolgen: <Arbeitspaket>_<Titel>_<Akronym des Autors>_<Datum>.<Dateiformat>. Als Arbeitspaket versteht sich die Bezeichnung des Arbeitspaketes einer Expertengruppe. Die Arbeitspakete werden durchnummeriert und haben demnach die Form AP1, AP2, ... Als Titel ist eine eindeutige Bezeichnung des Dokumentinhalts zu wählen. Akronym des Autors ist dreistellig und setzt sich aus dem Anfangsbuchstaben des Vornamens und den beiden ersten Buchstaben des Nachnamens zusammen (Bsp.: Clint Eastwood $\Rightarrow$ cea). Das Datum ist in der Form YYMMDD mit aufzunehmen (Bsp.: Der 4. März 2003 $\Rightarrow$ 030304). Ein abschließendes Beispiel für eine zulässige Dateibenennung sieht also wie folgt aus: AP2_Konstruktionsdetails_pro_030304.tex

A.3.2 Dateiformate

Um Mitschriften und erstellte Dateien allen Teilnehmenden zugänglich zu machen, sollte ein plattformunabhängiges Format für die Dateien gewählt werden. Für textbasierende Dokumente ist das pdf-Format zu wählen und entsprechend für grafische Ausarbeitungen das jpeg-Format. Sollte der Wunsch bestehen, können Originaldateien bei dem Autor angefordert werden, um Modifikationen vorzunehmen.

Plenums- und Arbeitsvorhabenprotokoll

Zu protokollieren sind stets folgende Sachverhalte: Ortsangabe, Zeitangabe, Teilnehmende, Moderator, Protokollant. Innerhalb eines Protokolls sollen Vornamen stets komplett ausgeschrieben werden und durch den Anfangsbuchstaben des Nachnamens ergänzt werden. Wahlweise kann der Nachname ausgeschrieben werden. Die Gliederung des Protokolls richtet sich an einer Auflistung der Tagesordnungspunkte (TOP's). Inhaltlich sollen nur solche Themen aufgenommen werden, die auch tatsächlich im Plenum oder Arbeitsvorhaben besprochen werden. Bei einer Diskussion sollte nur der Impuls, der zur Diskussion führt, und die Resultate aufgenommen werden. Insgesamt ist auf eine formale Ausdrucksform zu achten. Das Protokoll wird direkt über den Internetauftritt des Projektes in die Datenbank eingespeist. Gleichzeitig wird ein Textdokument generiert, welches in dem Download-Bereich der Website abgelegt wird.

Zwischen- und Semesterbericht

Eine Vorlage für das Layout eines Berichtes liegt im TEX-Format im Download-Bereich der Website vor. Diese Vorlage ist zu nutzen. Ein Zwischenbericht soll monatlich verfasst werden. Grundlage hierfür sind die Protokolle der Projekttage. AutorIn des Zwischenberichts variiert und wird zu einem anstehenden Termin gewählt. Über die Protokolle hinaus soll der Zwischenbericht weitere aktuelle Informationen zum Stand des Projektes enthalten und insgesamt elaboriert ausformuliert sein. Der Semesterbericht wird mit Ende der Orga-Amtszeit von eben jener verfasst. Der Semesterbericht setzt sich aus den Zwischenberichten zusammen, die innerhalb des Semesters verfasst wurden. In den Semesterbericht sollten nach Möglichkeit auch Grafiken mit einfließen. Ebenso wie der Zwischenbericht sollte der Semesterbericht in gleicher Weise ausformuliert sein und neben aktuellen Informationen den Fortschritt des Projektes darstellen.

Termine

Das Projekt betreffende Termine werden auf der Internet-Plattform veröffentlicht. Hierzu lässt sich ein Kalender öffnen, der die aktuelle und zwei Folgewochen anzeigt. Termine werden dort von der Orga eindeutig eingetragen.

Projekt- (Abschluss-) Bericht

Der Abschlussbericht greift den Inhalt der Semesterberichte auf und rundet die Informationen ab. Insgesamt ist auf eine ebenso elaborierte Form zu achten. Der Abschlussbericht wird von allen Teilnehmenden im gleichen Maße verfasst und gegenseitig Korrektur gelesen. Hier soll der gesamte Verlauf des Projektes aufgegriffen und ggf. bebildert dargestellt werden.

Anforderungsdefinition

In der Anforderungsdefinition werden minimale Ziele der baulichen Konstruktion, sowie der Hard- und Softwareimplementierung aufgenommen. Hierzu ist eine Expertengruppe eingesetzt, bestehend aus P. Rodacker, C. Lauterbach und C. Koschack.

Spezifikation (Software)

Die Softwarespezifikation legt die Struktur der zu entwickelnden Software fest. Zusätzlich werden Coding Guidelines als bindende Regelung veröffentlicht und im Projektraum ausgehängt.

Konstruktionszeichnungen /-berichte

Zu der Realisierung des Gerüstes sollten Entwurfs und Konstruktionszeichnungen entstehen. Diese sollten in die Berichte eingespeist und allen Teilnehmenden entsprechend den Regeln für Dateiformate und -benennung zur Verfügung gestellt werden. Hierzu sollte die Internetpräsenz genutzt werden.
Insgesamt sind alle entstehenden Entwurfszeichnungen zu sammeln und solche, die durch Handskizzen entstanden sind, sind zu digitalisieren.

Coding Guidelines

Die Coding Guidelines liegen jedem/jeder Teilnehmenden vor und sind als verbindlich zu erachten.

Anhang B

Anforderungsdefinition

Folgender Anhang ist eine Kopie der Anforderungsdefinition, welche für das Projekt im April 2003 aufgestellt wurde. Die Anforderungsdefinition ist auch getrennt auf der Projekt-Seite abrufbar. Diese ist zu finden unter (`http://www.micarpet.de`).

B.1 Übersicht

B.1.1 Einleitung

Das Projekt MiCa-C hat zum Ziel, ein Mixed-Reality-System im Rahmen eines Hauptstudiumsprojektes der Universität Bremen zu entwickeln. Die Anforderungen an dieses System werden in diesem Dokument detailliert beschrieben, so dass eine Referenz vorhanden ist, welche Ziele das Projekt genau hat. In einem späteren Dokument (Systemspezifikation) wird dann das System genauer spezifiziert, insbesondere die Klassenstruktur der Komponenten.

Im ersten Teil dieses Dokumentes soll eine Übersicht über die Komponenten sowie die Anwendung gegeben werden, im zweiten werden dann die genauen funktionalen Anforderungen an die Teile des Systems beschrieben. Weitere, nicht-funktionale Anforderungen werden im dritten Teil aufgeführt und die organisatorische Planung schließlich im vierten.

B.1.2 Zielsetzung

Das Ziel des Projektes ist es, ein komplett funktionsfähiges Mixed-Reality-System (Hardware und Software) zu entwickeln. Das System wird als projektorbasierte, CAVE-ähnliche VR-Umgebung laufen, in welcher durch Mixed-Reality-Elemente der Benutzer mit dem System interagieren kann. Die Hardwarekomponenten sind die CAVE [**?**] selbst (B.2.2, S.224), die Soundausgabe (B.2.5, S.225) sowie die Interaktionsgeräte wie die Motion-Platform (B.2.3/B.2.4, S.224/224). Als Anwendung wird auf der Hardwareplattform ein Spiel laufen, in welchem der Benutzer auf einem „fliegenden Teppich“ durch Landschaften fliegt (siehe den folgenden Abschnitt). Da mehrere Projektionsflächen angesteuert werden, wird die Darstellung der Grafik auf mehrere Rechner verteilt, die sich über das Netzwerk synchronisieren.

B.1.3 Spielidee

Im fertigen System wird die Anwendung ein Spiel sein, in dem der Spieler auf der Motion-Platform wie auf einem „fliegenden Teppich“ durch eine Landschaft fliegen muss. Aufgaben für den Spieler sind dann etwa das Durchfliegen von Ringen (wozu Geschicklichkeit benötigt wird) oder aber die Navigation durch

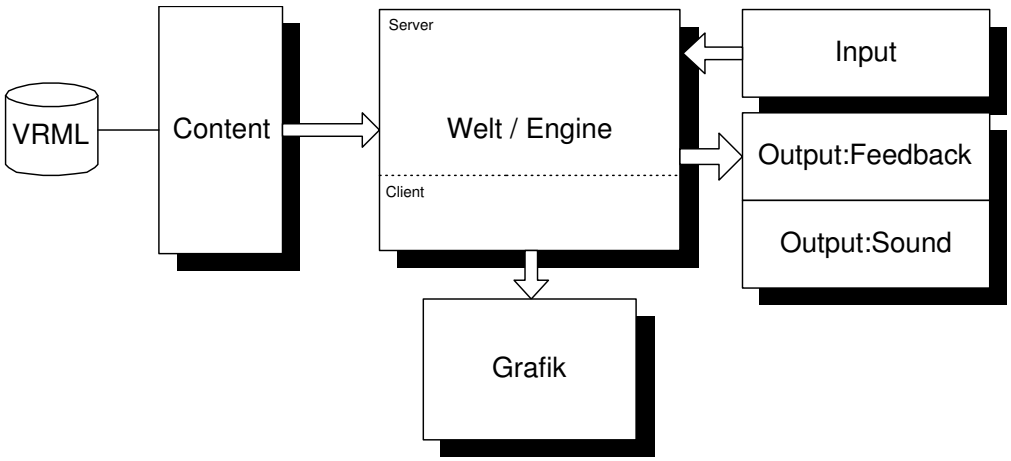


Abbildung B.1: Komponenten des Gesamtsystems

eine Canyon-Strecke. Dabei hat der Spieler aber völlige Freiheit und ist nicht auf eine bestimmte Route beschränkt, sondern kann sich in der Welt bewegen. Das Spiel erlaubt das Laden verschiedener Gebiete, die auch unterschiedliche Aussehen und Herausforderungen besitzen können.

Falls genug Zeit für die Implementation ist, wird der Spieler auch durch Aktionen seine Umwelt verändern können (z.B. durch „Zaubersprüche“), etwa einen Vulkan an einem Punkt schaffen. Denkbar wären auch computergesteuerte Gegner für den Spieler.

B.1.4 Bestandteile

Das System lässt sich konzeptuell etwa in folgende Komponenten aufteilen, welche in den funktionalen Anforderungen (siehe Teil B.2) einzeln beschrieben werden:

- Engine
- Grafik
- CAVE (Hardware)
- Sensorik
- Aktorik
- Sound

Der Zusammenhang zwischen diesen Komponenten lässt sich auch in Abbildung B.1 ersehen.

B.2 Funktionale Anforderungen

B.2.1 Grafikengine

Die Grafik ist einer der wichtigsten Punkte in der Implementation des Systems, da durch sie ein großer Teil des Realismus’ entsteht. Unter Grafik wird hier auch das Laden und die Verwaltung von verschiedenen Gebieten oder Umgebungen gefasst.

Darstellung

Die Grafikimplementation muss folgende Fähigkeiten haben:

- Darstellung der kompletten Szene inklusive Texturierung, Beleuchtung und Animation
- Beliebige Bewegung des Betrachters in der Szene
- Kollisionserkennung mit Objekten in der Szene und das Erzeugen von Feedback auf die Kollision

Die folgenden Fähigkeiten sind optional und werden bei ausreichender Zeit implementiert:

- Darstellung von Wettereffekten wie Blendung durch die Sonne, Wassereffekte, Tag- und Nachtzeit uvm.
- Darstellung von Partikelsystemen für Spezialeffekte
- Unterstützung von Shadern für die Texturierung

Grafikformat

Um zur Erstellung der Modelldaten für die Grafikengine die Verwendung von verfügbaren Standardprogrammen zu erlauben, muss die Implementation ein offenes Dateiformat unterstützen. Als Format wurde das VRML-Format ausgewählt, weil es ein offener Standard ist und von praktisch allen Programmen unterstützt wird. Dadurch kann die Erstellung der Grafikdaten zur Darstellung über Standardprogramme (etwa 3D Studio MAX, Maya, Blender,...) geschehen, so dass im Rahmen des Projektes keine eigenen Programme zur Modellierung entwickelt werden müssen.

