Diplomarbeit

Entwicklung einer interaktiven Storytelling AR-/VR-Spiel- und Lernumgebung

durchgeführt am
Zentrum für Graphische Datenverarbeitung e.V.
Darmstadt

von

Christian Manz
(660 767)

Fachhochschule Gießen-Friedberg
Bereich Friedberg
Fachbereiche IEM, MND, MNI
Studiengang Medieninformatik

Sommersemester 2005

Referentin:
Prof. Dr.-Ing. Dipl.-Math. Monika Lutz

Korreferent:
Dipl.-Ing. (FH) Oliver Schneider

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass die vorliegende Diplomarbeit ohne unzulässige Hilfe und ausschließlich unter Verwendung der angegebenen Literatur angefertigt wurde.

Friedberg, den 11. April 2005

_____

Christian Manz

Danksagung

Ich möchte mich bedanken bei

- Prof. Dr.-Ing. Dipl.-Math. Monika Lutz für ihr großes Engagement und die ausgezeichnete Betreuung,

- Dipl.-Ing. (FH) Oliver Schneider für seine Unterstützung und die wertvollen Tipps bei der Entwicklung der Anwendung sowie für seine Bereitschaft, die Arbeit als Korreferent zu betreuen,

- den Mitarbeitern der Abteilung Digital Storytelling des Zentrums für Graphische Datenverarbeitung e.V. in Darmstadt für das sehr angenehme Arbeitsklima und die Beantwortung vieler Fragen,

- StD Richard Krings für das Korrekturlesen der Diplomarbeit,

- meiner Familie für die Unterstützung während des Studiums, das Korrekturlesen und die konstruktive Kritik.

Berglicht, 11. April 2005

Christian Manz

Inhaltsverzeichnis

1 Einleitung

1.1 Zielsetzung der Arbeit

Ziel dieser Diplomarbeit ist die Entwicklung einer interaktiven, nichtlinearen Storytelling AR[1]-/VR[2]-Spiel- und Lernumgebung. Die Umsetzung der Anwendung erfolgt auf Basis der Opensource-Software Blender, welche sowohl den benötigten 3D-Modellierer als auch die erforderliche Game-Engine beinhaltet. Die Diplomarbeit wird im Rahmen des Projektes EduTeCH[3] durchgeführt. „EduTeCH verbindet Interactive Storytelling, Augmented Reality und Location Based Services zu einer unterhaltsamen Cultural Heritage Anwendung. Basierend auf bestehenden INI-GraphicsNet[4]-Projekten wird ein System entwickelt, welches den Besucher von historischen Orten im asiatischen Raum auf unterhaltsame Weise geschichtliche Informationen vermittelt [Goebel03]“. Das Projekt wird am Zentrum für Graphische Datenverarbeitung e.V. in Darmstadt durchgeführt. Projektpartner sind NEMETech in Seoul, CAMTech in Singapore, Fraunhofer IGD und GIStec GmbH in Darmstadt [Herzig03].

Mit dieser AR-Anwendung wird es möglich sein, sich in der koreanischen Tempelanlage Gyeongbok Palace in Seoul frei zu bewegen und zusätzliche lokalspezifische Informationen auf einem Headmounted-Display abzurufen (siehe Abbildung 1). Die Blickrichtung des Nutzers wird über einen Tracker bestimmt. Der zu überblendende Bildausschnitt wird mit Hilfe des zurück gegebenen Vektors berechnet.

---

[1] Augmented Reality (Definition siehe Kapitel 2.2.2)

[2] Virtual Reality (Definition siehe Kapitel 2.2.1)

[3] EduTeCH (**Edu**tainment **Te**chnologies for **C**ultural **H**eritage in Asia)

[4] INI-GraphicsNet (**I**nternational **N**etwork of **I**nstitutions for Advanced Education, Training and R&D in Computer Graphics Technology, Systems and Applications)

Zielgruppe der Anwendung werden sowohl Einheimische als auch Touristen sein, bei denen das Interesse an der Geschichte des Palastes sowie dem koreanischen Alphabet geweckt und diesbezüglich grundlegende Informationen vermittelt werden sollen. Dies geschieht mit Hilfe eines virtuellen Charakters, der auf vorher bestimmten Bühnen auftaucht, dem Besucher Informationen gibt und ihn mit Fragen in ein Lernspiel einbindet, durch welches dann die Nutzer mit einer Auseinandersetzung mit den Buchstaben des koreanischen Alphabetes konfrontiert werden. Der motivierende Charakter kann dabei sowohl die Funktion eines Helfers wie auch die Funktion eines Gegenspielers einnehmen. Dies muss vom Spieler erkannt werden, um die richtigen Antworten geben zu können.

Abb. 1: Skizze der Tempelanlage Gyeongbok Palace in Seoul

Der Spielverlauf wird mit Hilfe einer Story-Engine überwacht und gesteuert. Dadurch bleibt die Handlung im Spiel sehr variabel und individuell anpassbar, da die Story-Engine den weiteren Verlauf der Geschichte ständig

neu berechnet und dabei das schon Geschehene mit einbezieht. Dadurch ist es möglich, dass der Nutzer auf verschiedenen und auch immer neuen Wegen letztlich doch zum gleichen Ergebnis kommt, da die Informationen jeweils von der Story-Engine in anderen Reihenfolgen oder wechselnden Orten präsentiert werden.

Das Spiel wird nicht nur als AR-Anwendung umgesetzt. Zusätzlich wird das Geschehen noch als VR-Spiel realisiert. Somit wird es jedem ermöglicht, den Rundgang durch die Tempelanlage am heimischen PC zu erleben. Um dies zu erreichen, muss die gesamte Tempelanlage originalgetreu modelliert und mit Texturen versehen werden, um so einen realistischen Eindruck des Komplexes zu erhalten.

Um die Installation des Spieles so einfach wie möglich zu gestalten, wird es für das verwendete Betriebssystem ein Installationsprogramm geben, das automatisch alle benötigten Programme installiert, so dass die Anwendung sofort spielbar ist.

1.2 Gliederung der Arbeit

Das nun folgende zweite Kapitel wird die für das Verständnis der Arbeit notwendigen Grundlagen vermitteln und dabei Begriffe wie Virtual- und Augmented-Reality sowie interactive und non-linear Storytelling definieren. Zusätzlich zur Erklärung der Begriffe wird die für dieses Vorhaben erforderliche Hardware vorgestellt und erläutert. Abschließend wird noch auf die Konzeption und Anwendung der für die Scripte verwendeten Programmiersprache Python eingegangen.

Das dritte Kapitel behandelt dann den Bereich der Virtual Reality Anwendung im Projekt EduTeCH. Zunächst wird auf die Zielsetzung des Spiels eingegangen, um anschließend die einzelnen Schritte hin zu einer funktionsfähigen Virtual Reality Anwendung aufzeigen und erklären zu können. Die Erörterung umfasst sowohl den Modellierungsprozess (von

Gebäuden, Landschaft, Charakter etc.) als auch die Programmierung der Game-Logic (z.B. Spielersteuerung, Ablauf des Spiels, Integration externer Komponenten), die das eigentliche Herzstück eines jeden Spiels ist.

Auf dem gerade Beschriebenen baut dann das vierte Kapitel auf, in dem die Augmented Reality Anwendung vorgestellt wird. Notwendige Abänderungen der VR-Anwendung werden beschrieben. Die Einbindung der für die AR-Anwendung erforderlichen Hardware (AR-Glasses, Tracker und GPS) wird aufgezeigt.

Das fünfte Kapitel befasst sich mit der Erstellung von Installationsskripten (Ebuild) für das verwendete Linux-Betriebssystem (Gentoo), durch das es dann möglich sein wird, das Spiel ohne Aufwand zu installieren. Dabei werden sowohl die allgemeinen Regeln für die Erstellung wie auch die konkrete Umsetzung für das vorliegende Projekt beschrieben.

Den Abschluss dieser Diplomarbeit bilden das sechste Kapitel mit dem Fazit, sowie die Kapitel sieben und acht (Literaturverzeichnis bzw. Anhänge).

Um Begriffe aus der Programmierung wie Schlüsselworte aus Programmiersprachen, Programmfragmente, Klassen-, Objekt- oder Methodennamen besser vom Fließtext unterscheiden zu können, werden diese zur Verdeutlichung durch die Schriftart `Courier New` typographisch hervorgehoben.

2. Grundlagen

2.1 Einleitung

Gegenstand dieser Diplomarbeit ist die Entwicklung einer interaktiven Spiel- und Lernumgebung, die sowohl in einem VR-System als auch in einem AR-System verwendet werden kann. Um das Verständnis der in dieser Diplomarbeit behandelten Thematik zu erleichtern, soll dieses Kapitel dazu dienen, einen grundlegenden Überblick über die dafür notwendigen Begriffe und Techniken zu vermitteln.

Aus diesem Grund findet zunächst eine Erläuterung der Begriffe Virtual Reality und Augmented Reality statt. Anschließend wird auf die hier verwendete Form des digitalen Storytellings eingegangen. Des Weiteren wird die für diese Technologien notwendige Hardware vorgestellt und deren konkrete Anwendung in dieser Arbeit geschildert. Zum Abschluss wird dann das Konzept der für die Scripte verwendeten Programmiersprache Python erläutert.

2.2 Virtual und Augmented Reality

2.2.1 Virtual Reality

Bevor 1988 der Begriff der Virtual Reality (VR, virtuelle Realität) entstand, „wurde die Idee einer sichtbaren, interaktiven und computergenerierten Scheinwelt ‚Cyberspace' genannt" [Ruegge99]. Myron Krueger nannte es „Artificial Reality" [Krueger83], was sich jedoch nicht durchsetzen konnte. Der letztlich bis heute gültige Name der Virtual Reality wurde von Jaron Lanier kreiert und damals wie folgt umschrieben: „‚Virtuell' heißt, dass etwas nur als elektronisches Bild existiert, aber sonst keine konkrete Gegenständlichkeit hat. Es ist, als wäre es da, aber es ist nicht" [Heilbrun91]. Die Virtual Reality stellt heute „eine neue Generation von Mensch-Maschine-

Schnittstellen, gleichzeitig aber auch ein völlig neues Medium, welches die zwischenmenschliche Kommunikation unter Einbeziehung aller Sinne in eine Scheinwelt verlagern kann“ [Henning97] da.

Es handelt sich bei Virtual Reality also um ein System, das dem Nutzer den Eindruck vermitteln soll, dass das dargestellte Szenario der realen Umwelt entspricht (siehe Abb. 2). Um dies zu erreichen werden verschiedene Methoden im Bezug auf die räumliche Darstellung der verschiedenen Objekte, das natürliche Empfinden von Tönen sowie diverse Möglichkeiten zur Interaktion mit dem System genutzt. Da es sich aber bislang nur um ein digitales Abbild der Welt im Speicher des Rechners handelt, kommt dem Ausgabemedium eine hohe Bedeutung zu. Neben der visuellen Ausgabe stellt die Interaktion einen wichtigen Bestandteil der Virtual Reality dar. So soll es dem Nutzer möglich sein, sich natürlich in der Welt zu bewegen und mit ihr zu interagieren (z.B. einen Gegenstand hochheben oder drehen). Aufgrund des gewählten Ausgabemediums können zwei grundlegende Formen von Virtual Reality unterschieden werden.