Zur Verarbeitung der Modelldaten muss die Implementation folgende Punkte unterstützen:

- Laden von Terrains sowie zusätzlichen Modellen aus VRML-Dateien
- Laden von Animationen aus VRML-Dateien
- Laden von Texturen aus Bitmapdateien (etwa TGA, BMP)
- Aufbau eines Szenengraphen für die Modelldaten zur effizienten Darstellung

Die Implementation kann außerdem noch das zusätzliche Laden eines selbstdefinierten Dateiformates erlauben, in welchem weitere Informationen zu dem gewünschten Gebiet gespeichert sind. Dieses könnte etwa Informationen zu den Zielen oder Ereignissen im Gebiet enthalten.

Netzwerkfähigkeit

Da die Grafik auf mindestens drei Projektionsflächen ausgegeben werden muss (siehe Abschnitt B.2.2), kann die Ausgabe nicht von einem Rechner alleine bewältigt werden. Für die Ausgabe müssen also mehrere Rechner zusammen arbeiten. Die Grafikengine muss auch dies erlauben:

- Koordinierte Darstellung der Szene auf mehreren Rechnern mit unterschiedlichen Blickwinkeln
- Steuerung der Darstellung über ein lokales Netzwerk (LAN) auf Basis von Fast Ethernet-Technik und Standardprotokollen (TCP/IP, UDP)

- Teilung der Rechner in einen Master-Rechner (verarbeitet Eingabe und steuert die Darstellung) sowie mehrere Clients, welche die Szene aus einem bestimmten Blickwinkel darstellen
- Synchronisation der Änderungen an der Szene (Position des Betrachters, Animation von Modellen, Veränderungen am Terrain) zwischen Master und Clients
- Fehlertoleranz bei Ausfall eines Clients

B.2.2 CAVE

Die CAVE (die Benennung geht auf [**?**] zurück) ist hier das Konstrukt, auf das die erzeugte Grafik des Systems mittels Projektoren ausgegeben wird. Die CAVE muss drei Projektionsflächen bieten (Front sowie linke und rechte Seiten), optional kann sie außerdem eine Bodenfläche haben, auf die eine weitere Projektion möglich ist. Falls es notwendig ist, gehören zur CAVE auch noch Spiegelinstallationen, um das Projektorbild auf die entsprechende Bildfläche zu projizieren. Eine weitere Anforderung an die CAVE ist, dass sie einfach auseinander- und zusammengebaut werden kann, da sie transportabel sein muss. Weiterhin müssen Möglichkeiten vorgesehen sein, Sensorik- und Aktorik-Komponenten (siehe unten) zu montieren; dies muss insbesondere bei der Statik berücksichtigt werden. Die CAVE muss ausreichend groß sein, um die Motion-Platform aufnehmen zu können und noch einen Sicherheitsabstand von der Plattform zu den Seitenwänden haben, sowie ausreichend hoch sein, dass der Unterbau der Motion-Platform nicht zu weit oberhalb der Bodenprojektion ist.

B.2.3 Sensorik

Unter Sensorik wird hier alles gefasst, was im System Eingaben jeglicher Form vom Benutzer annimmt. Der wichtigste Sensor ist die Motion-Platform. Diese wird in der CAVE aufgestellt, so dass der Benutzer darauf sitzen kann. Durch angebrachte Sensoren an der Plattform muss dann festgestellt werden können, wie die Neigung der Plattform zum aktuellen Zeitpunkt ist und dies vom Computer abgefragt werden können. Insbesondere muss die Bestimmung der Neigung von zwei Achsen möglich sein, um zwei Freiheitsgrade zu haben.

Über eine allgemeine Schnittstelle zum System muss es auch möglich sein, gewöhnliche Eingabegeräte wie Maus und Tastatur zur Steuerung des Systems (oder zur Ergänzung) zu verwenden.

Zusätzlich kann noch ein weiteres Eingabegerät integriert werden: Es könnte etwa ein Datenhandschuh verwendet werden, um eine weitere intuitive Eingabemöglichkeit zu bieten und so fehlende Freiheitsgrade der Motion-Platform zu ergänzen.

B.2.4 Aktorik

Unter Aktorik sind hier alle weiteren Ausgaben bzw. physischen Reaktionen des Systems gefasst, welche nicht zu Grafik oder Sound gehören. Der wichtigste Punkt ist hier das Force-Feedback, welches mit der Motion-Platform möglich sein muss. Diese muss in der Lage sein, nicht nur die Neigung zu erfassen (siehe oben), sondern auch die Plattform zu bewegen. Mindestens muss das Force-Feedback dabei ein „Rütteln“ der Plattform ermöglichen, es kann aber auch die komplette Steuerung der Neigung gestatten (etwa durch Pneumatik-Zylinder).

Eine weitere Aktorik-Komponente muss die Einbindung von einem oder mehreren Ventilatoren sein, welche durch das System gesteuert werden können und eine Simulation eines „Flugwindes“ ermöglichen sollen.

Weitere Aktorik-Komponenten sind optional (etwa Nebel oder anderes), allerdings muss die Schnittstelle im System so spezifiziert und implementiert werden, dass die spätere Einbindung von Komponenten noch möglich ist.

B.2.5 Sound

Die Sound-Komponente soll die Realitätswirkung zusammen mit der Grafik unterstützen. Dementsprechend muss die Ausgabe des Sounds über mindestens vier Lautsprecher geschehen, so dass eine Raumpositionierung von Geräuschen möglich ist. Optional und wenn sinnvoll kann die Ausgabe um weitere Lautsprecher (Bass, Center) ergänzt werden.

Auf der Softwareseite muss die Sound-Komponente eine einfache Schnittstelle bereitstellen, welche folgendes erlauben muss:

- das Abspielen von Musik in einem noch zu spezifizierendem Format (MIDI, MP3, Ogg Vorbis, ...)
- das Abspielen von Soundsamples (WAV) bei einfacher Positionierung der Soundquelle im Raum
- die Kontrolle der derzeit abspielenden Samples oder der Musik (abbrechen, Position ändern, ...)

Optional kann die Sound-Komponente außerdem noch die Erzeugung von Effekten für die Sounds unterstützen, etwa Halleffekte oder ähnliches.

B.3 Nicht-Funktionale Anforderungen

B.3.1 Dokumentation

Allgemein muss am Ende des Projektes ein Projektbericht erstellt werden, der den Ablauf des Projektes dokumentiert. Zu jedem Semester muss außerdem ein Zwischenbericht angefertigt werden, welcher den Verlauf des Projektes in diesem Semester beschreibt. Die Zwischenberichte können später auch in den engültigen Projektbericht übernommen werden.

Zu jedem Projektplenum muss es ein Protokoll geben, in dem die Ergebnisse des Plenums etc. festgehalten werden (zu diesen Punkten siehe auch das Projekthandbuch).

Der Programmcode muss unter Berücksichtigung der gerade gültigen Coding-Guidelines dokumentiert werden, dadurch wird dann aus dem Programmcode mit einem Dokumentationssystem (Doxygen) eine technische Referenz in verschiedenen Formaten erzeugt werden.

Anhang C

Ideensammlung

C.1 Radsport Cave (Tina)

Ziel: Simulation einer Radtour

Zielgruppe: Radsportler, Leistungssportler, Amateure, Freizeitsportler

Cave: Der Cave soll insgesamt fünf Flächen umfassen. Vier seitliche Flächen und eine obere Fläche. Jede Fläche ist dabei in mehrere Frames (Abschnitte) eingeteilt. Es gibt ein Hauptframe, in dem die Tour simuliert wird und es gibt mehrere kleine Frames, die sich unterhalb des Hauptframes in einer Leiste befinden. Die kleinen Frames geben dem Sportler jeweils Auskunft über seine Pulsfrequenz, Daten über die Strecke, Daten über die Effektivität seiner Leistung. D.h. er wird dahingeführt, mit Hilfe dieses Programms eine optimale Lesitung zu erzielen. Es sollen realitätsnahe Strecken nachempfunden werden. Dabei sollen äußere Gegebenheiten, wie z.B. die Steigung eines Berges durch “Hochpumpen“ des Fahrrads mit Hilfe von einer speziellen Hydraulik nachempfunden werden. Die Strecke kann kurvenreich gestaltet werden. Dabei soll sich der Sportler in Kurven “legen“ könne, wobei sich dabei sein Fahrrad wirklich bewegen soll. Körperbewegungen sollen also auf dieses System übertragen werden, so dass das Ziel dieses Caves nicht bloß das Erreichen einer realitätsnahen Strecke sein soll, sondern der Sportler seine Leistung mit Hilfe seiner Werte optimal steigern und verbessern kann. Durch die Einrichtung mehrerer Kameras am Fahrrad kann der Sportler zusätzlich seine eigene Trittleistung und Technik sehen, erkennen und simultan verbessern und verändern.

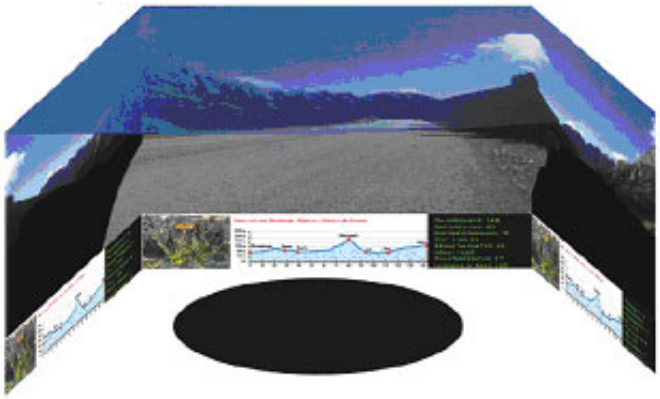


Abbildung C.1: Cave-Ansicht

Sensorik

Für die Trainingssteuerung im Radsport ist der aktuelle Kraftaufwand beim Treten ein wichtiger Faktor. Es dafür gibt bereits Systeme, die mittels einfach anzubringender Sensoren an Kettenstrebe, Kettenrad und Speiche drei Funktionen kontinuierlich anzeigen und speichern.

Power (Krafteinsatz in Watt) kontinuierliche Messung; der durchschnittliche Krafteinsatz wird aus den jeweils letzten Pedalumdrehungen errechnet.

Left/Right Balance (Links/Rechts - Verteilung) prozentuale Verteilung des Krafteinsatzes zwischen linkem und rechtem Pedal.