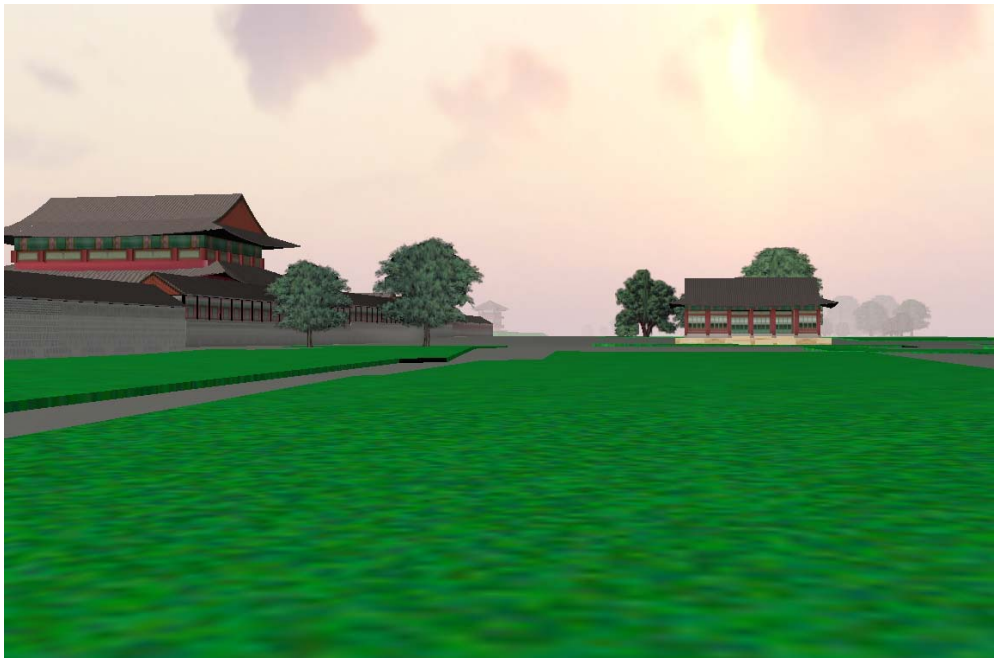

Abb. 2: Screenshot aus der VR-Anwendung dieser Diplomarbeit

Zum einen gibt es die Desktop VR-Systeme. Sie werden auf einem normalen PC-Desktop ausgegeben und stellen somit keine speziellen Hardwareanforderungen. Aus diesem Grund sind diese Systeme die am weitesten verbreitete Form der Virtual Reality. Da hierbei der normale Bildschirm als Ausgabemedium verwendet wird, wird vom „Window on World“ [Ruegge99] gesprochen. Der Benutzer schaut wie durch ein Fenster in die virtuelle Welt hinein. Die Interaktionen mit Desktop VR-Systemen finden meist über die Tastatur, die Maus oder einen Joystick statt.

Zum anderen: Die immersiven VR-Systeme lassen den Nutzer völlig in die virtuelle Welt eintauchen. Dies ist bei der Desktop VR nur begrenzt möglich, z.B. bei 3D-Computerspielen, wobei die Immersion hierbei nur zum Teil erreicht wird. Um die völlige Immersion zu erreichen, wird die Verwendung von spezieller Hardware nötig, wobei verschiedene Ansätze existieren. Zum einen werden Head-Mounted-Displays (HMD) verwendet, die direkt auf dem Kopf getragen werden und für jedes Auge einen eigenen Bildschirm haben. Zum anderen sind zusätzlich Kopfhörer integriert, die sicherstellen, dass ausschließlich Geräusche der virtuellen Umgebung zum Nutzer vordringen können. Ergänzt wird die Hardware durch einen Tracker, der die Kopfposition überwacht und dadurch den entsprechenden Bildausschnitt darstellt. Zwar findet hier eine Verschmelzung mit der virtuellen Welt statt, allerdings bleibt der Nachteil des Gewichtes, das der Nutzer auf seinem Kopf tragen muss. Außerdem besteht durch die notwendige Verkabelung eine räumliche Einschränkung.

Eine weitere Möglichkeit des optimalen VR-Systems stellt der CAVE dar (siehe Abb. 3). Hierbei handelt es sich um einen Raum, dessen Wände die Funktion von Monitoren haben und somit den Nachteil des HMD beheben. Das Problem der räumlichen Beschränktheit besteht jedoch auch bei diesem Typ weiter.

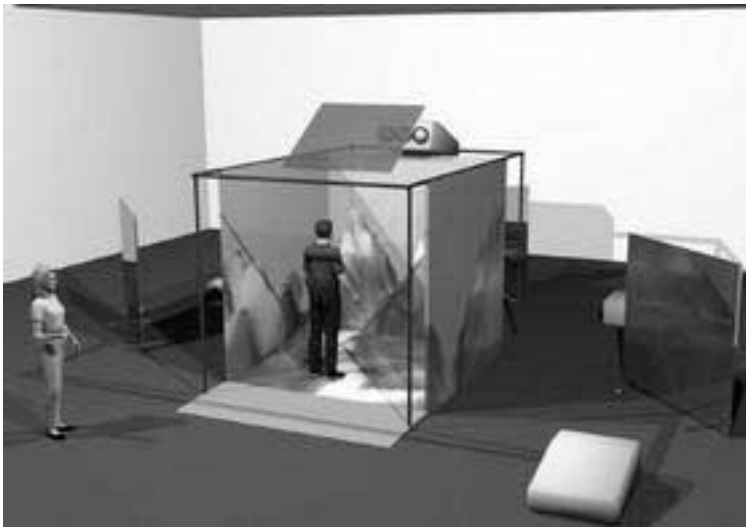

Abb. 3: Darstellung eines Caves [Anthes02]

Aus diesem Grund wurde dann die Cybersphere entwickelt. Mit ihr fällt auch die Beschränktheit des Raumes und man kann sich frei in jede Richtung bewegen. Allerdings bleibt die Benutzung einer derart komplizierten und aufwendigen Technik (siehe Abb. 4) wohl einigen wenigen Menschen vorbehalten.

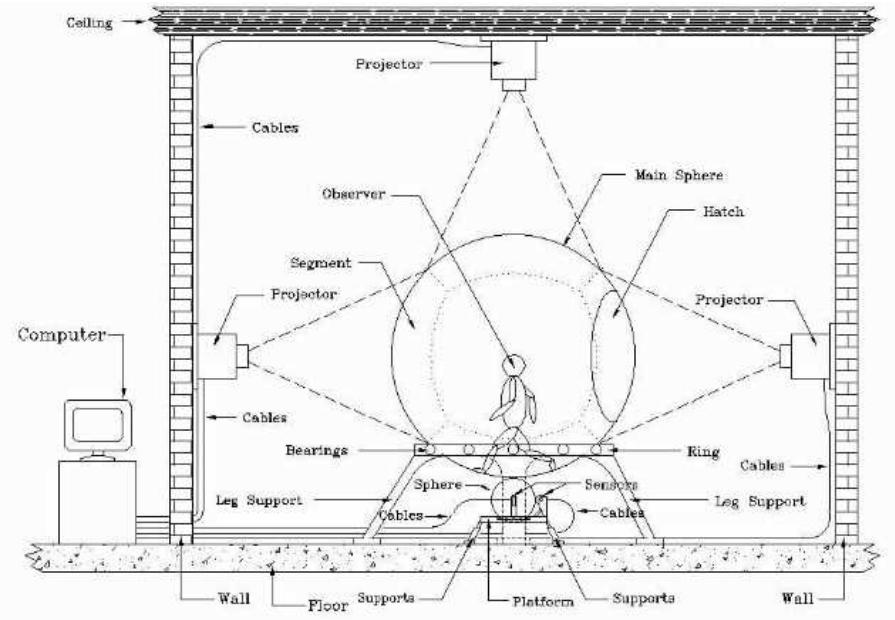

Abb. 4: Zeichnung zum Aufbau einer Cybersphere [Anthes02]

Die Interaktionen in diesen immersiven VR-Systemen finden nicht mehr über die klassischen Eingabemedien statt, sondern über neuartige Entwicklungen wie zum Beispiel Datenhandschuhe oder auch Daten Anzüge.

Die in dieser Arbeit erstellte VR-Anwendung wird in erster Linie auf Desktop VR-Systemen zum Einsatz kommen, da die Technik für die immersiven VR-Systeme wie den Cave oder die Cybersphere einfach zu teuer ist. Lediglich die Nutzung in Kombination mit einem Head-Mounted Display wäre denkbar, da hier die Kosten überschaubar bleiben. Es ist jedoch im Prinzip möglich, die VR-Anwendung so anzupassen, dass sie in jedem der hier vorgestellten Systeme zum Einsatz kommen könnte.

2.2.2 Augmented Reality

Im Gegensatz zur Virtual Reality, die versucht eine künstliche Welt zu erschaffen und den Nutzer völlig damit verschmelzen zu lassen, wird bei der Augmented Reality (AR) die reale Umgebung des Nutzers mit einbezogen und diese dann um virtuelle Objekte ergänzt. „Der Mensch wird also in seiner gewohnten Umgebung belassen, ihm werden in Relation zu dieser Umgebung sonst nicht unmittelbar wahrnehmbare Informationen computergeneriert und zumeist visuell als Ergänzung zur Welt angeboten“ [Ruegge99]. Dabei muss es sich nicht zwangsläufig um Gegenstände handeln, es können auch Informationen wie beispielsweise Erklärungen, technische Zeichnungen o.Ä. dazu dienen, das reale Bild zu erweitern. Die Augmented Reality stellt somit die Mischung aus realer Umgebung und einer virtuellen Szene dar, die durch den Computer das reale Bild überlagert. Deshalb erweitert Augmented Reality lediglich die Realität, anstatt sie vollständig zu ersetzen [Azuma97].

Genau dieser Punkt stellt auch die Schwierigkeit bei Augmented Reality Systemen dar. Die Überlagerung der virtuellen Objekte mit der Realität muss sehr genau geschehen und darf auch bei wechselnder Position des

Nutzers nicht ungenau werden. Die Objekte müssen außerdem dort erscheinen, wo sie vom Betrachter erwartet werden, d.h. eine Kiste darf weder in der Luft schweben, noch im Boden versinken. Sollte dies dennoch der Fall sein, so geht der realistische Eindruck der gesamten Szene verloren. Im Idealfall der optimalen Umsetzung sollte der Anwender die virtuellen nicht von den realen Objekten unterscheiden können. Da die in dieser Arbeit entwickelte AR-Anwendung die reale Bausubstanz nutzt und lediglich durch virtuelle Charaktere ergänzt, ist die Ausrichtung auf Grund der geringeren Anzahl an Objekten nicht ganz so schwierig. Allerdings muss dabei trotzdem sehr genau gearbeitet werden, da es bei einer Person eher auffällt, wenn ein Schuh leicht in den Boden eintaucht als bei einem großen Gebäude.

Nach den Definitionen einiger Wissenschaftler ist das Vorhandensein eines AR-Systems fest mit der Nutzung eines Head-Mounted Displays verknüpft, was jedoch die Augmented Reality stark an bestimmte Technologien bindet. Aus diesem Grund definierte Robert T. Azuma die folgenden Eigenschaften eines AR-Systems:

- Reale und virtuelle Informationen werden von dem System miteinander verknüpft.
- Interaktivität mit dem System ist in Echtzeit möglich.
- Es ist eine korrekte Zuordnung zwischen den Objekten der realen und virtuellen Welt möglich.

Durch diese Definition sind auch andere Technologien neben den Head-Mounted Displays erlaubt, wenn sie die grundlegenden Eigenschaften der Augmented Reality besitzen [Azuma97]. Dies schließt aber Filme mit integrierten 3D-Modellen aus, da ihnen die laut Definition geforderte Interaktivität fehlt und sie somit nicht als Augmented Reality Anwendungen bezeichnet werden können.

Bei der Erstellung von Modellen für eine Augmented Reality Anwendung, muss besonderes Augenmerk auf die Detailtreue der Modelle gerichtet werden, denn der spätere Anwender hat den direkten Vergleich zwischen

Realität und virtuellem Objekt. Dadurch wird die Modellierung wesentlich aufwendiger als bei Virtual Reality Anwendungen, bei denen Vergleichsmöglichkeiten nicht gegeben sind. Ein weiteres Problem, das den Eindruck der nahtlosen Integration beeinflusst ist die exakte Positionierung von virtuellen Objekten in der realen Umwelt. Da das Auge selbst kleinste Differenzen bemerkt, sind solche Unstimmigkeiten unter allen Umständen zu vermeiden.

All diese Punkte sind bei der Entwicklung einer AR-Anwendung zu beachten. Es ist allerdings bis heute noch nicht gelungen, alle Anforderungen in vollem Umfang zu erfüllen. Dabei stellt die Positionierung der Objekte das wohl größte Problem dar, weil dabei vor allem der komplexe menschliche Bewegungsablauf berücksichtigt werden muss. Es ist somit klar, dass auch diese Diplomarbeit keine hundertprozentige Perfektion erreichen kann. Allerdings soll versucht werden, diesem Ziel so nahe wie möglich zu kommen. Erleichtert wird dies durch die Tatsache, dass die Augmented Reality Anwendung erst einmal nur die virtuellen Charaktere beinhaltet. Bei einer exakten Modellierung, die jedoch nur erreicht werden kann, wenn die exakten Gebäudedaten vorliegen, kommt zudem noch der Vorteil der Kollisionserkennung in der Game-Engine zum Tragen, die sowohl das Eintauchen wie auch das In-der-Luft-Schweben verhindert und somit eine sehr genaue Positionierung der Objekte ermöglicht.