Pedalling Index (Gleichmäßigkeit des Antritts) Beschreibt das Verhältnis zwischen maximalem und minimalem Krafteinsatz während einer Pedalumdrehung. Es gibt eine Software, welche Messwerte aus einer Leistungsanalyse (Lactat, Herzfrequenz, Erholungspuls, etc.) speichert, verwaltet, auswertet und analysiert. So ist es beispielsweise möglich, die Trainingszustände zu verschiedenen Zeitpunkten oder zwischen verschiedenen Sportlern zu vergleichen. (siehe Anhang)

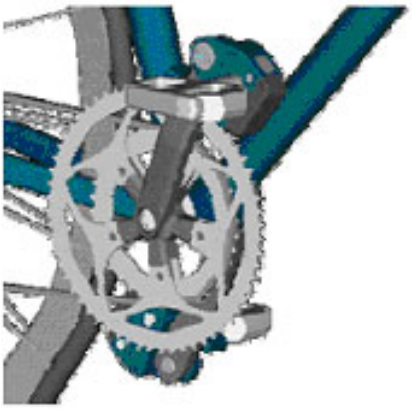


Abbildung C.2: Pedale

- Pulsuhr
- Kurvenlage
- Körperschwerpunkt (wann Kurve, wann Fahrradbewegung?; bei Schwenken des Fahrrads: Körpergewicht in entgegensetzter Richtung des Fahrrads; bei Kurve: Körpergewicht auf gleicher Seite des Fahrrads; d.h. bei wie viel Gewicht prozentual handelt es sich um eine Kurve und wann um eine Strecke geradeaus)
- Bremsung
- Beweglichkeit des Fahrrads

Technik

Fahrrad Das Fahrrad ist eine Mischung aus einem Heimtrainer und einem Rennrad. Das Fahrrad soll durchaus eine vollelektronische Wirbelstrombremse** besitzen, die drehzahlunabhängig den Belastungswiderstand regelt. Gleichzeitig soll jedoch die Wendigkeit eines Rennrades nicht verloren

gehen. D.h. der Sportler soll sich durchaus mit seinem Körpergewicht gegen das Fahrrad legen können, wenn er beispielsweise einen Berg hinauf fährt.

Fahrradstützen Das Fahrrad wird am Hinterrad durch Stützen fixiert. Dies bewirkt, dass das Fahrrad bei einer Kurvenbewegung oder bei einer Bergab- bzw. auffahrt nicht von der Plattform rutscht und dennoch eine reale Fahrradbewegung ermöglicht wird.

Plattform Das Fahrrad befindet sich auf einer kreisförmigen Plattform. Diese Plattform liegt auf vier Hydraulik Hebern auf. Diese Hydraulik Heber wiederum befinden sich in einer kreisförmigen Schiene. Die Heber realisieren eine Bergauffahrt und die Lehnung in eine Kurve.

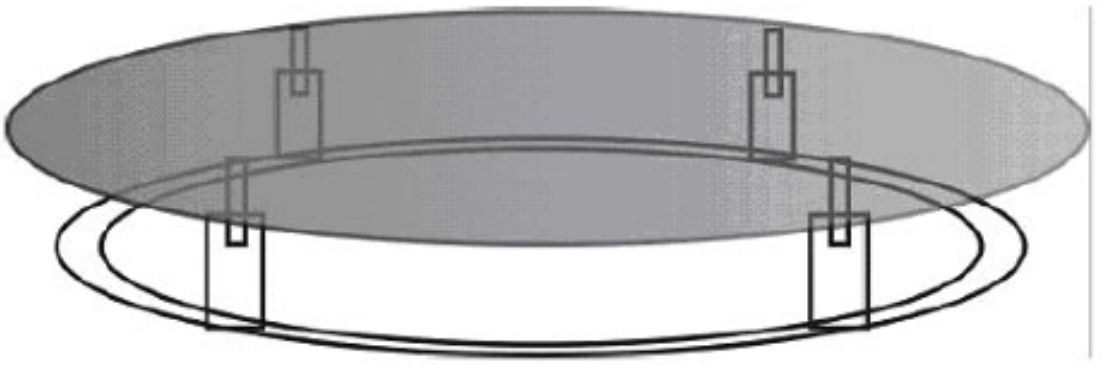


Abbildung C.3: Plattform

Sportler Der Sportler trägt eine Pulsuhr, mit der seine Frequenzen gemessen werden, anhand derer er eine Leistung optimieren kann.

Kamera Rings um das Fahrrad herum sind mehrere Kameras eingerichtet, mit denen der Sportler zusätzlich seine eigene Trittleistung und Technik sehen, erkennen, simultan verbessern und verändern kann.

Leistung

- Leistungsanzeige in Watt
- Trittfrequenz
- Zeitmessung
- Energieverbrauch
- Trainingsstrecke
- Gesamtkilometer
- Geschwindigkeit
- Raumtemperatur
- Pulsmessung
- Trainingspuls mit prozentualrer Anzeige zur vorgegebenen Pulsobergrenze
- Durchschnittswerte bei Trainingsende für Puls

Auswahlmöglichkeiten

- Leistung

- Zeit
- Energieverbrauch (maximal)
- Strecke (Tageshöhenmeter, Gesamthöhenmeter, Steigung/Gefälle in Prozent, aktuelle Höhe)
- Optimaler Trainingspulsbereich
- Pulsobergrenze

Messungen elektronische Pulsmessung wahlweise über

- Ohrclip
- Handpulssensoren
- Pulsuhr
- Verschiedene gespeicherte Trainingsprogramme, Pulsgesteuertes Training - Vergleich der individuellen Leistung

Leistungsverbesserung bei gleicher Strecke und gleichen Gegebenheiten

- Manuelle Belastungssteuerung (Watt)
- Übertragung der Herzfrequenz
- Angabe der maximalen Herzfrequenz
- Durchschnittliche Herzfrequenz

Fahrrad

- Vollelektronische Wirbelstrombremse regelt drehzahlunabhängig den Belastungswiderstand
- Gespeicherte Wattgesteuerte Trainingsprogramme
- Pulsgesteuertes Programm
- Manuelle Belastungssteuerung -> dosierbares individuelles Training

Anhang

Das Messsystem soll die ins Pedal- und Kurbelsystem eingebrachten Kräfte und damit die Leistung und das Drehmoment für beide Seiten (rechts/links) getrennt messen. Damit ist die Bestimmung der vortriebswirksamen Kraft und des biomechanischen Wirkungsgrades für das einzelne Bein möglich. Die vortriebswirksame Kraft und der biomechanische Wirkungsgrad sind Kriterien des Tretmusters beim Radfahren. Das Tretmuster hat sowohl unter leistungssportlichen als auch unter orthopädischen Gesichtspunkten eine hohe Bedeutung. Für die Leistungsmaximierung im Radsport ist es notwendig, die vortriebswirksamen Kräfte zu optimieren, was häufig alltagssprachig mit dem so genannten “runden Tritt“ bezeichnet wird. Die auf dem Markt erhältlichen Messsysteme lassen keine bilaterale Erfassung der Kräfte und damit der Leistung oder des Drehmoments zu. Das zu entwickelnde Messsystem soll möglichst einfach aufgebaut und an die handelsüblichen Pedalsysteme ohne Umbauten montierbar sein. Die über den Leistungssensor gemessenen Daten werden von einem Mikrokontroller vor-ausgewertet und einmal pro Kurbelumdrehung drahtlos an die Auswerteeinheit gesendet. Die Auswerteeinheit soll aus einer Koppelstation mit integriertem Empfangsmodul und einem handelsüblichen Handheld bestehen. Am Heimcomputer sollen die Daten vom Handheld übertragen werden können, um über eine zu entwickelnde Software die Analyse der Leistungs- und Drehmomentverteilung vornehmen und Informationen zu einer entsprechenden biomechanischen Optimierung im Training ausgeben zu können. Die Software soll biomechanische Informationen in alltagstaugliche trainingswissenschaftliche Anleitungen umsetzen.

Hall Sensorik Die Sensoren sind speziell für die berührungslose Erfassung von Drehzahlen an rotierenden Maschinenteilen konzipiert. Je nach Aufgabenstellung, Einsatzgebiet und Frequenzbereich werden unterschiedliche Messprinzipien angewandt. Der Sensor detektiert die Bewegung von ferromagnetischen Strukturen, wie zum Beispiel Zahnrädern, über die Veränderung des magnetischen Fluss. Das Sensorelement ist mit einem Permanentmagneten vorgespannt. Ein Zahn oder eine Lücke, die sich am Sensor vorbei bewegt, beeinflusst das Magnetfeld. Dadurch wird bei einem Hallsensor eine Änderung der Hallspannung erzeugt. Die Magnetfeldänderungen sind somit in eine elektrische Größe umsetzbar und werden entsprechend gefiltert und aufbereitet. Das Ausgangssignal des Sensors ist eine Rechteckspannung, welche die Magnetfeldänderung widerspiegelt.

Wirbelstrombremse Bewegt sich ein elektrischer Leiter in einem Magnetfeld, wird eine Spannung induziert, die einen Stromfluss bewirkt. Hat der elektrische Leiter die Form eine Scheibe, bilden sich in ihr kreisende Wirbelströme aus, die ein eigenes Magnetfeld hervorrufen. Nach der Regel von Lenz (1834) wirken beide Magnetfelder so ineinander, dass die drehende Scheibe gebremst wird.

C.2 Trainingssimulator (Tobias)

Mein Modell lehnt sich sehr an modernen Trainingssimulatoren an, welche eine bestimmte Strecke vorgeben, die dem Benutzer eine möglichst realitätsnahe Simulation präsentiert.

Auch mein Cave soll die Möglichkeit zur Trainingssimulation im freien Gelände geben. Allerdings gehe ich von einer echtzeitgerenderten Simulation aus, welche durch die Körperwerte des Benutzers bestimmt werden.

Ich versuche damit dem Anspruch an den Cave genugzutun, eine Extrem- oder Grenzerfahrung zu liefern.

Meiner Meinung nach bilden Drogen, sowie extreme körperliche Anstrengungen, wie z.B. Marathonläufe mit dem sogenannten “runners high“ eine Art der nicht mehr realen, aber auf realen Grundelementen basierenden Wahrnehmung.

Anhand einer echtzeitberechneten Welt, bzw. das Ändern gewisser Parameter dieser Umgebung anhand der physischen Werte des Benutzers könnte ein solcher Zustand simuliert werden.

Kurz:

Der Läufer baut sich seinen eigenen Wald. Der Radfahrer kreiert sich seine Alpen, ...

Neben der wahrscheinlich sehr überwältigen Landschaftskulisse (sie ist immer so wie man sich gerade fühlt...) würde man ein perfekt auf die Bedürfnisse des Sportlers zugeschnittenes Trainingsprogramm entwickeln. Es wäre außerdem möglich, dass für Ausdauerathleten sehr wichtige gleichmäßige Belasten des Körpers zu trainieren, da man direkt in den Trainigsprozess eingreifen kann, ohne das Training zu unterbrechen.

Aufwand:

Kosten: Laufband (bzw. Fahrradrolle, Spinningrad,...) Ca. 120 - 400 Euro

Cave: Haben wir!

Software zum Dateninput: Selbst schreiben!

Interface zur Überwachung der Körperwerte (Puls- Temperaturmesser,...) Gibt's entweder beim Trainingsgerät, oder ca. 100-200 Euro

(3x) Rechner zur Erstellung Überwachung und dem Erstellen der VR: Haben wir!

(3x) Beamer: Haben wir (manchmal...)!

Hier eine Skizze eines Caves mit einem Laufband:

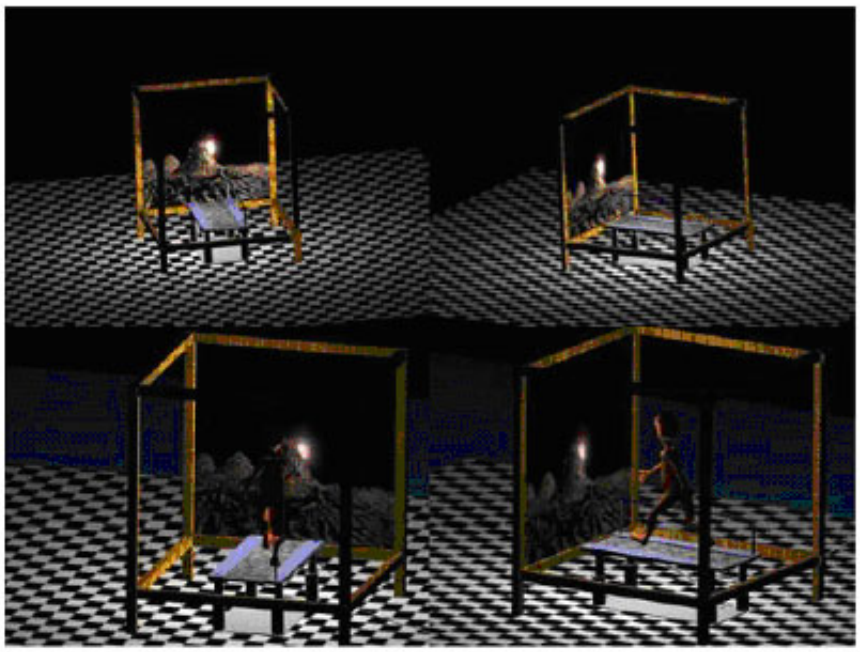


Abbildung C.4: Trainingssimulator

Anmerkung Die Zylinder simulieren Kurven und Berge und an den Pfosten können weitere Aktoren, wie z.B. Ventilatoren, Boxen,.... angebracht werden.

C.3 Simulation eines fliegenden Teppichs (Manfred)

Thema dieses Caves war es, die Simulation eines fliegenden Teppichs. Die Bewegung des Teppichs sollte durch ca. vier pneumatische Pumpen, die sich an den vier Ecken der Sitzfläche befinden, simuliert. So sollte ein Kippen in mehr als vier Richtungen möglich sein, wie z.B. nach schräg vorne links. Es stellt sich natürlich die Frage nach einer einfachen Steuerung des Teppichs. Die Idee dabei war, die Pumpen als Waage zu verwenden, die eine Gewichtsverlagerung des Benutzers messen könnte. So wäre es möglich, den Teppich durch z.B. Gewichtsverlagerung nach vorne kippen zu lassen. Der Sinn des Caves ist es, auf einem Teppich über bzw. durch ein Areal zu fliegen und sich Dogflights mit feindlichen Teppichen o.ä zu liefern. Das Attackieren sollte hierbei über bestimmte Handbewegungen geschehen, die von einem Datenhandschuh erfasst würden. So sollte es nach dem Erlernen einiger Bewegungsabfolgen möglich sein, Feuerbälle zu werfen, sich zu heilen, seine eigene Geschwindigkeit zu erhöhen etc. Es wäre möglich, hier noch eine Art Würfel einzubauen, der je nach Schwere der Zauber mit einem Faktor würfelt, um den Zauber zu wirken, bzw. fehlschlagen zu lassen, ähnlich, wie es in PC-Rollenspielen wie "Baldurs Gate" und dergleichen praktiziert wird. Es wären fünf Wände mindestens sinnvoll; vier an den

Seiten, und entweder eine weitere Leinwand oben oder unten. Dies hinge vor allem von der konkreten Realisierung ab. Hätte der Boden Bedeutung, so wäre eine Leinwand unten nützlich; ansonsten wäre eine Darstellung des Himmels für “atmosphärisch“ unerlässlich. Das Areal sollte aus einem hügeligen offenen Gelände bestehen. Ein Vorteil des Fliegenden Teppichs wäre die Tatsache, dass man die zur Verfügung stehende Fast-Rundumsicht ausnutzen würde, da man sich in drei Dimensionen bewegen könnte, wenn auch nur eingeschränkt. Die zu realisierende Steuerung und die Zauber bieten reichhaltige Interaktionsmöglichkeiten mit der Welt.

C.4 Universumserkundung(Björn)

Der Cave soll die Begehung des Universums simulieren. Planeten, Sterne, Galaxien, schwarze Löcher, Meteore, Asteroiden sollen in einer begrenzten Ausdehnung dargestellt werden. Die Objekte in diesem Raum bekommen genaue physikalische Werte zugeordnet. Die Interaktion soll erlauben, Planeten bzw. Galaxien neu anzuordnen, wobei die Änderung der physikalischen Zusammenhänge, Gravitationskonstanten, die dann neu wirken, berücksichtigt werden müssen. Der Cave soll deutlich die Ausgewogenheit der Planetenanordnung darstellen. Durch Veränderung werden Konstrukte instabil. Die Interaktion, das Greifen von Planeten etc. soll über pneumatisch-gestützte Apparaturen geschehen. Die Bewegung im Cave soll einen gewissen Grad an Schwerelosigkeit demonstrieren. Evtl. kann eine Art Abenteuer (s. “Der Kleine Prinz“ oder “Roger Wilco“) dargestellt werden. Die Ausdehnung zu einem Spiel muss als optionales Add-on gesehen werden. Für die Realisierung eines Spiels kann entweder eine Sportsimulation (Squash oder Planeten-Tennis) herangezogen werden. Ebenso ist es möglich, den Faktor an Spannung durch eine Verteidigungssimulation zu erhöhen (s. “Armageddon“). Je nach Einsatz müssen andere Möglichkeiten der Interaktion gestellt sein. Für eine Sportsimulation ist es sinnvoll eine Art Force-Feedback-Tennisschläger zu entwickeln, welcher beim Schlagen eines Meteors das entsprechende Abbremsen an den Akteur weiter gibt. Eine Verteidigungssimulation benötigt eine entsprechende Apparatur, mit der es möglich ist, herannahende Meteore abzubremsen (zu stoppen). Des Weiteren ist es möglich, den Akteur zunächst ein Spielfeld -durch Versetzen der Planeten etc.- aufbauen zu lassen, welches dieser dann in einer Simulation nutzen kann.

Als Weltentyp wurde für diesen Cave-Entwurf ein offenes Areal angestrebt. Weltbestandteile sollten vor allem Planeten, Asteroiden, Meteore, etc. sein. Die Spiel-Aufgabe sollte nicht eingeschränkt sein und könnte variiert werden. Möglich wäre das Spiel mit dem Gleichgewicht der Galaxie. Über eine Stahlseilaufhängung würde die Bewegungsrichtung erfasst werden. Weiter dient diese dazu, den Akteur in die Physik der virtuellen Welt zu überführen. Je nach Gewichtsverlagerung würde kalkuliert werden, wo die Hauptlast wäre und entsprechend in diese Richtung bewegt werden. Außerdem könnten zur Interaktion mit Objekten Trackinghandschuhe oder gar Schlaggeräte verwendet werden. Vom System sollte als Rückkopplung auch die Funktionalität von Force-Feedback den Akteur erreichen. Die Trackinghandschuhe könnten mittels Pneumatik das Greifen eines Objektes realisieren. Ein virtuelles Objekt sollte dadurch ein realistisches Greifgefühl widergeben. Über akustische Signale sollten Aufprallen etc. an den Rezipienten zurückgegeben. Der Cave soll in seinen Seitenflächen abgeschlossen sein, d.h. es würden 5 Seitenflächen benötigt werden. Die Höhe des Caves müsste der höchsten Bewegung, die sich in dem Cave ausführen lässt, entsprechen. In dem Cave müsste genügend Platz sein, um die Arme in jede Richtung ausstrecken zu können, ohne dabei in Berührung mit dem Konstrukt zu kommen. Materialien für das Konstrukt wären beliebig, solange sie den Zweck erfüllen würden. Der/die AkteurIn sollte sich in einer Aufhängung aus Stahlseilen, verbunden mit Federn, befinden. Diese Aufhängung müsste sich in alle Richtungen drehen lassen und sollte den Körper insgesamt zu einem bestimmten Grad entlasten. Der/die AkteurIn könnte sich über die Bewegung in eine bestimmte Richtung durch das Szenario navigieren. Je nach Stärke des Zugs an einem bestimmten Seil, würde die Bewegung in eine eben jene Richtung veranlasst werden.

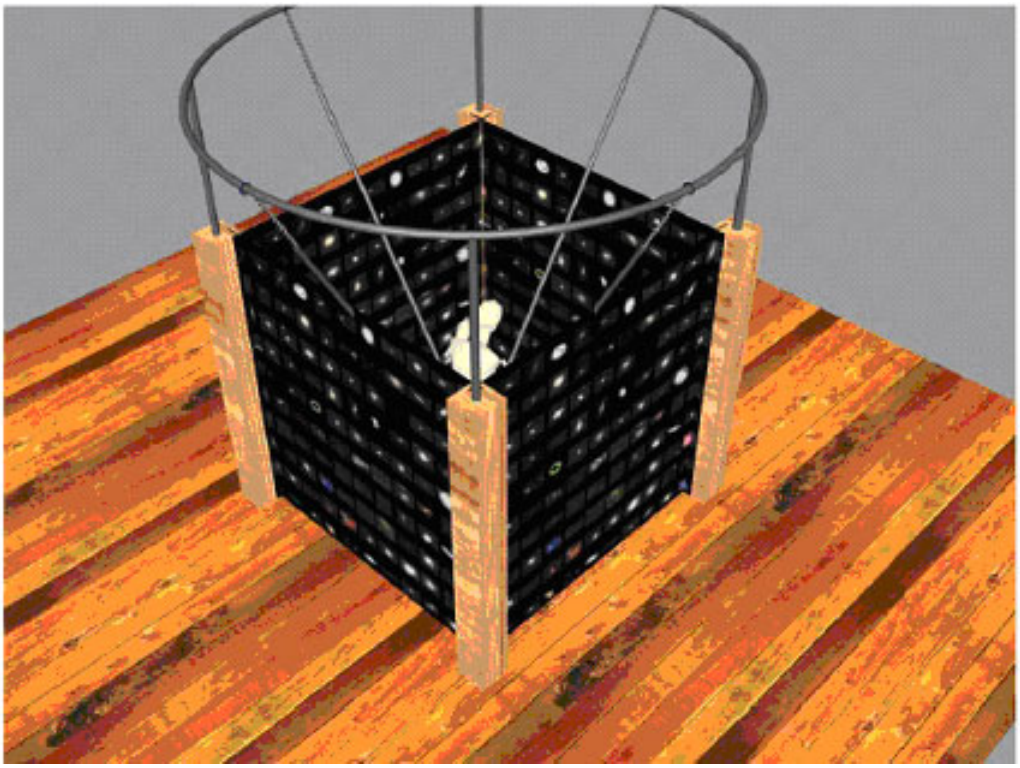


Abbildung C.5: Universumserkundung Abb. 1

C.5 My Mechwarrior (Christopher)

Mechwarrior ist eine Serie von PC-Spielen, die sich in dem Science Fiction Szenario Battletech abspielt. Im Kern geht es um Kriegsführung mit bis zu 10 Meter hohen Kampfrobotern, die von einem Piloten dem Mechwarrior gesteuert werden und mit Lasern, schweren Autokanonen und Raketen bewaffnet sind.

Welttypen

große bergige Areale, Städte, Wälder

Weltbestandteile Die Kampfroboter (Mechs), andere Fahzeuge, Häuser, Infrastruktur, ...

Spielaufgabe Missionen wie zB Zerstöre alles, beschütze Konvoi, Angriffe auf Basen, Auskundschaften des Feindes, ...

Input 2 Joysticks zum steuern des Mechs. 1 für Torso, der andere für die Beine.