Die Augmented Reality wird heute in den unterschiedlichsten Bereichen eingesetzt. Dazu zählen unter anderem die Medizin, verschiedene Bereiche der Industrie, Unterhaltungselektronik (Computerspiele) und militärische Interessen (Navigation, Zielerfassung) (siehe Abb. 5, 6 und 7). Zusätzlich zu diesen schon fast klassischen Einsatzgebieten der Augmented Reality entstehen neuerdings auch Anwendungen für den Tourismus, um historische Stätten so erleben zu können, wie sie früher einmal waren. In diesem Einsatzbereich ist auch die hier entwickelte Anwendung einzuordnen.

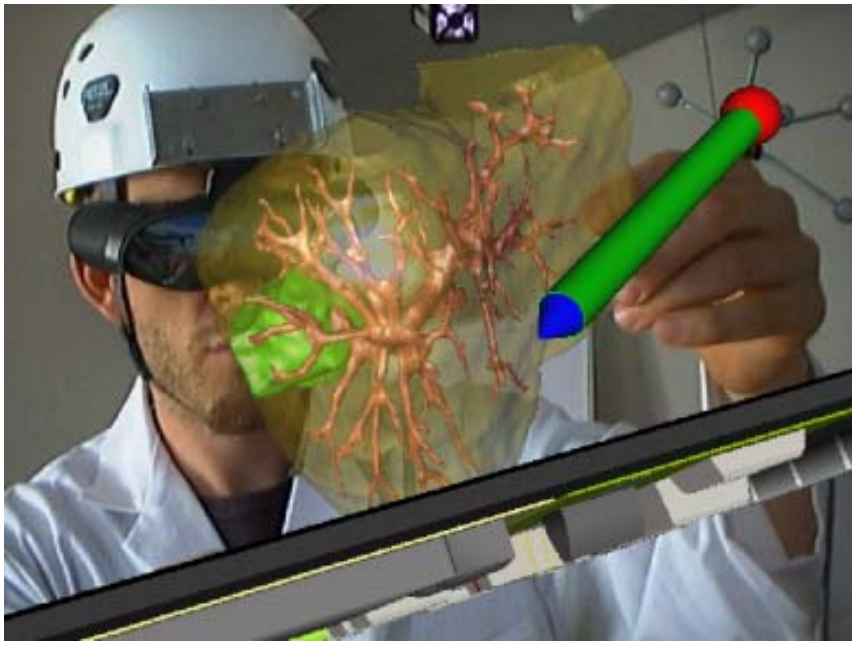

Abb. 5: Medizinische Augmented Reality Anwendung 1 [Sorantin04]

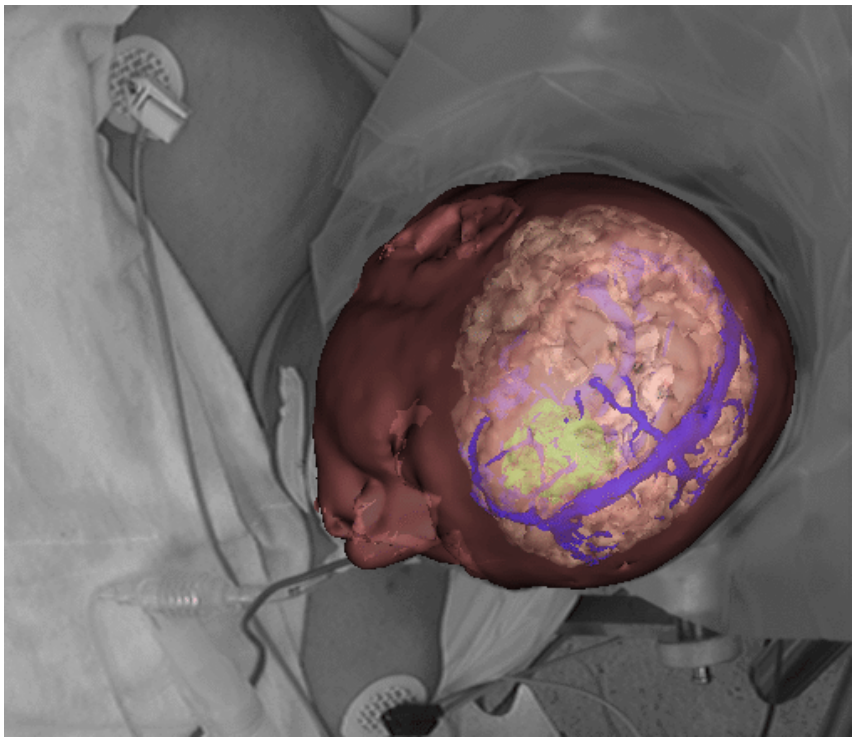

Abb. 6: Medizinische Augmented Reality Anwendung 2 [Brown]

Abb. 7: Möglicher Einsatz von Augmented Reality in der Stadt [Bild01]

Abb. 8: Einsatz von Augmented Reality in der Archäologie [Bild02]

2.3 Interaktives, nichtlineares Storytelling

Im Gegensatz zu herkömmlichen, linearen Geschichten, die nur vom Autor beeinflusst werden können, steht hinter dieser Arbeit der Ansatz der nichtlinearen Narration. Eine nichtlineare Geschichte besteht nach [Braun03] „aus einem Storymind[5] und Throughlines[6], die an einer oder mehreren Stellen nicht fest vorgegeben und beeinflussbar sind“. Das bedeutet, dass der Zuhörer mit der Geschichte interagieren und dadurch den Verlauf der Geschichte beeinflussen kann. „Die Beeinflussung einer Geschichte durch den Zuschauer muss jedoch gezielt eingesetzt werden, um das Ziel des Autors tatsächlich erreichen zu können, denn der Autor steht bei interaktivem Storytelling vor einem besonderen Dilemma.

Dieses Dilemma ist vor allem seit der Verbreitung von Hypertext und den damit verbundenen Möglichkeiten der Hypertext-Storys bekannt geworden: Wann weiß das Publikum, dass eine Geschichte tatsächlich zu Ende[7] ist bzw. dass es auf dem richtigen Weg durch die Geschichte ist?

Die Möglichkeiten der Interaktion stellen den Autor aus dem oben notierten Grund vor besondere Schwierigkeiten: Er möchte dem Publikum seiner Geschichte ein bestimmtes Ziel in Reichweite bringen. Dieses Ziel wird jedoch, auf Grund der mannigfaltigen Einflussmöglichkeiten durch das Publikum, eventuell nicht erreicht. Der Autor muss dem zu Folge Vorkehrungen

---

[5] Die klassische Literaturtheorie kennt hier den Begriff der Fabula.

[6] Meint die Perspektive, aus der die Geschichte erzählt wird. Es wird zwischen den Perspektiven des allwissenden Erzählers, der Hauptperson, des Anti-Helden und der subjektiven Sicht mit dem Focus auf den Kampf zwischen Held und Anti-Held unterschieden.

[7] Ein sehr bekanntes Beispiel dieser Problematik wird anhand der Hypertext-Story Afternoon durch Walker, vgl. [Wal99], besprochen. Walker erkennt, dass die Suche nach dem Ziel bzw. nach dem Ende der Geschichte durch das Publikum ein nicht triviales Unterfangen ist. Letzten Endes weiß das Publikum erst, dass es das Ende der Geschichte erreicht hat, wenn es alle Hyperlink-Möglichkeiten ausprobiert hat.

treffen, um dem Publikum das Erreichen des Zieles zu ermöglichen ohne das Publikum in seiner Interaktionsfähigkeit zu sehr einzuschränken und damit die Immersion zu stören“ [Braun03].

Um die Kontrolle über das System nicht zu verlieren gibt es Mechanismen, die die Szenen verwalten. Bei diesem Verfahren der diskreten nichtlinearen Narration unterscheidet man zwischen zwei Ansätzen, die eine relativ gute Kontrolle zulassen:

- *Branching:* „Die nichtlineare Geschichte ist aus verschiedenen Zweigen aufgebaut (vgl. Abbildung 9). In jedem Zweig der nichtlinearen Geschichte sind feste Verzweigungspunkte integriert. An diesen Punkten müssen Entscheidungen getroffen werden, welche den Partizipienten der Geschichte in einen neuen Zweig der Geschichte führen. Für jeden Zweig existiert eine vordefinierte Dramaturgie, welche der Benutzer bis zum nächsten Entscheidungspunkt durchlebt“ [Braun03].

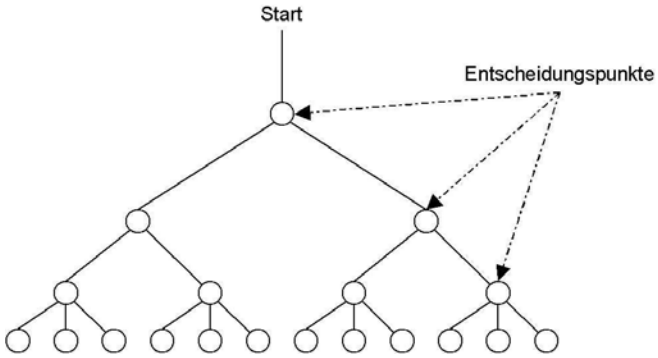

Abb. 9: Nichtlineare Erzählstruktur basierend auf Branching [Braun03]

- *String of Pearls:* Ähnlich zum Branching ist auch der String of Pearls aus verschiedenen Zweigen aufgebaut (siehe Abbildung 10). In jedem Zweig existieren notwendige Aufgaben und zusätzliche (quasi freiwillige) Aufgaben. Um von einem Zweig in den nächsten zu gelangen, müssen vom Benutzer alle notwendigen Aufgaben gelöst werden. Die zusätzlichen Aufgaben gestalten die Geschichte abwechslungsreicher. Der Benutzer muss nicht explizit wissen, wel-

che der Aufgaben notwendig und welche zusätzlich sind“ [Braun03].

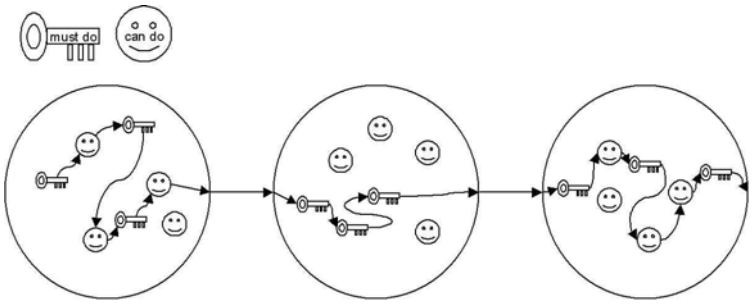

Abb. 10: Nichtlineare Erzählstruktur basierend auf String of Pearls [Braun03]

Wird lediglich eine grobe Kontrolle des Systems angestrebt, so kann man die Reihenfolge der Plots in einer nicht mehr netzartigen Struktur angelegen. „Eine solche Geschichte besteht nicht aus vorher definierten linearen Versatzstücken. Tatsächlich wird durch eine Evaluations-Funktion die Publikumsaktion so in die Narration der Geschichte einbezogen, dass der nächste Plot aus der Summe der Publikumsaktionen, der Historie von Plots und dem zu erreichenden Ziel der Geschichte bestimmt wird. Die Narration bildet quasi eine benutzerabhängige Funktion innerhalb eines relativen, nichtdiskreten Narrationsraumes“ [Braun03] (siehe Abbildung 11).

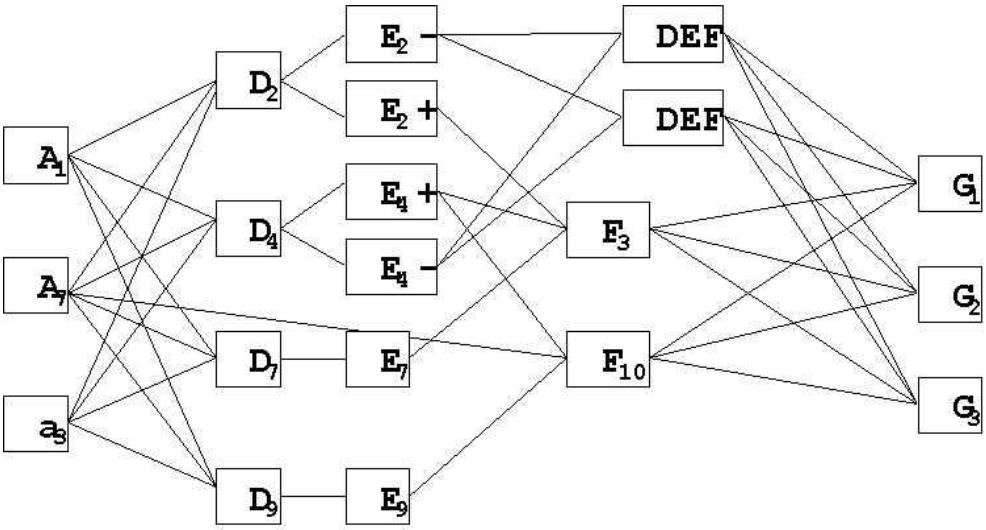

Abb. 11: Nichtlineare Erzählstruktur basierend auf morphologischen Funktionen [Braun03]

Das in dieser Arbeit verwendete nichtlineare Verfahren zur Vermittlung einer Geschichte entspricht dem zuletzt vorgestellten Modell. Dies ergab sich aus der Tatsache, dass die für ein anderes AR-Projekt entwickelte Story-Engine, die auch in EduTeCH zum Einsatz kommt, auf diesem Narrationsansatz aufbaut. Diesem Ansatz entsprechend müssen die einzelnen Szenen so konzipiert werden, dass sie Abhängigkeiten von vorherigen Spielzügen enthalten, die eine Entscheidung über das Abspielen der nächsten Szene ermöglichen.