Output

Plattform, die durch an den 4 Ecken angebrachte Druckzylinder gesteuert gehoben/gesenkt werden kann.

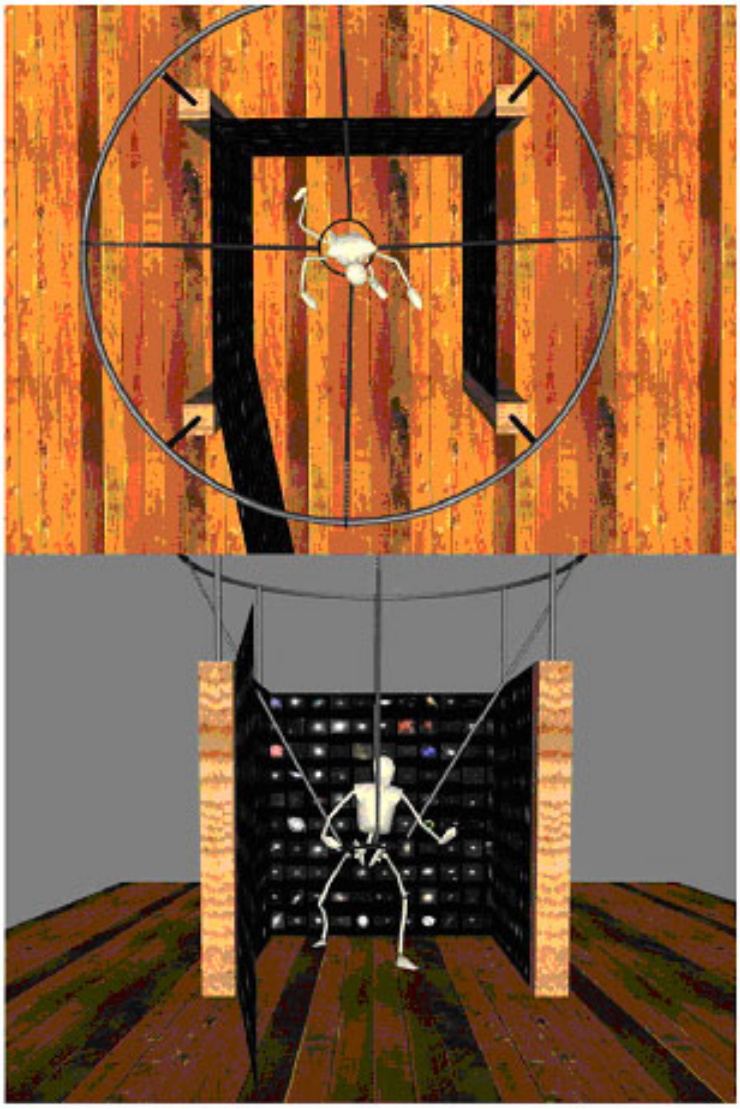


Abbildung C.6: Universumserkundung Abb.2

Welt

Statusupdates durch trigger (also beim erreichen von einem Punkt im Spiel wird ein Skript ausgelöst).

Physik, KI, User Input, ...

minmale größe = ähm recht groß, wir wollen doch lange und große kämpfe haben :-)

Multiplayer: ja klar, ist echt cool (wohl aber nur feature)

Weshalb ist das mein Vorschlag?

Die Mechwarrior Serie war schon cool, aber mit echter Rundumsicht und Force Feedback könnte diese Simulation echt cool sein. Schon allein das Gefühl wenn der 20 - 100 tonnenschwere Mech durch die Welt stapft kann sehr gut durch eine Simulation rüberkommen, oder wenn der Mech sich unter einer Raketensalve schüttelt und bockt, bis man ihn wieder unter Kontrolle hat kann viel besser als am Rechner zu Hause dargestellt werden. Ist ja auch nur ein Vorschlag.

Cave:

3 Wände sollten reichen, aber man nimmt was man kriegen kann. Wahrscheinlich müsste man dann auch ein Cockpit auf die Plattform platzieren um dem Anwender das Gefühl zu geben in eben jenem zu sitzen.

Aufwand

Engine 2 - 3 Jahre, kommt aber auch darauf an wieviel implementiert werden soll 3D Engine von null 6 Monate + x

Weltrepräsentation 3 - 6 Monate (plus etliche Bugfixes)

Input 1 Monate

Output div. Monate

KI, (eigentlich Content) 3 Monate (eher Kanonenfutter)

Physik 3 Monate + x

Tools 9 Monate (sehr wichtig und immer eine "Fehlerfinde-Quelle")

Cave-Anpassungen (also Kommunikation zwischen den 3 Rechnern der Darstellung) 3 Monate

Output 4 - 8 Wochen

Content, Objekte wenn ich diese machen soll, 2 - 3 Wochen (sehen dann zwar schlecht aus, aber ...)

Content, Missionen kommt auf die Komplexität an, pro Mission, wenn der andere Content steht, 2 - 4 Wochen

C.6 Unterwasser/U-Boot-Simulation (Tatiana)

Der Unterwasser-Cave soll sich als ein Adventurespiel darstellen, das unter Wasser stattfindet. Die Welt besteht aus Dingen die zu einer Unterwasserwelt eben gehören: Fische, gesunkene Schiffe, Wasserpflanzen etc. Ein Ziel des Spiel kann z.B. sein, ein Geheimnis herauszufinden, wozu man verschiedene Rätsel lösen soll. Bewegt man sich in dieser Welt mit Hilfe eines Batiskaphes. Der Spieler befindet sich in einem Sessel, der in der Mitte des Caves steht. Dabei sind auch verschiedene Steuerelemente möglich, wie z.B. Joystick oder 3D-Mouse, um das Spiel zu steuern und verschiedene Tablos, auf denen man stehen kann, wie tief das Wasser an dieser Stelle ist etc. Das Spiel fängt oberhalb des Wassers an, dann sinkt man langsam nach unter und je tiefer man ist, desto dunkler wird die Welt. Ab einer bestimmten Tiefe bracht man Scheinwerfer. Um das alles realistischer zu machen, soll das Geräusch von U-Booten auch dabei sein. Außerdem, falls das Batiskaph an irgendetwas stößt, soll der Sessel beben.

C.7 Fliegender Teppich (Patrick R.)

Allgemein

Bei den Überlegungen zu einer konkreten Anwendung schweben mir mehrere Kriterien vor, die ich ganz gerne umsetzen würde. Zum einen der Einsatz von unterschiedlichen Interaktionsmöglichkeiten mit der Anwendung, welche ein regelrechtes Lernen des Benutzers voraussetzen. Hierzu gehört z.B. das

Abbildung C.7: U-Boot-Sessel

Ausnutzen der völlig freien Positionsveränderung in einem dreidimensionalen Raum. Zum anderen der spielerische Charakter der Anwendung. Ich denke, um den Spaß nicht zu kurz kommen zu lassen und um wesentlich mehr Freiheiten bei der Wahl der Technik zu haben, sollte die Anwendung ein Szenario mit Eigenschaften von Spielen unterschiedlichen Genres sein. Dies erfordert dann nicht nur Geschick bei dem Umgang mit den verschiedenen Interfaces, sondern beansprucht zugleich auch noch analytische und kombinatorische Fähigkeiten um in der Anwendung selber fortzuschreiten. In den folgenden Punkten möchte ich auf den Einsatz möglicher Interaktions- und Steuerungstechniken eingehen, wie auch auf das inhaltliche einer solchen Anwendung.

Grundlage - Magic Carpet

Als Grundlage für die Anwendung sehe ich eine Spieleidee, welche im Jahre 1994 von Electronic Arts auf den Markt gebracht wurde - Magic Carpet. Durch den Einsatz eines fliegenden Teppichs haben wir eine schöne Möglichkeit das vollkommen freie Navigieren in einer dreidimensionalen Welt zu realisieren. Auf die Technik der Steuerung gehe ich später ein, jedoch soll hier kurz erwähnt werden, dass ich mir eine geradlinige Vorwärtsbewegung vorstelle, welche - genauso, wie bei einem einem Flugzeug - Veränderung der Höhen- und Seitenposition nur durch das Anpassen der entsprechenden Neigung ermöglicht. Die Interaktion mit aktiven Objekten in dem sichtbaren Bereich der Anwendung, ist eine Grundvoraussetzung einer jeden Implementierung eines solchen Systems. Soweit ich mich erinnere ist dies bei Magic Carpet z.B. das Aufsammeln von magischer Energie, oder der Luftkampf mit anderen Besitzern fliegender Teppiche. Eine weitere interessante Idee, die ich gerne mit einbauen möchte ist die Sprachsteuerung gewisser Aktionen. So können mittels eines Headsets “Befehle“ an ein Spracherkennungsprogramm übertragen werden, welche dann Events auslösen. In Bezug auf eine Anwendung mit Magie kann ich mir da ganz gut den Einsatz von Zaubersprüchen vorstellen.

Konstruktion des Caves

Anzahl Projektionsflächen Der Cave hat 4 Projektionsflächen. 3 Seitenteile und wahlweise Boden oder Decke, wobei ich denke, dass eine angestrahlte Bodenfläche bei einer Welt ohne Bodenkontakt noch wesentlich interessantere Eindrücke vermitteln kann, als die Deckenfläche.

Maße Da ich nicht an den Einsatz von schwerem Gerät mit großen Abmessungen denke, können die Maße des Prototypen übernommen werden.

Content (Inhalt der Welt)

Weltentyp große, wenn nicht sogar riesige, weitläufige Welten, über hügelige Landschaften, begrenzt durch z.B. Wasser oder Berge. Interessant sind auch Bergmassive mit Schluchten, so wird der völlig freie Bewegung ein paar Grenzen diktiert.

Weltbestandteile Viele magische Gegenstände und Wesen. Elfen, Trolle, Orks und Greifen. Tiere allgemein sind denke ich immer willkommen. Andere fliegende Magier werden auf jeden Fall gebraucht, denn es geht schließlich nicht ohne Gegner. Ebenso wichtig riesige Bäume in denen man dann im 15. Stock den Teppich parken kann.

“Spiel“-Aufgabe Das Sammeln von magischer Energie um Zugang zu gewissen Rätseln zu erlangen. Das Lösen von Rätseln ermöglicht dann den Zugang zu anderen Welten. Die Energie kann und soll auch dazu eingesetzt werden, um den Umfang von Zaubersprüchen zu erweitern und um diese zu verbessern. Um es auf den Punkt zu bringen, so stelle ich mir die Aufgabe vor. Setze alles Böse was Dir beim Lösen der Rätsel im Weg ist mit Hilfe Deines fliegerischen Könnens und dem Einsatz von Magie außer Gefecht, verwalte Deine Ressourcen sinnvoll und strenge Dich an die unsagbar schweren Aufgaben zu lösen, um so den Zugang zu immer schöneren Welten zu erlangen.

Input

Steuerung des Teppichs - Richtungsänderung Als Plattform soll hier eine waagen-ähnliche Fläche dienen, auf der der Benutzer an festgelegten Stellen mit den Füssen Platz nimmt. Durch Gewichtsverlagerung wird dann eine Verschiebung des Schwerpunktes registriert, welches eine Änderung der Richtung bewirkt. Dies ist eine der zwei Komponenten, welche die Bewegung der Szene beeinflussen.

Steuerung des Teppichs - geradlinige Beschleunigung Hier sehe ich mehrere Möglichkeiten, welche realisiert werden können. Zum Beispiel eine feste Beschleunigung, die keine Veränderungen durch den Nutzer erlaubt, oder aber ein zusätzliches Eingabegerät wir einen einfachen Schalter, welcher bei Druck beschleunigt und beim Loslassen die Beschleunigung abbricht. Eine weitere Möglichkeit wäre das Starten und Stoppen der Beschleunigung über die akkustische Steuerung (s.u.).

Akkustische Eingabe Ansteuerung eines Spracherkennungssystem über ein Headset. Kurze prägnante Worte (Befehle) lösen Zaubersprüche aus. Bewegung der Szene wird dadurch nicht beeinflusst, es sei denn, die Spracheingabe wird auch dazu verwendet, die geradlinige Bewegung zu steuern. Dies z.B. über zwei Befehle (“start“, “Stopp“).

Zielsystem Um mit Objekten agieren zu können, müssen diese anvisiert werden. Dies kann mittels eines Trackingssystems realisiert werden. So z.B. mit 4 Sensoren in einem Magnetfeld, welche ein Zwei-Achsensystem bilden, wobei die Achsen senkrecht aufeinander stehen und die Sensoren in einem Datenhandschuh untergebracht sind.

Output

Ein Feedback in der Form, dass die Plattform zur Steuerung zu rütteln beginnt, wenn der Spieler von einem Gegner getroffen, oder den Teppich gegen ein Hindernis lenkt. Das Rütteln wäre sicherlich mit Mitteln der Pneumatik zu realisieren.

Grafik

Realisierung von Wolken, in die sich der Spieler zurückziehen kann, um sich vor Angriffen zu verstecken (Feigling).

Visualisierung der Zaubersprüche (Feuerball, Manaregen, Blitz, etc.) Welt

KI: KI-Gegner (s.o.).

C.8 Fliegender Teppich (Franck)

Der fliegende Teppich besteht aus 4 Zylindern,die in einer geeigneten Position für die Erhaltung der Stabilität der Sitzfläche verteilt, eine flache und Quadratische Sitzfläche und eine andere Materiale wie einen Zylinder, deren Spitze aus einer Kugelform ist und Aktivitäten die Bewegung der Sitzfläche als Rolle haben.

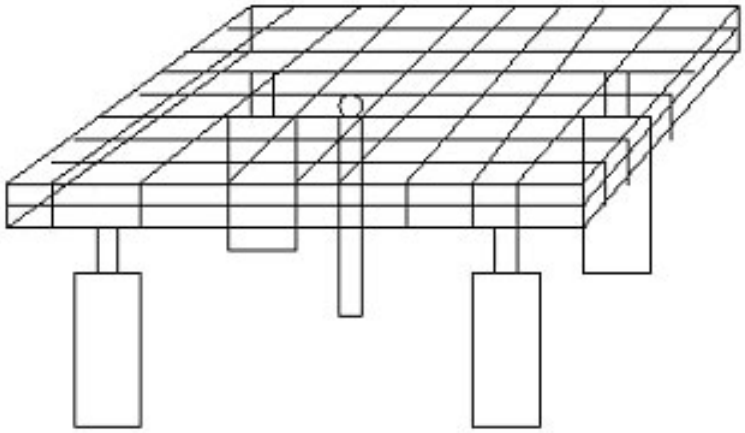


Abbildung C.8: Fliegender Teppich Abb. 1

Funktionalität

Um der Teppich in Aktivitäten zu bringen, soll man auf diesen steigen und sich auf die quadratische Platte setzen. Wobei alle 4 Zylinder in ihren Ruhigen Zuständen befinden, die genau für jede koppelte Ventilschaltung auf die 4 Druckluftleitungen, die den Druckluft in jedem der 4 Zylinder führen, den Wert 1 entsprechen. Je nachdem Benutzer seinen Körper in einer bestimmten Position lenkt, ändert sich die Position der Stange in jeder jede Zylinder . und so wird die beschriebene Form der Sitzfläche (der so genannte fliegende Teppich) eine Aktion auflöst, die gleichzeitig auf die virtuelle Welt in der Cave ein Geschehnis bestimmt.

Wie soll die Steuerung der Zylindern organisiert werden?

Die Bewegungen der Stangen in der Zylinder sollen Synchronisiert sein. Bei Auflösung einer Lenkung der Benutzer gescheht folgenden: Wenn einige Stange von meisten Zylinder nach unten gehen, gleichzeitig müssen die Anderen, auf die keinen Kräfte der Lenkung der Benutzer ausgeübt werden, nach oben steigen. Auf das Bild unten kann man ein Beispiel der Position der Stangen in Zylinder A, B, C und D sehen, wenn der Kraft der Benutzer nach Link, nach Recht, nach vorn oder nach Hinten ausgeübt wird.

Was funktioniert die Ventilschaltung?

Auf das System unten, wird eine Ventilschaltung mit 3-Anschlüße und 2 Stellung (0,1) benutzt, je nach dem einem Ventil aktiviert wird (Stellung =1), wird der Druckluft in bestimmten Zylindern geführt, und in anderen die Luft befreiet (Stellung = 0).

Sei z das Signal
$z = 0_A1 = 1_B1 = 1_C1 = 1_D1 = 1$ Der Teppich fliegt geradeaus
$z = 0_A1 = 0_B1 = 0_C = 0_D = 0$ Der Teppich fliegt geradeaus
$z = 1_A1 = 1, B1 = 1, C1 = 1D1 = 0$ Teppich in schiefer Position
$z = 1_A1 = 1, B1 = 0, C1 = 1, D1 = 1$ Teppich in schiefer Position
$z = 1_A1 = 1, B1 = 1, C1 = 0, D1 = 1$ Teppich in schiefer Position
$z = 1_A1 = 1, B1 = 1, C = 0, D1 = 0$ Teppich nach links, nach rechts oder nach vorn geneigt.
etc.

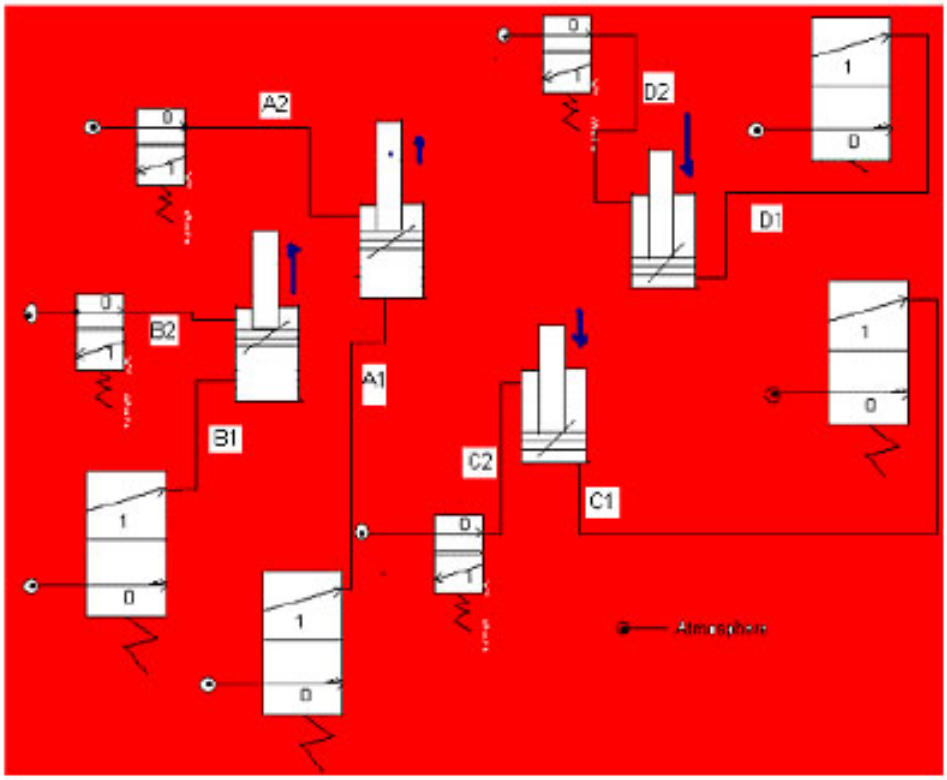


Abbildung C.9: Ventilschaltung

Nachteil: Der Druckluft ist höher als die Kraft des Benutzers

Anstrengung

Gesundheitsproblem

Verlust des Spaß

Nicht für alle Menschen geeignet

C.9 Flugsimulation (Elmar)

Content

Mehrere Möglichkeiten sind denkbar:

Flugsimulation über eine Landschaft (Weltentyp: offen) Die dargestellte Welt kann sowohl eine fiktive Landschaft sein, kann aber auch eine bestehende Landschaft sein (z.B. eine bekannte Stadt oder generell ein bekanntes Gebiet). Dabei kann es sich einerseits um eine Luftkampfsimulation mit auszuführenden Missionen handeln. Zu Beginn kann es eine Art Briefing/Einführung geben, die die auszuführende Mission spezifiziert. Anschließend soll der Cave-Nutzer diese Mission nachspielen. Punktemäßig kann es am Ende der Simulation einen Abschluss geben, der den Erfolg der Mission interpretiert. Im Spiel gibt es beispielsweise andere Flugzeuge, die dem Spieler feindlich oder freundlich gesonnen sind. Der Spieler kann diese abschließen oder von diesen abgeschossen werden. Ebenfalls kann es auf der Oberfläche der Landschaft Ziele geben, die der Benutzer zerstören soll, bzw. die auch Maßnahmen gegen den Spieler erheben. Andererseits kann es sich bei dieser Anwendung um einen simulierten Flug in einem historischen Fluggerät handeln oder über eine interessante Landschaft. Beispielhaft sei hier New York mit bestehendem World Trade Center, die Alpen, der Grand Canyon oder ähnliches zu nennen.

Flugsimulation durch das Weltall (Weltentyp: offen) Analog zum Flugsimulator durch eine Landschaft ist es auch denkbar, einen futuristischen Flugsimulator zu bauen, der den Flug außerhalb der Erdatmosphäre simuliert. Auch hier ist eine militärische Simulation möglich, in der es ebenfalls eine Mission zu erfüllen gibt, die anschließend bewertet wird. Ebenso ist aber auch hier ein Flug durch das Sonnensystem denkbar, eine Mondlandung nachzuempfinden oder ähnliches.

Flugsimulation durch ein Höhlensystem (Weltentyp: begrenzt) Ebenfalls analog zu den anderen Möglichkeiten soll in einem solchen Simulator das Spielziel das “Zerstören von Gegnern“ sein. Dabei sollen sinngemäß Treffer durch gegnerische Geschosse oder die Kollision mit der Wand zum Verlieren des Spiels führen.

Input

Als Eingabemöglichkeit in die Simulation sind folgende Systeme denkbar:

Joystick oder Steuerrad zur Flug-Richtungsbestimmung Der Joystick oder das Steuerrad sollen analog zu real existierenden Systemen funktionieren. Eine Bewegung des Eingabegerätes nach links oder rechts führt zu einer Neigung des Fluggeräts in die entsprechende Richtung. Das Drücken oder Ziehen führt zum Sink- oder Steigflug.

Diverse Schalter Schalter/Buttons können zur Interaktion mit dem System genutzt werden. Beispielhaft seien hier das Starten des Flugzeugs, das Laden von bestimmten “Waffensystemen“, Aktivierung eines Funkgeräts und anderer Funktionen, die für den Flugbetrieb notwendig sind.

Geschwindigkeitspedal oder Gashebel Entsprechend der Einstellung dieses Interaktionsgeräts soll die Geschwindigkeit des Fluggeräts gesteuert werden.