2.4 Verwendete Hardware

2.4.1 Head-Mounted Displays

Bei allen Augmented Reality Systemen stellt sich die Frage des zu verwendenden Ausgabemediums. Zwei unterschiedliche Prinzipien können zum Einsatz kommen: optische oder videobasierte Systeme. Zunächst werden die grundlegenden Eigenschaften der Head-Mounted Displays (HMD) dargestellt. Im Anschluss daran werden die unterschiedlichen Funktionsweisen der Systeme erläutert und die allgemeinen Vor- und Nachteile der jeweiligen Technik diskutiert.

In den meisten HMDs werden zwei Monitore für die Darstellung der Bildinformation verwendet. Diese sind vor den Augen des Benutzers platziert und ermöglichen ihm so das räumliche Sehen. Dazu erzeugt der Computer stereoskopische Bilder, indem er für jeden Monitor ein eigenes, perspektivisch leicht verschobenes Bild berechnet (siehe Abbildung 12). Da die Verschiebung der Bilder dem natürlichen Augenabstand entspricht, hat der Nutzer einen ganz natürlichen Bildeindruck. Im Vergleich zum normalen Bildschirm hat der Benutzer stereoskopisch dargestellter Bilder daher einen sehr realistischen Eindruck der projizierten Bilder und kann dadurch viel leichter in die erweiterte Welt eintauchen.

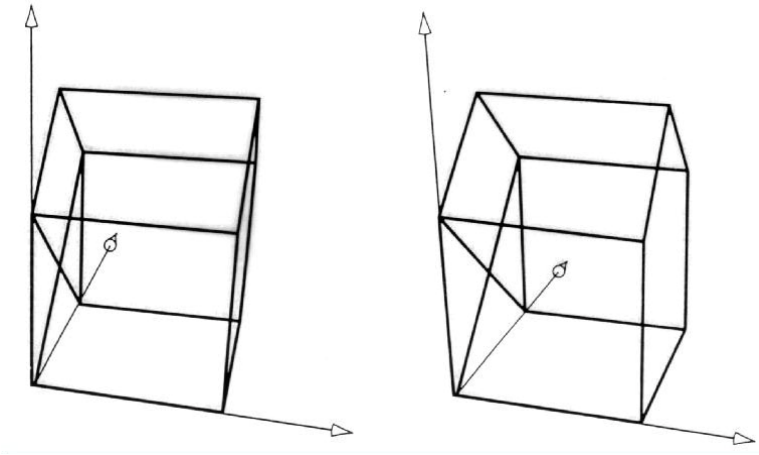

Abb. 12: Perspektivische Verschiebung eines Bildes [Leberl01]

Für die Monitore können entweder Röhrenmonitore (Cathod Ray Tube, CRT) oder Flüssigkristall-Bildschirme (Liquid Crystal Display, LCD) verwendet werden. Allerdings eignen sich CRT-Systeme nur für die videobasierten HMDs, da sie aufgrund ihrer Bauweise blickdicht sind. Außerdem sind sie wesentlich schwerer als LCDs. Vorteile haben die CRTs noch beim Bildaufbau, der wesentlich schneller erfolgt als bei den meisten LCDs, weshalb diese bei schnellen Bewegungen zur Schlierenbildung neigen. Lediglich sehr hochwertige und damit auch teure LCDs schaffen den Grauwechsel in so kurzer Zeit, dass sie das Nachziehen des Bildes verhindern können. Bei einem Vergleich der Bildqualität, sind inzwischen kaum noch Unterschiede zwischen den beiden Techniken festzustellen.

Ein Problem bei den HMDs stellt der eingeschränkte Sichtwinkel dar. Während das Blickfeld des menschlichen Auges in der horizontalen einen Sichtwinkel von etwa 120° bis 180° (in den Randbereich wird nur noch Bewegung wahrgenommen) und in der vertikalen von maximal 150° besitzt, verfügen die HMDs über ein wesentlich eingeschränkteres Blickfeld. So beträgt der horizontale Blickwinkel zwischen 45° und 110° und der vertikale liegt zwischen 25° und 65°. Durch ein HMD wird die Sicht des Benutzers deutlich eingeschränkt. Je kleiner die Blickwinkel des HMDs sind, desto eher entsteht eine Art „Tunnelblick“.

Bei den optischen HMDs ist es dem Benutzer möglich, seine reale Umgebung direkt zu sehen. Dies wird durch eine halbdurchsichtige Optik erreicht, die je nach Bauweise eine unterschiedliche Menge an Licht zu den Augen durchlässt. Die virtuellen Informationen werden von oben auf die Optik projiziert und von dort ins Auge reflektiert. Abbildung 13 zeigt eine schematische Skizze eines optischen HMDs.

Im Gegensatz zu optischen HMDs sind videobasierte Systeme blickdicht. Dies macht den Einsatz von zwei Videokameras auf dem HMD notwendig, um die reale Umwelt überhaupt wahrnehmen zu können. Der Rechner kombiniert die beiden Videosignale – zum einen reale Umwelt, zum anderen virtuelle Information – zu einem Ausgabesignal, das dann an die Monitore weiter geleitet wird. Der Aufbau eines videobasierten HMDs wird in Abbildung 14 verdeutlicht. Die Zusammenführung der realen und virtuellen Videosignale kann zum Beispiel mit Hilfe des so genannten Alphablending durchgeführt werden. Da bei dieser Methode das künstliche Bild bekannt ist, kann auf ein aufwändiges Keying-Verfahren verzichtet werden. Wenn das System über einen Tiefen-Index der realen Welt verfügt und so jeder Pixel eine Entfernungsangabe besitzt, wird es möglich, dass virtuelle Objekte von der realen Welt verdeckt werden oder umgekehrt.

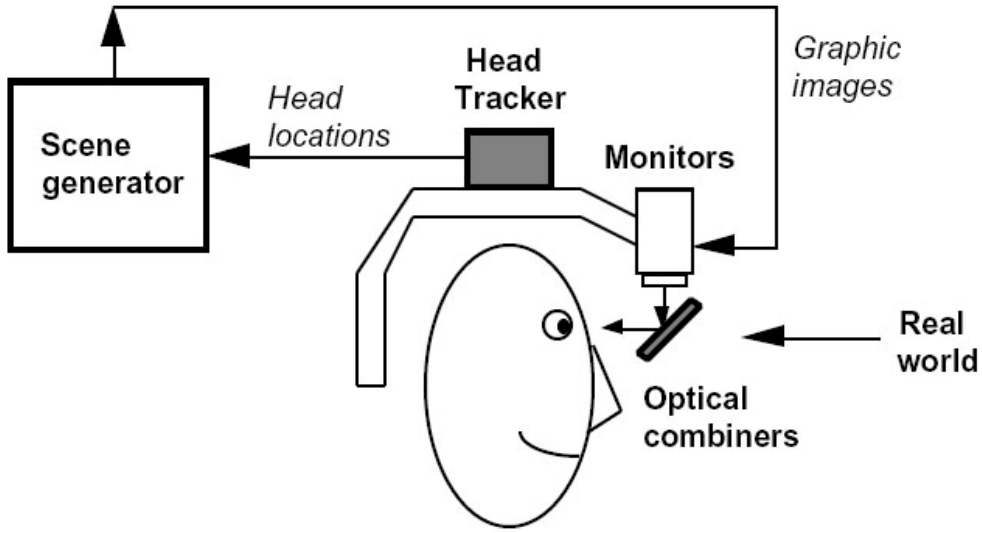

Abb. 13: Schematische Darstellung eines optischen HMDs [Azuma97]

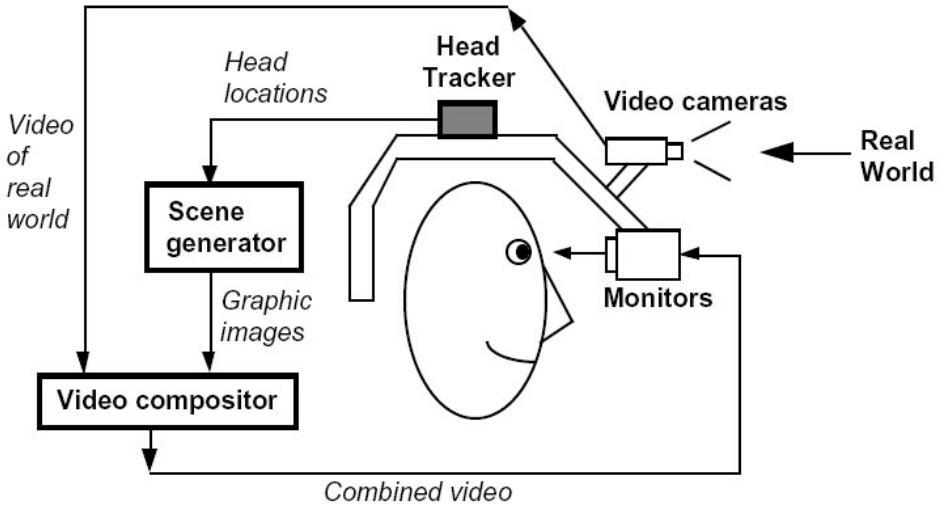

Abb. 14: Aufbau eines videobasierten HMDs [Azuma97]

Die unterschiedlichen Gerätekonzeptionen zeigen im Vergleich diverse Vor- bzw. Nachteile, aus denen sich die unterschiedlichen Einsatzmöglichkeiten ergeben. Im folgenden Abschnitt sollen diese Unterschiede kurz aufgezeigt werden.

Als Vorteile des optischen HMD sind insbesondere zu nennen:

- *Einfachheit:* Bei der optischen Variante muss lediglich ein Videosignal verarbeiten werden. Dies reduziert den Rechenaufwand erheblich.
- *Auflösung:* Beim videobasierten HMD sind sowohl die reale wie auch die virtuelle Sicht auf die Auflösung des verwendeten Displays beschränkt. Da bei den optischen HMDs aber nur der virtuelle Teil auf das Display übertragen wird, bleibt die reale Welt uneingeschränkt.
- *Sicherheit:* Bei einem Stromausfall ist Nutzer mit dem videobasierten HMD quasi blind, da keine Bilder mehr übertragen werden können. Das optische HMD wird in diesem Fall zu einer Art Sonnenbrille, da lediglich die virtuellen Zusatzinformationen fehlen und die reale Umgebung sichtbar bleibt.

Videobasierte HMDs bieten demgegenüber folgende Vorteile:

- *Flexibilität der Kompositionsstrategie:* Bei videobasierten HMDs können reale Objekte vollständig von virtuellen Objekten überlagert werden. Dies verstärkt die Immersion des Anwenders in das System. Bei optischen HMDs ist dies auf Grund der halbtransparenten Bauweise nicht möglich, wodurch die virtuellen Objekte immer leicht geisterhaft erscheinen.

- *Einheitliche Darstellungsebene:* Bei optischen Systemen treten Verzerrungen auf, wenn weit entfernte Objekte betrachtet werden. Dies liegt an dem großen Unterschied zwischen den Betrachtungsebenen, da die realen Objekte im Gegensatz zu den virtuellen Bildern eine Tiefe haben und nicht auf einer festen Projektionsebene liegen. Videobasierte Systeme verrechnen die beiden Welten und geben sie gemeinsam aus, so dass dieser Effekt nicht auftritt.

- *Abstimmen der Verzögerung:* Ein optisches HMD ermöglicht dem Anwender die verzögerungsfreie Sicht auf die reale Welt. Bei der Darstellung der virtuellen Objekte kann es jedoch zu Verzögerungen kommen, da diese erst auf die Bewegungen des Nutzers reagieren muss. Ein videobasiertes HMD kann in diesem Fall das Bild der realen an das der virtuellen Welt anpassen und so die Verzögerungen ausgleichen. Dadurch können sich allerdings neue Probleme ergeben, da bei einer größeren Verzögerung nicht mehr auf schnelle Bewegungen reagiert werden kann.

- *Erweiterte Zuordungsmöglichkeiten:* Durch die zusätzlichen Informationen, die das digitale Bild der realen Welt liefert, ist es möglich, die virtuellen Objekte besser in die reale Welt zu integrieren. Auch können reale Objekte die virtuellen verdecken, was bei optischen Systemen auf Grund der fehlenden Informationen nicht möglich ist.

- *Bessere Helligkeitsanpassung:* Im Idealfall sollten reale und virtuelle Objekte die gleiche Helligkeit besitzen. Eine Anpassung der Helligkeit ist jedoch nur in videobasierten Systemen möglich, da

nur hier Helligkeitsinformationen aus der realen Welt vorliegen und die virtuellen Daten daran angeglichen werden können. Zusätzlich kann die Beleuchtungsrichtung an die jeweilige Tages- bzw. Jahreszeit angepasst werden, wodurch auch der Schattenwurf der virtuellen Objekte entsprechend angepasst werden kann.

In der vorliegenden Arbeit kamen optische AR-Glasses zum Einsatz, da solche bereits von anderen AR-Projekten vorhanden waren. Die Entscheidung für diese Art von HMD wäre allerdings auch bei einer Neuanschaffung getroffen worden, da hierbei wesentlich weniger Technik notwendig ist und der Anwender auch seine reale Umwelt noch uneingeschränkt wahrnehmen kann, selbst wenn das System einmal ausfallen sollte.

2.4.2 Tracking-Systeme

Damit die Blickrichtung des AR-Systems mit der des Benutzers übereinstimmt, ist ein Head-Tracker notwendig. Dieses Gerät bestimmt die Ausrichtung des Kopfes in Bezug auf einen Ausgangswert. Somit kann der Computer ermitteln, in welche Richtung gerade geschaut wird. Ohne dieses Gerät ist es nicht möglich, die Projektion der AR-Daten korrekt auf die reale Umgebung abzubilden.

Es gibt verschiedene Arten von Tracking-Systemen, die die Kopfbewegung des Benutzers erfassen. Es wird zwischen zwei Gruppen von Messgeräten unterschieden: Zum einen gibt es absolute Tracking-Systeme (akustische, magnetische, mechanische und optische), zum anderen relative Tracking-Systeme (Accelerameter bzw. Beschleunigungssensoren). Basierend auf Haller ergibt sich folgende Übersicht für die einzelnen Tracking-Systeme [Haller02a]:

Absolute Tracking-Systeme:

- *Akustische Tracking-Systeme:* Hier werden Ultraschallsensoren verwendet, um die Position mit Hilfe von Laufzeitunterschieden in den Schallwellen zu ermitteln. Diese Systeme sind sehr preisgünstig und benötigen keine mechanischen Verbindungen. Negativ fallen allerdings die geringe Genauigkeit und eine niedrige Abtastrate auf. Auch muss immer eine Sichtverbindung zum Empfänger bestehen. Außerdem reagieren sie sehr sensibel auf Temperatur- und Feuchtigkeitsschwankungen.

- *Magnetische Tracking-Systeme:* Ein magnetischer Tracker verfügt über drei orthogonale magnetische Felder, die im Sender integriert sind. Der Empfänger misst die Feldänderungen und berechnet mit den Messdaten die genaue Position und Ausrichtung des Senders [FREI03]. Von Vorteil sind die schnelle Verarbeitung der Daten sowie die gute Genauigkeit und die gute Abtastrate. Nachteilig wirken sich die Beeinflussbarkeit durch ferromagnetische Felder (Monitore, Stahlträger, Stromleitungen etc.), die abnehmende Genauigkeit (sie nimmt mit dem Quadrat der Entfernung ab) und die geringe Reichweite (bis zu 3 Meter Durchmesser) aus.

- *Mechanische Tracking-Systeme:* Bei diesem System besteht eine mechanische Verbindung zwischen dem Objekt und dem Referenzpunkt. Durch Winkelmessungen wird die Position des Anwenders bestimmt [FREI03]. Vorteile der mechanischen Tracker sind ihre geringe Latenzzeit sowie ihre hohe Genauigkeit. Als Nachteil erweisen sich die unkomfortable Handhabung sowie die eingeschränkte räumliche Freiheit.

- *Optische Tracking-Systeme:* Ein optischer Tracker arbeitet mit Hilfe von 2 bis 16 um den Benutzer verteilten Kameras, um die relativen Veränderung zu angebrachten Messpunkten zu erfassen. Diese Systeme sind sehr genau, haben eine hohe Abtastrate (bis zu 250 Abtastungen pro Sekunde) und sind unempfindlich gegenüber Temperatur und Luftfeuchtigkeitschwankungen. Nachteilig sind die hohen An-

schaffungskosten, mögliche Störungen durch auftretende Reflexionen sowie der notwendige Sichtkontakt.

- *Hybride Tracking-Systeme:* Bei den hybriden Tracking-Systemen werden mehrere unterschiedliche Sensoren miteinander kombiniert. Dadurch können die Vorteile der verschieden Tracker-Systeme ausgenutzt werden und es wird neben einer guten Genauigkeit auch eine hohe Abtastrate erreicht. Leider ist der Aufbau des Trackers sehr komplex.

Relative Tracking-Systeme:

- *Accelerameter/ Beschleunigungssensoren:* Die relativen Tracking-Systeme arbeiten mit Hilfe von Translation und Rotation. Sie messen dabei die Kraft, die auf eine beschleunigte Masse wirkt und berechnen daraus die Blickrichtung des Nutzers. Das System benötigt dabei keinen Referenzsender und kann unabhängig von der Raumgröße eingesetzt werden. Einen Nachteil stellt jedoch der sich addierende Fehler dar (’Drift’). Deshalb ist es notwendig, den Tracker vor jeder Messung zu kalibrieren.

Das in dieser Arbeit verwendete Tracking-System der Firma InterSense gehört zu den relativen Trackern und besteht aus drei Sensoren, die die Rotation um die drei Raumachsen erfassen. Als Sensoren werden dabei Gyroskope verwendet. Diese "rotierenden Körper" weichen immer senkrecht zur einwirkenden Kraft aus, wodurch die Abweichung zum Anfangswert gemessen und dadurch auch die Neigung des Kopfes berechnet werden kann. Wie bereits bei den anderen Geräten, wurde auch hier auf den bereits vorhandenen Bestand des ZGDV zurückgegriffen. „Diese Tracker sind klein und kostengünstig. Die besten unter ihnen arbeiten mit einer Genauigkeit von 0,5 Grad. Sie sind besonders für den Outdoor-AR-Bereich geeignet, da keine Anpassung der Umgebung beispielsweise durch Sensoren notwendig ist“ [KRENZEL04].

2.4.3 Global Positioning System

Um die korrekte Überlagerung der realen mit der virtuellen Welt zu erreichen, ist neben dem Tracking der Blickrichtung auch die exakte Positionsbestimmung des Anwenders notwendig. Bei früheren AR-Systemen wurde deshalb auf größere Benutzerbewegung verzichtet. Stattdessen wurden die Bewegungen durch Tretmühlen oder Laufbänder ermittelt. Dies ermöglicht auch die Nutzung von räumlich begrenzter Hardware (z.B. mechanische Tracker), da der Anwender trotz seiner Bewegung seine Position im Raum nicht veränderte.

Bei der Entwicklung von Outdoor-AR-Systemen ist es zwingend erforderlich zu berücksichtigen, dass sich der Anwender frei in der realen Welt bewegt. Seine exakte Position muss jederzeit bestimmt werden können, um so eine Anpassung der virtuellen Objekte zu ermöglichen. Diese Tatsache stellt erhebliche neue Ansprüche an das benötigte Equipment, da dieses nun leicht und transportabel sein muss, um den Anwender nicht in seiner Bewegungsfreiheit einzuschränken.

Aus diesen Gründen wird vor allem bei Outdoor-AR-Anwendungen das Global Positioning System (GPS) eingesetzt. „Das bei GPS angewandte Ortungsverfahren beruht auf dem Prinzip der Entfernungsbestimmung durch Messung der Laufzeit von Signalen zwischen dem Nutzer und je einem von mehreren Bezugspunkten (hier Satelliten), deren Positionen genau bekannt sind.“ [Schum96]. Für eine genaue Ortung wird der Empfang von mindestens drei Satelliten benötigt. Mit den Daten der drei Satelliten lässt sich die genaue Position des Anwenders triangulieren. Die Position ist in der Regel auf drei Meter genau [Haller02]. Da für die GPS-Peilung ein Sichtkontakt zu den Satelliten bestehen muss, kann diese Technik nicht für die Positionsbestimmung in Häusern verwendet werden.

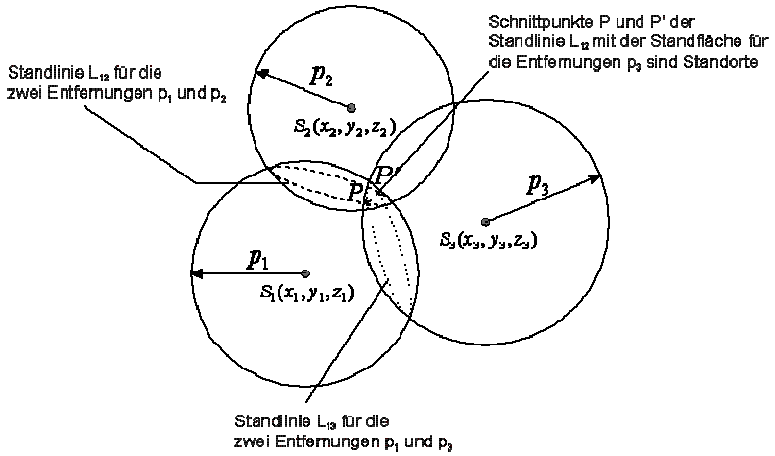

Abb. 15: Dreidimensionales Orten eines Objektes im Raum [Schum96]

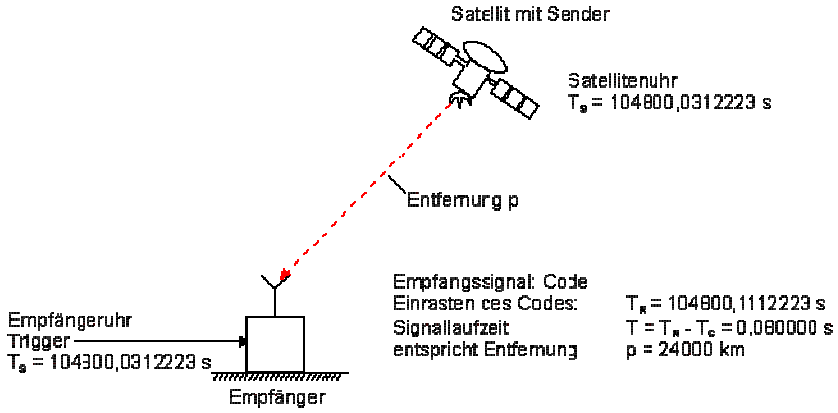

Abb. 16: Prinzip der Entfernungsmessung – Einwegmessung [Schum96]

Um Fehler bei der Positionsbestimmung per GPS zu reduzieren, wurde das Differential-GPS (DGPS) entwickelt. „Die Grundidee besteht darin, dass Beobachtungsdifferenzen frei von vielfältigen Fehlereinflüssen, wie sie bei Einzelmessungen mit gleichem Vorzeichen und Betrag auftreten, sind. Um dies zu realisieren wird mit Hilfssendern gearbeitet. Dabei wird die Position der DGPS - Sende- / Empfangsstation sehr genau vermessen. Die mit dieser Station empfangenen bzw. berechneten Positionsdaten werden mit den tatsächlichen (bekannten) Positionsdaten verglichen und zu einem Korrekturwert verarbeitet. Dieser wiederum wird … über einen Funkkanal (z.B. LW) mehreren mobilen Empfängern zu Verfügung gestellt“ [Schum96].