Spracheingabe ("Funkgerät") Mit einer Spracheingabesteuerung ist es möglich, einen Funkverkehr (mit vordefinierten Befehlen, die zu erlernen sind) zu simulieren. Entsprechend gibt das "Funkgerät" auch Sprache aus (siehe Outputs).

Output

Als Ausgabe der Simulation können folgende Systeme eingesetzt werden:

Anzeige- /Kontrollgeräte Die simulierte Geschwindigkeit und ein simulierter Ölstand (der durch Treffer fallen könnte) können über analoge Anzeigen mit Zeiger ausgegeben werden, die mit dem System rückgekoppelt sind. Ebenfalls kann (bei der Simulation modernerer Fluggeräte) ein Monitor als Display eingebaut und angesprochen werden, über den beispielsweise die Anzeige eines Radarsystems oder einer simulierten Infrarotkamera erfolgen kann. Andere Darstellungen sind möglich.

Sprachausgabe ("Funkgerät") Zur Interaktion mit einem Tower/Flughafen ist bei einer Spracheingabe (siehe oben) auch die zu simulierende Antwort auszugeben.

Sound Über ein Soundsystem/Lautsprecher sind die Fluggeräusche des eigenen Flugzeugs als auch vorbeifliegender Flugzeuge zu simulieren. Auch der Abschuss einer Rakete oder ähnliche Geräusche sollten dargestellt werden. In einer nicht-militärischen Simulation könnte neben den Geräuschen des Flugzeugs auch eine Musik eingespielt werden (z.B. französische Musik beim Flug über Paris o.ä.).

Nebelwerfer Die Simulation von durchflogenen Wolken oder Rauchentwicklung im Cockpit könnte durch Nebelwerfer erfolgen. Dadurch entwickelt sich eine dreidimensionale Darstellung der Simulation noch stärker. Rauch verhindert auch den Blick auf die Projektionsflächen... analog zu realen Wolken.

Bewegliche Flächen innerhalb des Cockpits Über bewegliche, gepolsterte Flächen an den Innenseiten des Cockpits in Höhe des Nutzers kann auf den Nutzer ein Hieb ausgeübt werden. Andere reale Einwirkungsmöglichkeiten auf den Benutzer wären auch simulierbar.

Propeller Um Flugwind zu erzeugen, können vor dem Cockpit Propeller installiert werden. Die Stärke der Propeller kann an die eingestellte Geschwindigkeit gekoppelt werden.

Plattform Eine bewegliche Plattform, auf der sich das Cockpit mit Umbauten befindet, könnte leichte Neigungen des Flugzeugs simulieren. Problematisch ist dies jedoch, da ein Flugzeug z.B. im Extremfall auch auf dem Kopf fliegen könnte. Also ist keine reale Umsetzung mit einer Plattform möglich.

Cave

Der Cave soll aus drei Wänden, einem Boden und einer Decke bestehen. Jede Fläche wird entsprechend bestrahlt. Im Inneren des Caves ist ein Cockpit aufgebaut in Form des dargestellten Fluggeräts mit integrierten, oben beschriebenen I/O-Geräten.

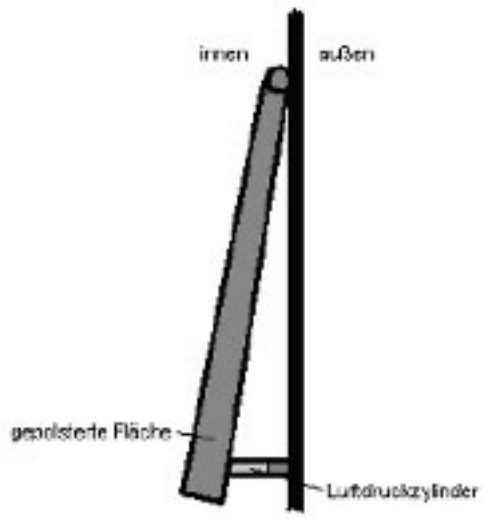


Abbildung C.10: Flugsimulation

C.10 Action-Adventure (Markus)

Vorab - Grundspielart:

Adventure: Denkbar ist in dieser Hinsicht ein Spiel, in dem esgrundsätzlich darum geht, eine Umgebung zu erkunden. Da diese Umsetzung später in der Cave-Umgebung genutzt werden soll, halte ich es für sinnvoll, den Spieler aus der Ich-Perspektive agieren zu lassen. Möglich ist hierbei zum Beispiel die Erkundung einer virtuellen Stadt und deren Sehenswürdigkeiten. Diese müsste aber im Hinblick auf die Grafikengine vielleicht räumlich begrenzt werden. Favorisieren würde ich in dieser Hinsicht eine Umsetzung, die der des Spiels Diablo nachgeahmt ist. Es bietet eine Menge Interaktion des Benutzers, hohen Spielwert, die Möglichkeit, einfache grafische Umsetzung, grosses Potenzial zur Verbesserung und einfache Möglichkeiten der Abwandlung.

Abarbeitung des eigenen Konzepts anhand des Entwurfsrahmens von Christopher:

Spielaufgabe:

Erkundung einer unbekannten, dunklen, mystischen Umgebung, die geprägt ist durch das Vorhandensein einer bösen Macht, deren Ziel es zu sein scheint, die Welt des Spiels, in Chaos und Verderben zu stoßen. Ziel des Spielers ist es, mit dem gewählten Charakter die Pläne der finsteren Seite zu durchkreuzen und den Bewohnern dieser Welt wieder Hoffnung zu geben. Dabei sollte darauf geachtet werden, dass die Hauptaufgabe nicht im “Niedermetzeln“ von Gegnern, sondern im Lösen von Rätsel besteht. Dennoch ist das Vorhanden sein von Gegnern, die nur durch das Austragen von (Zwei-)kämpfen überwunden werden können von Wichtigkeit, da dies dazu beiträgt, dass permanent Spielspannung aufgebaut wird und sich der Spieler nicht durch eine übermässige Anhäufung von Rätseln langweilt. Die Einführung eines Punktesystems, ähnlich dem der Spielvorlage Diablo kann dazu genutzt werden, den gewählten Charakter zu Kampfzwecken zu verbessern oder schwerere Rätsel, die im Spielverlauf auftreten, mit einer verbesserten Magie (kleine Hilfen) zu lösen.

Welttyp:

Als Grafikengine eigenen sich zum Beispiel die Quake-Engine oder die Madness-Engine. Der Grossteil des Spiels findet nämlich in “dunklen, winkligen Korridoren“ statt. Diese müssten eine schaurige und dunkle Stimmung erzeugen, die den Zustand der Umgebungswelt auf psychischer Ebene auf den Spieler übertragen. Dieser greift dann um so beherzter in das Spielgeschehen ein. Rätsel, die in das ganze Spiel eingebettet sind, können in diesen Korridoren sowie in der Oberwelt, die als Basis dient, auftreten. Dabei ist die Oberwelt als eine stark räumlich begrenzte Umgebung zu sehen. Denkbar wäre ein kleines Dorf mit einer natürlichen Begrenzung. Also zum Beispiel einem reißenden Fluss, dessen einzige Überquerungsmöglichkeit (Brücke) zerstört ist, so dass der Spieler in der Oberwelt-Umgebung nur beschränkte Möglichkeiten zur Erkundung hat (andere Möglichkeiten: ein Moor oder eine Insel). Der Spieler kann dabei im Dorf Kontakt aufnehmen zu anderen Bewohnern der Welt um sich Hilfe und Rat im Kampf gegen den übermächtig wirkenden Gegner holen.

Weltbestandteile:

Weltbestandteile sind, wie schon erwähnt, einmal Monster und Waffen, zum anderen aber auch schön gestaltete, realistisch wirkende Umgebungen wie zum Beispiel das Dorf und seine Bewohner. Des weiteren gehören aber auch gut animierte Rätsel zum Spiel wie auch einige Videosequenzen, die dem Spieler das nötige Wissen vermitteln, so dass er dem Spielstrang folgen kann.

Input

Da der Spieler seine Umgebung erkunden will und muss, um die Rätsel zu lösen und die Welt vor der endgültigen Finsternis zu retten, schlage ich vor, dass er auf vielfältige Weise mit der Umgebung in Kontakt treten kann und diese auch darüber beeinflussen kann. Denkbar für die Fortbewegung des Spielers ist zum Beispiel eine Mischung aus den uns schon bekannten selbstgebastelten Schuhsohlen, die mit Sensoren versehen sind und einer im Cave begrenzt im Boden verarbeiteten Sensorik, die eine feinere “Abtastung“ erlaubt. Die Lösung der oben erwähnten Rätsel sollte so gestaltet sein, dass der Spieler dies über einen Datenhandschuh tun kann. Mit diesem kann er Objekte greifen, verschieben etc. Zudem sollte es ihm mit diesem Handschuh möglich sein, auch mit den Dorfbewohnern in Kontakt zu treten. Hierbei kann er den Datenhandschuh zur Steuerung des Kontextmenüs nutzen. Dies enthält neben den persönlichen Fähigkeiten der Dorfbewohner auch die Möglichkeit zu einem ungezwungenen Gespräch, welches wiederum bei korrekter Anwendung nützliche Tipps zum Spiel geben kann. Weiterhin wäre es möglich, den Spieler mit einem Messgerät für die Vitalwerte auszustatten. So ist es möglich, den Druck auf den Spieler zu erhöhen, wenn festgestellt wird, dass sich dieser in der Umgebung langweilt, weil er “stark“ genug ist für das Auftreten wiederstandsfähigerer Monster oder Rätsel, die mehr Geschick und Verstand benötigen.

Output

Grundsätzlich ist hier alles erwünscht, was zu einer noch düsteren und mystischen Umgebung beiträgt als es die Grafik an sich schon tun sollte. Ansonsten sind hier Möglichkeiten aufgelistet, die dem Spieler einen noch realistischeren Eindruck seiner Umgebung und eine tiefere Empfindung seines Gesundheitszustandes zulassen. Als erste Möglichkeit, die Umgebung düsterer zu gestalten sei nun eine “Nebeldüse“ erwähnt, wie sie zum Beispiel in Discotheken verwendet wird. Dabei könnte der Cave softwaregesteuert, zum Beispiel beim Durchqueren eines Moores, im Bodenbereich mit Nebel gefüllt werden, der sich nur langsam verzieht. Eine weitere Möglichkeit, Wettergegebenheiten oder ähnliches zu simulieren, wäre ein Gebläse, das sowohl warme wie auch kalte Luftströme verbreiten kann. Dies käme zum Einsatz, um

zum Beispiel beim Betreten einer neuen Höhle dem Spieler zusätzlich zuzusetzen und ihm mit kalter Luft eine Gänsehaut zu entlocken. (Das Gebläse ist aber nur bedingt einsetzbar, und zwar dann, wenn die Projektionsflächen durch Materialien ersetzt werden, die nicht so sehr von Luftzirkulationen beeinflusst werden.) Grundsätzlich sollte die Szene auch mit einer zum Rest passenden Musik untermalt werden. Dies bedeutet, dass ständig die Grundstimmung mit einer Hintergrundmelodie aufrechterhalten wird. Treten Sonderfälle (Rätsel, Endgegner) auf, sollte diese Hintergrundmusik gegen entsprechend passende und anregende Musik ausgetauscht werden. Ähnliches gilt für die Stimmen der Dorfbewohner und Monster. Es sollte dafür gesorgt werden, dass diese als entweder besonders vertrauenserweckend oder besonders abstossend beim Zuhörer ankommen. Selbstverständlich sollte dabei sein, dass es sich beim Sound um Surround Sound handelt. Um eine realistischere Empfindung der Umwelt zu erreichen, kann zudem durch eine Plattform dafür gesorgt werden, dass der Spieler den Eindruck erlangt, in einem Gang bergauf oder bergab zu laufen oder gerade einen nahegelegenen Hügel zu erklimmen (begrenzte Steigung/Gefälle). Dies wirkt sich auch auf den nächsten Punkt aus. Eine nette Erweiterung stellt dabei meiner Meinung nach eine Art Straffungsgürtel im Brust - oder Bauchbereich dar. Mit diesem ist es möglich über eine zunehmende Atemnot bei lange anhaltendem Laufen oder beim Erklimmen von Hügeln und Bergen zu simulieren. Wenn aber schon die Atemnot simuliert wird, dann darf der Extremfall und ein paar andere Abstufungen nicht fehlen. Gemeint sind einmal der Tod des Charakters, sowie eine sinnvolle Bestrafung beim Versagen in einer Prüfung oder das Zuziehen von Verletzungen im Kampf mit den lauernden Monstern. Dies könnte realisiert werden über Anschlüsse an der freien Hand. Dort könnte dem Spieler in abgeschwächter Form über Elektroschocks Schmerz zugefügt werden. Zudem könnten drei mit Flüssigkeiten gefüllte Zylinder den Innenraum des Caves abrunden. Einen mit roter Flüssigkeit gefüllten für die Lebensenergie, einen zweiten für die spirituelle Energie in blau und den letzten mit fast klarer zur Anzeige des Erschöpfungsgrades. Die Flüssigkeiten werden dabei mittels handelsüblicher Pumpen abgesaugt oder wieder aufgefüllt (entsprechend dem Zustand des Charakters).