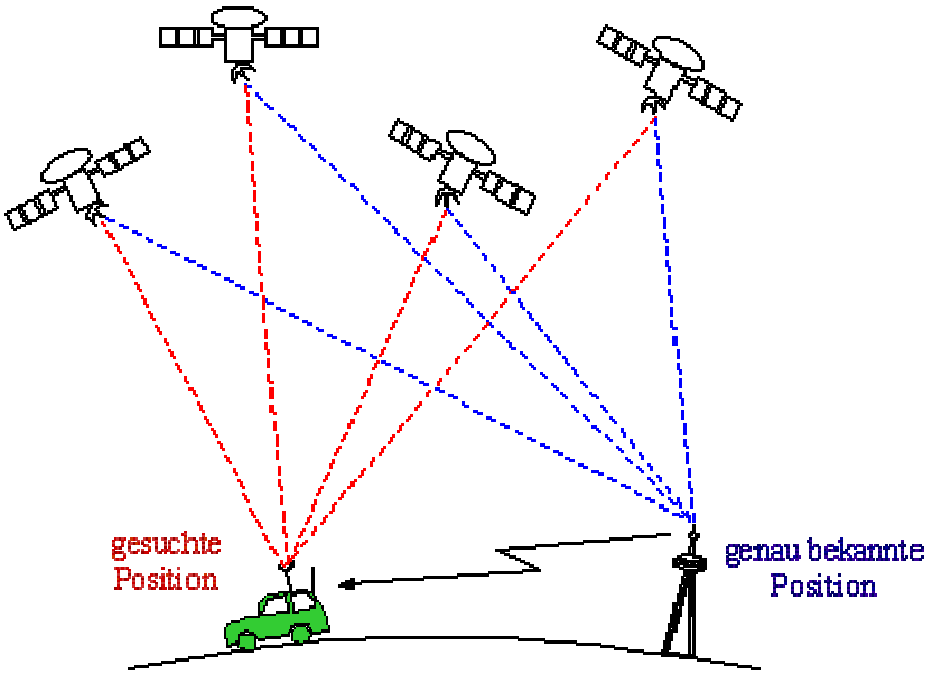

Abb. 17: Prinzip des DGPS bei der Fahrzeugnavigation [Schum96]

Die größten Probleme der AR-Systeme liegen in der Überlagerung der virtuellen mit der realen Welt sowie in den Sensorenfehlern der Tracking-Systeme. Da es nicht möglich ist, die unter 2.2 definierten Bedingungen für ein AR-System zu erfüllen, wenn das verwendete Tracking-System nicht fehlerfrei funktioniert, wird auch zukünftig bei der Entwicklung von AR-Systemen viel Zeit in die optimale Integration der virtuellen Welt investiert werden müssen. Denn nur so ist eine exakte Positionierung des Benutzers zu gewährleisten, die für die exakte Überlagerung der realen Objekte zwingend erforderlich ist [Azuma97].

2.4.4 Zusammenfassung

Nach diesem Überblick über die verschiedenen zur Verfügung stehenden Techniken bleibt festzuhalten, dass die in dieser Arbeit verwendeten Geräte alle den heute geltenden Ansprüchen gerecht werden. Da ausschließlich Geräte verwendet werden, die bereits im Bestand des ZGDV vorhanden sind, entspricht nicht mehr alles dem neuesten Stand der Technik, was jedoch auch nicht zwingend erforderlich ist. Für die zu bewältigenden Aufgaben sind alle Geräte problemlos geeignet, da sie für frühere Augmented Reality Projekte angeschafft wurden und somit optimal im vorliegenden Projekt genutzt werden können.

Auf die Schwierigkeiten, die sich bei der Einbindung der verschiedenen Geräte und bei deren Betrieb ergeben haben, wird in den entsprechenden Abschnitten des fünften Kapitels eingegangen.

2.5 Überblick der verwendeten Software

2.5.1 Betriebssystem

Als Betriebssystem, unter dem die in der Diplomarbeit erstellte Anwendung entwickelt wurde und auf dem diese später auch zum Einsatz kommen soll, wurde die Linux-Distribution Gentoo[4] ausgewählt. „Gentoo Linux ist eine quellenbasierte Linux-Distribution für fortgeschrittene Benutzer, die ihr System komplett individuell einrichten wollen. Dies hat neben dem Geschwindigkeitszuwachs (im Vergleich zu anderen Betriebssystemen; Anmerkung des Verfassers) aber auch zur Folge, dass sich die Installation wesentlich komplizierter gestaltet. Um Gentoo zu installieren, sollte man schon fundiertes Wissen über seine Hardware und über GNU/Linux mitbringen“ [GenLia].

„Die Distribution benutzt nicht wie die meisten anderen Distributionen vorkompilierte Programmpakete, sondern nur Quelltexte, die man selbst kompilieren muss. Lediglich sehr große Programme sind binär zu erhalten, hierzu zählen z.B. KDE und OpenOffice.org. Anfangs konnte man nur das Basissystem optional fertig kompiliert installieren (Stage 2), aber mittlerweile werden auch CDs mit binären Paketen vertrieben (Stage 3), die eine zeitraubende Erstinstallation vereinfachen. Stage 1 beinhaltet zudem das […] Bootstrapping, also den Aufbau eines Minimal- bzw. Basis-Systems, das zum weiteren Bau des Systems benötigt wird. Gentoo Linux eignet sich hervorragend für alle Personen, die möglichst viel Kontrolle über ihr System erhalten möchten.

[8] Der Name des Betriebssystems, das von Daniel Robbins ins Leben gerufen und bis 2004 betreut wurde, stammt von den Eselspinguinen (englisch: gentoo), einer kleinen, aber besonders schnellen Pinguin-Art.

Gentoo hat ein BSD-ähnliches Paketverwaltungssystem namens *Portage*. Das Tool, um Portage zu benutzen, heißt *emerge*. Ein *emerge sync* aktualisiert zum Beispiel über das Internet den lokalen *Portage tree* auf der Festplatte. Er enthält eine komplette Sammlung von […] *ebuild*-Scripten. Darin werden Abhängigkeiten, Downloadort der Sourcen und der Ablauf der Kompilierung festgelegt. Anhand dieser Informationen ist es Portage möglich, Abhängigkeiten selbständig aufzulösen, Updates durchzuführen und vieles mehr.

Grundsätzlich läuft eine Softwareinstallation mit Portage wie folgt ab: Der Benutzer muss nur als root auf der Kommandozeile *emerge paketname* (z.B. *emerge apache*) eintippen. Portage sucht daraufhin das entsprechende ebuild im Portage tree und überprüft, ob die Software andere Pakete benötigt, um zu funktionieren. Sollte eine solche Abhängigkeit bestehen, wird sie aufgelöst, indem die benötigten Pakete installiert werden. Ist das erledigt, lädt Portage den Quelltext des Programms als komprimierte Datei vom Server und überprüft per MD5-Prüfsumme, dass die Datei nicht kaputt oder verfälscht ist. Anschließend wird sie temporär entpackt und das Makefile erstellt, das zur Kompilierung unbedingt notwendig ist. Der Quelltext wird nun in einer Sandbox kompiliert. Der Compiler ist dabei in der Lage, das Paket auf die benutzte Hardware zu optimieren. War die Kompilation erfolgreich, werden Dokumentation (Manpages etc.), Konfigurationsdateien und natürlich die entstandenen Binärdateien in die entsprechenden Verzeichnisse auf dem System kopiert. Die Software ist nun installiert.

Gentoo ist unter diversen Architekturen lauffähig. Dazu zählen Alpha, AMD64, Itanium, MIPS, PA-RISC, PowerPC, SPARC und x86. Es gibt auch Projekte, bei welchen der Linux-Kernel durch einen FreeBSD- bzw. OpenBSD-Kernel ersetzt wurde. Zudem kann Apples Mac OS X mittels Portage auf fast den gesamten Pool der EBuilds zugreifen, ohne das native Betriebssystem ersetzen zu müssen ("Gentoo for Mac OS X")“ [GenLia].

2.5.2 Blender

Die Open-Source[9] Software Blender (siehe Abb.18) wurde ursprünglich als firmeninterne Animationssoftware entwickelt und erst später frei zugänglich gemacht. Heute bietet die Software jedoch wesentlich mehr als ein Animationstool. Wird der Leistungsumfang mit anderen kommerziellen 3D-Programmen wie Maya, 3DStudio Max oder Softimage verglichen, so kann festgestellt werden, dass die Unterschiede nicht sehr groß sind. Die größten Unterschiede zeigen sich dabei in den Spezialgebieten der einzelnen Software. So hat 3DStudio Max durch das integrierte Charakterstudio Vorteile im Bereich der Charakteranimation. Zwar sind mit Blender auch diese Animationen möglich, aber sie sind nicht so einfach zu erstellen, da im Vergleich die oftmals sehr effektiven Hilfsmittel fehlen. Die Einarbeitung in Blender ist sehr zeitintensiv, da der Funktionsumfang sehr groß ist und die Bedienung des Programms recht stark von anderen 3D-Tools abweicht. Nach der Einarbeitung, die auch bei Programmen wie Maya ähnlich lange dauert und unbedingt notwendig ist, fällt die Orientierung jedoch deutlich leichter. Der große Vorteil von Blender ist jedoch, dass in das Programm eine Game-Engine integriert ist. Diese ist zwar längst nicht so leistungsstark wie die bei aktuellen 3D-Spielen (Doom3, Halflife2 usw.) verwendeten Engines, aber durchaus leistungsfähig genug, um auch umfangreichere Szenarios darstellen zu können. Ein Wechsel aus dem Modelliermodus ins Spiel erfolgt durch Drücken der Taste ’P’. Im Gamemodus ist es möglich sich frei durch die modellierten Objekte zu bewegen. Da die Game-Engine sowohl über eine Kollisionserkennung als auch eine recht gute Physik verfügt, sind die Einstellungsmöglichkeiten sehr umfangreich, wodurch sehr gute und ansprechende Ergebnisse erzielt werden können.

---

[9] Der Ausdruck Open Source steht für *quelloffen*, einerseits in dem Sinne, dass der Quelltext eines Programms frei erhältlich ist, andererseits für 'offene Quelle', also dass ein Werk frei zur Verfügung steht.

Abb. 18: Oberfläche von Blender 2.3.4 beim ersten Programmstart

Blender bleibt zwar gegenüber den jeweiligen Spezialanwendungen in deren Spezialgebieten zurück, dies aber wird durch die gelungene Kombination von 3D-Software und Game-Engine wieder ausgeglichen. Durch diese Kombination eignet sich Blender besonders gut für dieses Projekt.

2.6 Die Programmiersprache Python

Als Guido van Rossum ein neues Betriebssystem entwickeln und testen wollte, benötigte er eine Testumgebung. Aus diesem Grund erfand er 1990 am Centrum voor Wiskunde en Informatica in Amsterdam die Programmiersprache Python. Sein Ziel war es dabei, sehr schnell kleine Testprogramme entwickeln zu können, sowie eine leichte Erweiterbarkeit der Programmiersprache und ihrer Programme zu ermöglichen. Nachdem der Sourcecode 1991 veröffentlicht und somit frei zugänglich wurde, haben viele Programmierer die Möglichkeiten von Python erkannt und erweitert. Heute liegt Python in der Version 2.4 vor.

Python zählt zu den interpretierten (oder: Interpreter-) Sprachen, zu denen beispielsweise auch Smalltalk oder Basic gehören. Das Gegenstück stellen die kompilierten (oder: übersetzten) Sprachen wie C oder FORTRAN dar. Bei den kompilierten Sprachen wird der Programmtext erst übersetzt und dann an den Prozessor weitergeleitet. Da dieser Zwischenschritt bei den Interpreter-Sprachen fehlt, wird das Programm direkt ausgeführt. Welche Vorgehensweise nun die bessere ist, hängt von der jeweiligen Verwendung ab. So eignen sich die Interpreter-Sprachen besonders für die Entwicklung von Prototypen (Rapid Prototyping), da der oftmals sehr zeitaufwendige Schritt des Kompilierens wegfällt. Dadurch wird ein deutlich schnelleres Arbeiten ermöglicht. Die kompilierten Sprachen haben demgegenüber bei der Ausführung des Codes deutliche Geschwindigkeitsvorteile, was an der unmittelbaren Ausführbarkeit liegt, da hierbei kein Interpreter mehr dazwischen geschaltet ist. Um dieses Manko auszugleichen, generiert und speichert Python eine interne Darstellung des Quelltextes, den so genannten Bytecode. Der Bytecode, der zum Python-Skript `sePerson.py` gehört,

wird in `sePerson.pyc` abgelegt. Der Zeitaufwand, um den Bytecode zu erzeugen, ist im Vergleich zum Maschinencode deutlich geringer, da er keine reine Zeichenfolge ist, sondern dem Quelltext noch sehr ähnlich ist. Er spiegelt lediglich die Programmstruktur wider.

Ein weiterer Vorteil des Interpreters ist die Fehlertoleranz. Hauptursache für einen Programmabsturz und den oftmals damit verbundenen Datenverlust ist, dass die übersetzten Programme direkt auf dem Prozessor ausgeführt werden. Tritt dabei ein Fehler auf, so wird das Programm mit Hilfe des Betriebssystems sofort beendet. Falls allerdings ein Interpreter zwischen Programm und Betriebssystem/Prozessor steht, kann dieser alle Aktionen vor ihrer Ausführung überprüfen und im Fehlerfalle mit Programm- und Nutzerunterstützung ein sanftes Ende herbeiführen. Dadurch wird der Verlust von Daten verhindert. Die Grundidee hierbei ist die Ausnahmebehandlung (exception handling) [Loewis01].