Grafik

Der Spieler sollte die Umgebung aus der Sicht des Helden wahrnehmen und erforschen. Dies versetzt ihn vollends in die Lage der Person, die er im Spiel verkörpert und lässt ihn alles aus einer für ihn an sich gewohntenPerspektive erleben.

Welt

Cave, zusammengesetzt aus vier(!) Seitenflächen und einer Deckenfläche. Die Bodenfläche ist nicht zur Projektion geeignet, könnte aber entsprechend des Spieltyps hergerichtet werden. Ansonsten alles wie schon im Text beschrieben.

C.11 Picasso Cave (Benjamin)

Die Idee des Picasso Caves ist es, sich nach Belieben mit Pinsel, Farben und Sprühdose an den Wänden des Caves virtuell auszudrücken. Angedacht sind dabei alle Projektionsflächen außer der unteren (am liebsten wären mir aber alle!). Die zu bemalende Fläche kann dabei aus einem Archiv ausgewählt werden, denkbar wäre die Chinesische sowie Berliner Mauer, Züge der deutschen Bahn AG sowie Leinwand, Glas, Holz, Porzellan, Seide oder Pappe. Der “Künstler“ kann sich für Öl-, Aquarell-, Acryl- oder Ölfarben entscheiden und außerdem aus einem Sortiment verschiedenster Pinselstärken wählen sowie diverser Sprühkopfaufsätze für das Graffiti.

Auch für die musikalische Untermalung ist gesorgt, neben der Klassik und dem Jazz kann sich der Benutzer auch bei aktueller Musik entspannen (HipHop, Charts...). Für Liebhaber der Natur können zu-

dem auch Soundkulissen wie Meeresrauschen, Vogelgezwitscher oder säuselnder Wind erzeugt werden, schließlich ist für manche Künstler die Entspannung das A und O bei ihrer Arbeit. Falls das eben gemalte nicht gefällt, kann die entsprechende Cave Wand wieder gelöscht werden oder eben der letzte getätigte Pinselstrich. Ist das Kunstwerk vollendet oder möchte der Künstler eine Pause einlegen, um ein anderes Mal weiter zu arbeiten, besteht die Möglichkeit, das Gebilde zu sichern und auch in ausgedruckter sowie digitaler Form zu erhalten. So kann er sich das Kunstwerk später Zuhause am PC als 3D-Panorama anschauen und sich noch eventuelle Verbesserungen überlegen. In verschiedenen Benutzerprofilen werden die individuellen Einstellungen der einzelnen Künstler zusammengefaßt, so daß nach dem Einloggen ins System, sich dieses wieder auf den jeweiligen Benutzer einstellt.

Vorteile:

- Eventuelle Entstehung einer neuen Kunstform, bei welcher keine Kosten für Farbe und Malfläche mehr entstehen und somit auch die Belastung für die Umwelt geringer ist
- pubertierende Jugendliche haben Beschäftigung und
- das Stadtbild bleibt erhalten

Nachteile:

- Zum vollständigen Kunsterlebnis gehört es für die meisten Maler auch, den Geruch der Farben wahrzunehmen, es wäre zu überlegen, ob man dies nicht auch realisieren könnte, möglicherweise mit realem Geruch oder sowie Volker vorgeschlagen hatte, mit dem Vorgaukeln von Geruch durch intensive Fokussierung auf die anderen Sinne. Volker erzählte ja das Beispiel, wo einer Testperson ein flackerndes Feuer und der entsprechende Sound vorgesetzt wurde und wenn ich mich recht entsinne auch noch die Temperatur angehoben wurde und die Person dann meinte, den Qualm des Feuers riechen zu können. Also im Groben das Prinzip der Synästhesie wobei z.B. das Sehen der Farbe Orange unwillkürlich mit einem bestimmten Geruch, Klang in Verbindung gebracht wird.
- Für viele vielleicht eine Freude, für andere ein Nachteil: das Klecksen mit den Farben entfällt.
- Ein Hauptbeweggrund für die jungen Graffiti Künstler ist der Reiz des Verbotenen, der durch dieses System nicht mehr gegeben ist.

Christophers Punkte:

Content

Die Objekte die in der “Welt“ enthalten sind, bestehen aus den einzelnen Malwerkzeugen sowie der Malfläche. Die “Spiel-Aufgabe“ besteht darin sich zu entspannen, kreativ zu sein und Spaß zu haben.

Input

Zuerst hatte ich mir nachfolgende Realisierung für die Eingabe überlegt, bei der Vorstellung am Freitag wurde aber klar, das es vermutlich doch einfacher wäre, mit einem Datenhandschuh zu arbeiten, mit dem man durch bestimmte Fingerstellungen und -bewegungen konkrete Befehle umsetzen kann. Dabei hatte Björn auch die Idee, daß die Auswahl der Musik im Cave davon abhängig sein könnte, “wie“ der Künstler malt. Das heißt, wenn er langsam und behutsam den Pinsel benutzt, entsprechend ruhige und

entspannende Musik eingespielt wird und andersherum. Der Maler könnte also mittels seiner Maltechnik bestimmen, welche Musik gespielt wird.

Meine vorherige Alternatividee: Die Projektionsflächen beinhalten transparente Druckflächen, welche die Pinselaktivitäten wahrnimmt und so an dieser Stelle die Farbe gezeichnet wird. Die Farbe kann über ein am Pinselwerkzeug befestigtes Rädchen eingestellt werden. Auf der Sprühdose befindet sich ein Laser, der aktiviert wird, wenn der “Sprühknopf“ betätigt wird. Die Farben werden ebenso wie beim Pinsel über ein Rädchen ausgewählt. Aufgefangen wird dieser Laser von Photozellen befindlich auf der Projektionsfläche. Die Art der Farbe und der Malfläche sowie die Art der Musik kann über ein externes Eingabegerät eingegeben, welches den höchsten Designansprüchen gerecht wird.

Output

Eine Rückkopplung der virtuellen Welt zum Benutzer findet dort statt, wo ihm die Beschaffenheit der Malfläche suggeriert wird. Dies geschieht nun aber nicht über ein Rütteln am Pinsel oder ähnliches, sondern geschieht direkt über die Grafik. Außerdem wäre über eine Temperatureinstellung nachzudenken, was zum einen das Szenario unterstützen kann und zum anderen auf die Wünsche des Malers eingeht sowie der Geruch von Farbe (siehe oben).

Grafik

Bei bestimmten Matrialien wie Holz muß der Benutzer mehr Farbe auftragen als bei Leinwand, um eine ähnliche Farbdichte zu erhalten. Die dargestellte Grafik ist recht einfach, da statisch. Zu überlegen wäre, einen bewegungsfreien Teil zu nehmen in Zusammenspiel mit einem bewegten (ein kleiner Fluß zum Beispiel, der im Hintergrund fließt).

Welt

Die Welt ist wie bereits beschrieben recht einfach gehalten. Das ist deshalb so geplant, damit der Künstler so wenig wie möglich bei seiner Arbeit gestört wird. Wenn es erwünscht ist, einen kleinen rauschenden Bach zu haben, der auch visualisiert werden soll, wäre das aber auch im Bereich des möglichen. Denkbar wäre sicher auch eine gemeinsame Zusammenarbeit zweier Künstler. Dabei könnte sich der eine in einem Cave in Los Angeles und der andere in einem Cave in Weyhe befinden.

Die beiden oder mehreren Künstler (womöglich in mehr als zwei Caves) sehen in ihrem eigenen Cave genau das, was die anderen neu dazugemalt haben.

C.12 Jump and Run (Volker)

Allgemeines

Meiner Meinung nach sollten wir unser Ziel aus vielerlei Gründen vorerst nicht zu hoch stecken. Nur wer mit großen Schritten seinem Ziel schnell näher kommt, ist und bleibt auch motiviert. Außerdem ergeben sich beim Entwickeln sicher noch viele tolle Ideen, die man dann immer noch umsetzen kann. Ich fände es schade, wenn am Ende viele Ideen sich als nicht realisierbar herausstellen. Ich sehe auch noch genug Punkte, wo wir noch viel Erfahrungen sammeln müssen; Schnittstellen zum Benutzer zum Beispiel. In gewissen Dingen sind wir auch machtlos. Noch steht ja gar nicht fest, wieviele Projektoren wir am Ende zur Verfügung haben werden. Auch an solche Dinge sollten wir denken. Am Ende eine Projektionsfläche mehr in irgendeiner Render-Engine einzustellen, ist ein kleineres Übel, als eine wichtige nicht zu haben.

Ich habe z.B. eine Idee, wie man zumindestens einen fehlenden Projektor doch recht schön ersetzen kann: Bespannt man auch das Dach des Caves mit transparenter Folie und bestrahlt es mit farbig passendem Licht (blau und weiß zum Beispiel), hat man so etwas wie einen Himmel.

Meine Idee

Ich stelle mir ein (gewaltfreies) Spiel im Cave vor, wo man über eine Trittmatte gesteuert, durch eine 3D-Welt laufen muß. Über die Trittmatte sollten Vorwärts-, Rückwärts-, Links-, und Rechtsbewegung sowie Hüpfen realisiert sein. Bei der Matte könnte ich mir vorstellen, daß wir uns eine einfache DDR-Matte kaufen und sie zerlegen. Es gibt im Netz Dokumentation vom PlayStation-Stecker und auch Dokumentation, wie man diese Matte zum Beispiel an den Parallel-Port angeschlossen bekommt. Damit sollte es uns leicht möglich sein, die Matte auszulesen. Die Herausforderung ist hier dann nicht eine SPS-Steuerung, sondern die Erweiterung der Eingabemöglichkeiten im verwendeten Renderer (Madness). Ich hätte große Lust, irgendeinen alten Jump-And-Run-Klassiker (ich denke gerade an Giana Sisters) in abgewandelter Form auf die Cave zu übertragen. Das Wort Cave birgt dunkles, welches es mit einem hellem Spiel, das jung und alt zu begeistern weiss, zu durchbrechen gilt.

Abbildungsverzeichnis