In einem kurzen Überblick können noch einige weitere Vorteile der Programmiersprache Python auf Basis von [Pammer00] aufgeführt werden:

- objektorientiert: Python unterstützt neben den gängigen Konzepten der Objektorientiertheit auch die Mehrfachvererbung.
- dynamisch typisiert: Es gibt keine statischen Typen, die bei der Deklaration angegeben werden. Jede Variable kann Objekte eines beliebigen Typs annehmen, einschließlich Klassen- und Funktionsobjekten.
- leicht zu erlernen: Die Syntax ist klar und einfach und ermöglicht somit eine schnelle Einarbeitung.

2.7 Zusammenfassung

In diesem Kapitel wurden die Grundlagen, die für das Verständins der vorliegenden Arbeit notwendig sind, erläutert und deren Einsatz im Projekt verdeutlicht. Dabei wurden sowohl die Begriffe der Virtual- und Augmented-Reality, sowie der des interaktiven, nichtlinearen Storytellings definiert, als auch die für dieses Projekt notwendige Hardware vorgestellt. Dazu wurden zuerst die heutigen Möglichkeiten beschrieben und darauf aufbauend die Entscheidung für das verwendete Equipment begründet. Zusätzlich wurden die verwendete Software und die eingesetzte Skriptsprache kurz vorgestellt und ihre Vorteile herausgearbeitet.

Das nun folgende Kapitel befasst sich mit der erstellten Virtual Reality Anwendung und verdeutlicht deren Aufbau und Funktionsweise.

3. Anforderungen an die VR Anwendung

3.1 Zielsetzung der VR Anwendung

Ziel der entwickelten Virtual Reality Anwendung ist es, dem Benutzer einen virtuellen Rundgang durch die koreanischen Tempelanlage Gyeongbok Palace in Seoul zu ermöglichen.

Dabei wird der Nutzer auf den Avatar[10] Sung Sam Mun treffen, der einen historischen Schüler des Tempels verkörpert. Dieser Avatar bietet dem Spieler die Möglichkeit, in der Tempelanlage ein spielerisches Abenteuer zu erleben, indem er ihn bittet, bei der Suche nach den verschwundenen Buchstaben des koreanischen Alphabets zu helfen. Während dieser Suche trifft der Anwender immer wieder auf den Avatar, der zwar immer das gleiche Aussehen hat, hinter dem sich aber auch der Widersacher von Sung Sam Mun verbergen kann. Es ist also die Aufgabe des Spielers, den Avatar auf Grund seiner Aussagen der entsprechenden Person zuzuordnen und sein Verhalten ihr gegenüber dementsprechend anzupassen. Beantwortet der Anwender die an ihn gestellten Fragen richtig, so kann er Punkte sammeln. Für richtige Antworten auf Fragen, die von Sung Sam Mun gestellt wurden, erhält er zusätzlich einen Buchstaben. Bemerkt der Spieler nicht, dass die Frage von dem Gegenspieler des Avatars gekommen ist, so verliert er einen der gesammelten Buchstaben. Durch die gesammelten Punkte ist es dem Spieler möglich, den falschen Avatar zu enttarnen und dieser löst sich dann in Luft auf. Misstraut er jedoch Sung Sam Mun, so bricht dieser die Unterhaltung beleidigt ab. In Anlehnung an die später erläuterte Augmented Rea-

[10] **Avatar** (von Sanskrit *Avatara*, Herabkunft)
Ein Avatar ist eine künstliche Person oder ein grafischer Stellvertreter einer echten Person in der virtuellen Welt. Avatare werden beispielsweise in Form eines Bildes, Icons oder als 3D-Figur eines Menschen oder sonst irgendeines Wesens dargestellt. Die Verwendung des Begriffes Avatar in diesem Zusammenhang wurde 1992 von Neal Stephenson in seinem Science-Fiction-Roman "Snow Crash" populär gemacht [adLexicon].

lity Anwendung sieht der Spieler den Avatar nur, wenn er seine virtuelle Brille aufsetzt. Ist diese nicht aktiviert, sieht der Nutzer lediglich die normale Umgebung der Tempelanlage.

Im nun folgenden Kapitel werden die einzelnen Schritte beschrieben, die zur Umsetzung des eben beschriebenen Konzeptes notwendig sind.

3.2 Modellierung der virtuellen Umgebung

3.2.1 Gebäude und Landschaft

Damit am Ende der Arbeit eine ansprechende Virtual Reality-Anwendung vorliegt, ist es von großer Bedeutung, dass die virtuelle Welt möglichst real erscheint. Um dies zu erreichen, muss die Modellierung wie auch die Texturierung der Umgebung genau und detailgetreu erfolgen. Auf Grund der späteren Nutzung als Echtzeit-Anwendung sind hierbei allerdings die Leistungsgrenzen der Game-Engine zu beachten, da der Spielfluss bei einer Überschreitung deutlich schlechter und somit der Gesamteindruck der Anwendung negativ beeinflusst wird. Es muss also ein Kompromiss gefunden werden, der es ermöglicht, dass die Landschaft real erscheint, ohne dass der Spielfluss darunter leidet. Aus diesem Grund wurde für die Modellierung der 3D-Objekte das Verfahren des Polygonmodellings gewählt. Bei dieser Modellierungsmethode werden alle Objekte aus Grundkörpern aufgebaut und dadurch rechenintensive Objekte vermieden, wie sie beispielsweise bei der Verwendung von Bezier-Kurven entstehen. Außerdem wurden im Anschluss an die Modellierung alle verdecken Flächen (wie z.B. die Innenseiten der Wände) gelöscht und somit eine Vielzahl von Faces eingespart.

Bevor jedoch mit der Modellierung begonnen werden konnte, musste der Umfang der virtuellen Landschaft geklärt werden. Da es auf Grund des begrenzten Zeitrahmens nicht möglich war, das gesamte Palastgelände zu modellieren, wurde die Anzahl der Modelle auf die für das Projekt not-

wendigen Gebäude beschränkt. Bei der Größe des virtuellen Geländes wurden die Originalmaße übernommen und die ausgewählten Gebäude an der richtigen Stelle platziert. Um das Erscheinungsbild der Umgebung aufzuwerten, wurden einige Bäume in das Szenario eingefügt. Als Modellierungsvorlagen gab es einen Übersichtsplan der gesamten Palastanlage, wodurch die gesamten Ausmaße des Geländes recht gut zu bestimmen waren. Auch die Lage und ungefähre Grundfläche der einzelnen Gebäude ließ sich daraus entnehmen. Da ansonsten keine technischen Zeichnungen der Gebäude zur Verfügung standen, musste die Modellierung an Hand von Fotographien durchgeführt werden. Die erste Aufgabe bestand also darin, Baupläne mit Größenangaben für die zu modellierenden Gebäude zu erstellen. Dazu wurden in den vorliegenden Bildern Elemente mit bekannter Größe gemessen und diese Werte dann auf andere Bildelemente übertragen. Dabei entstehen jedoch mehr oder weniger große Fehler durch die bei Bildern auftretende perspektivische Verzerrung. Die Modelle wurden dann anhand dieser Zeichnungen angefertigt (siehe Abb. 19).

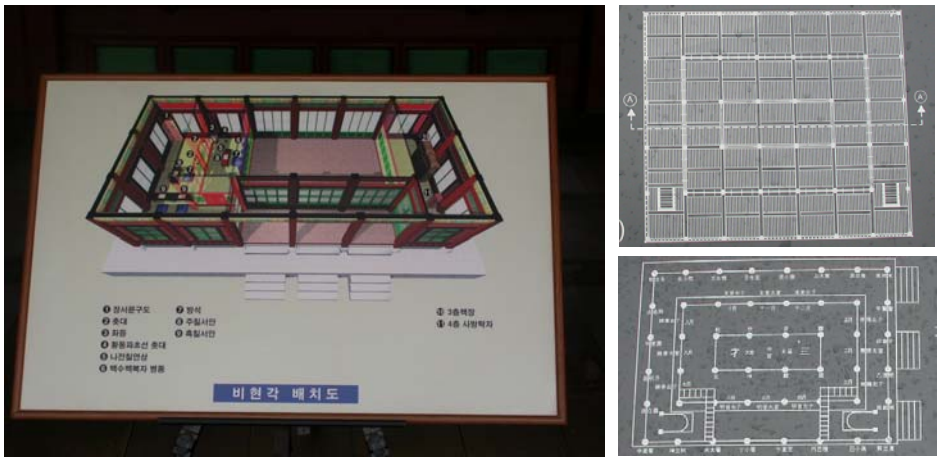

Abb. 19: Übersichten und Pläne, mit deren Hilfe die Modelle erstellt wurden

Die Modellierung der Gebäude ist dabei recht einfach, da sich die Häuser im Aufbau alle sehr stark ähneln und fast ausschließlich aus Rechtecken bestehen. Somit lassen sich auch viele Elemente in einem Haus wieder verwenden. Es wird also eine Säule mit Sockel und ein Wandelement modelliert und diese dann einfach kopiert und wieder an der richtigen Stelle eingefügt. Wesentlich aufwändiger wird dann jedoch die Modellierung des Da-

ches, da dieses, wie im asiatischen Raum üblich, keine gerade, sondern eine geschwungene Form mit diversen Überständen und Überschneidungen hat. Um diese Konstruktion mit möglicht wenigen Polygonen realistisch darstellen zu können, ist eine gute Anordnung der Flächen nötig, damit der Effekt einer geschwungenen Linie entsteht (siehe Abb. 20). Da sich jedoch die Dachform bei einigen Gebäuden wiederholte, war es möglich, auch diese mehrmals zu verwenden. Die Landschaft wurde möglichst einfach gestaltet, da sich eine komplexe Bodenplatte zu negativ auf die Spieleperformance ausgewirkt hätte. Die Bäume wurden in Form von sternförmig angeordneten Planes erstellt, die dann mit einer Baumtextur mit Alpha-Kanal versehen wurden. So ließ sich ohne große Polygonzahl (nur drei Planes pro Baum) ein sehr realistischer Eindruck erzeugen.

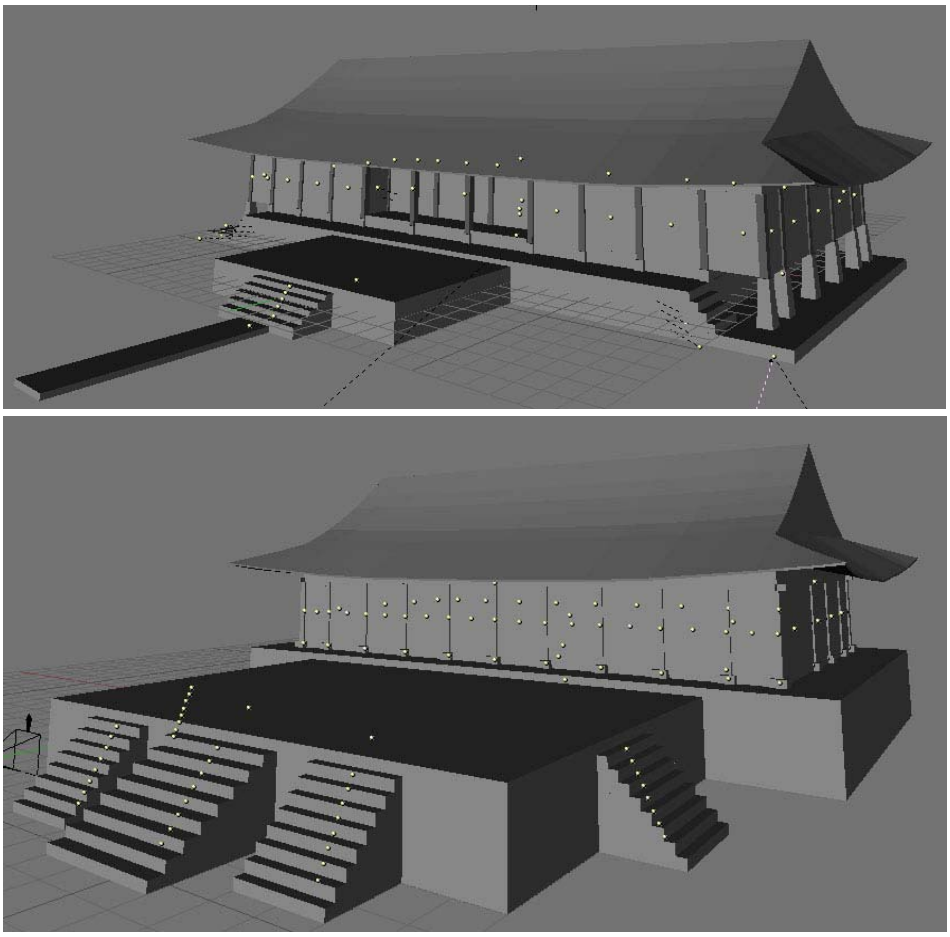

Abb. 20: Screenshots zweier untexturierter Gebäude

Die Texturierung der Gebäude gestaltete sich schwierig, da die benötigten Texturen aus den zur Verfügung stehenden Digitalphotos hergestellt werden mussten. Da diese Aufnahmen jedoch eigentlich nur als Übersichten gedacht waren, traten gleich mehrere Probleme auf. Zum einen haben Übersichtsbilder einen recht großen Bildausschnitt, was jedoch für die Erstellung von Texturen zu einem schwerwiegenden Problem wird. Denn die Texturen werden bei der Projektion auf die zu texturierende Fläche oftmals stark vergrößert, was dann ein unscharfes und grobes Aussehen zur Folge hat (siehe Abb. 21). Ein weiterer negativer Punkt ist der Winkel, aus dem die Bilder fotografiert wurden. Für die Textur wäre es optimal, wenn sich die zu

Abb. 21: Vergleich zwischen guter (links) und schlechter Textur (rechts)

fotografierende Fläche senkrecht zum Objektiv befände, da somit eine Verzerrung der Textur verhindert würde. Dies ist jedoch bei den wenigsten Aufnahmen der Fall, so dass die perspektivischen Verzerrungen mit Hilfe der Bildbearbeitung ausgeglichen werden müssen. Ein weiteres Problem ergibt sich aus der Tatsache, dass es sich bei dem Palastgelände um ein frei zugängliches Freilicht-Museum handelt und dort folglich viele Touristen sind. Das heißt aber auch, dass viele der Bilder einzelne Besucher oder sogar Be-

suchergruppen enthalten, die in der Textur unerwünscht sind. Diese Personen müssen dementsprechend aus den für die Texturierung verwendeten Partien entfernt werden. Nach diesen notwendigen Vorarbeiten konnten die Modelle texturiert werden. Da die Texturen in das Blend-File integriert werden, muss jede Textur nur einmal eingebunden werden und kann dann beliebig oft verwendet werden. Dadurch kann die Größe der entstehenden Datei so gering wie möglich gehalten werden. Aus diesem Grund wurden auch ausschließlich Bilder im JPEG-Format verwendet, da diese komprimiert werden können und somit deutlich weniger Speicherplatz beanspruchen. Die Texturierung der Modelle erfolgte durch UV-Mapping, was eine optimale Anpassung der Texturen an die Flächen ermöglichte.

3.2.2 Charakter

Bei der Modellierung des Avatars galt es mehrere verschiedene Forderungen zu berücksichtigen. Da insgesamt neun Avatare im Spiel zum Einsatz kommen, muss besonders auf die Polygonanzahl geachtet werden. Andererseits sollte jedoch auch ein realistisch aussehender Avatar entstehen, der zudem die Möglichkeit besitzt, für das Spiel animiert zu werden. Diese Anforderung machte es notwendig, den Charakter zu segmentieren, damit später die einzelnen Körperteile getrennt voneinander bewegt werden können. Deshalb wurde der Avatar in die folgenden Segmente unterteilt: Kopf mit Hut, Oberkörper, Arme, Beine und Schuhe.

Bei der Modellierung des Kopfes wurde auf einen bereits vorhandenen Standard-Kopf zurückgegriffen. Dieses Modell hatte jedoch das typische Aussehen eines mitteleuropäischen Jungen und musste deshalb noch stark verändert werden, um das gewünschte Gesicht eines koreanischen Mannes zu erhalten. Damit die spätere Kopfform den Vorstellungen entsprach, gab es eine Front- und eine Seitenansicht des Avatarkopfes, mit deren Hilfe dann die Veränderungen vorgenommen wurden. Dazu wurden die Vertices einzeln an die neue Form angepasst und so nach und nach das gewünschte Erscheinungsbild erreicht. Nachdem der Kopf modelliert war, mussten die

Kleider des Charakters erstellt werden. Auf die Modellierung eines Körpers konnte verzichtet werden, da damals als Hoftracht sehr weite, bis zum Boden reichende Gewänder getragen wurden. Durch die vor dem Körper verschränkten Arme, die auch in den späteren Animationen in dieser Position bleiben sollten, konnte auch auf die Modellierung der Hände verzichtet werden. Der Körper des Avatars besteht also aus zwei glockenförmigen Strukturen, die den Oberkörper bzw. den Rock bilden, sowie einem Schlauch für die Arme. Diese Formen wurden so angepasst, dass beim Spieler der Eindruck von Stoff entsteht. Als letztes Kleidungsstück wurden die Schuhe erstellt und einfach unter dem Rock angefügt.

Nach der Modellierung musste dem Avatar durch die Textur das richtige Erscheinungsbild gegeben werden. Durch geschickte Auswahl und Anpassung der Texturen können dann Probleme oder sogar Fehler bei der Modellierung ausgeglichen werden. Entsprechend der einzelnen Kleidungsstücke gibt es jeweils eine dazugehörige, speziell angepasste Textur (siehe Abb. 22). Durch die richtigen Texturen erhält der Avatar seinen ganz eigenen Charakter und eine besondere Ausstrahlung, er wird sozusagen „zum Leben erweckt“ (siehe Abb. 24).

Die Texturierung des Modells erfolgte wie bei der Landschaft durch UV-Mapping. Hierbei werden die Faces, die texturiert werden sollen, markiert und die zu verwendende Textur in den UV-Editor geladen. In diesem Editor lässt sich dann die Projektion der Textur auf dem Modell anpassen. Dazu werden die Faces einfach auf der Textur verschoben. Durch Auswahl einzelner Vertices ist so eine sehr genaue Anpassung möglich (siehe Abb. 23).

Damit der Eindruck eines lebendigen Charakters entsteht, ist neben der optimalen Texturierung auch eine real wirkende Animation des Charakters notwendig. Blender verfügt über verschiedene recht umfangreiche Möglichkeiten zur Animation von Objekten. Zum einen können Objekte über *key frames* animiert werden. Weitere Möglichkeiten sind die Verwendung von *motion curves*, *path* oder *armature*. Blender fasst dabei die *key frames* und

die *motion curves* unter dem Begriff des *IPO-Systems* (**I**nter**PO**lation) zusammen. Da die Game-Engine lediglich die Animation durch IPOs unterstützt, sollen die beiden anderen Methoden lediglich kurz erklärt werden.

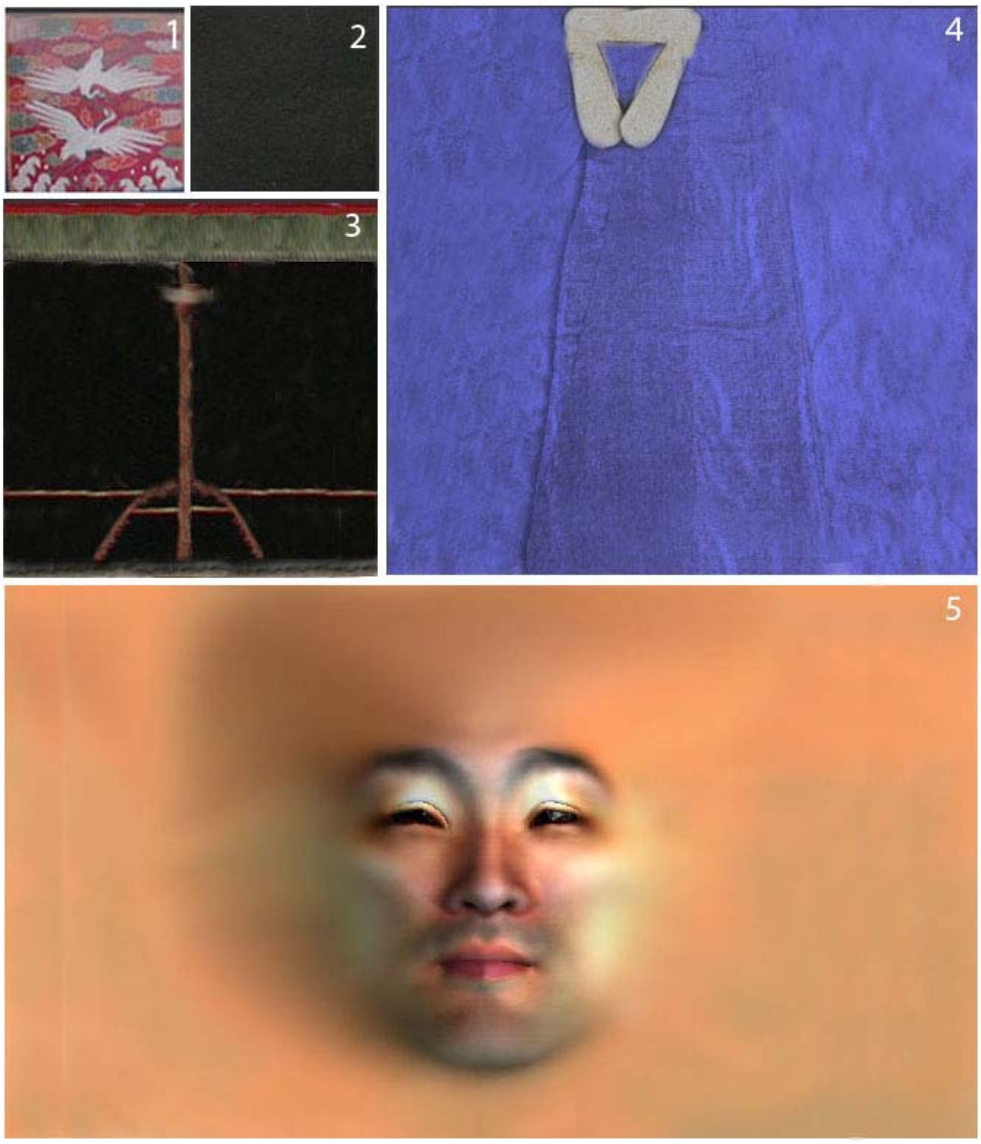

Abb. 22: Übersicht der für den Avatar verwendeten Texturen
(1: Verzierung, 2: Hut, 3: Schuhe, 4: Kleidung, 5: Gesicht)

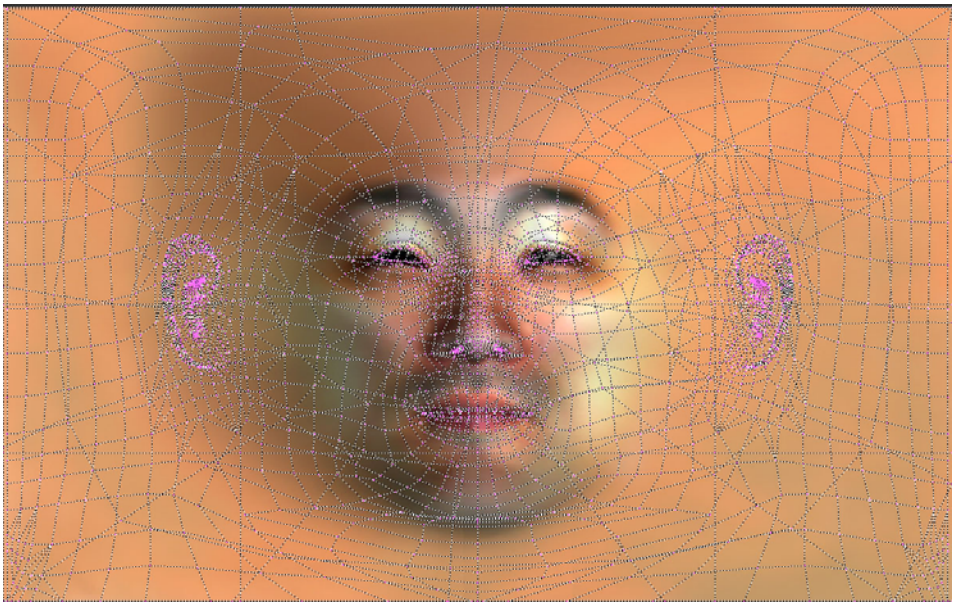

Abb. 23: Verteilung der Faces des Kopfes auf der Textur für das Gesicht

Abb. 24: Modell des Avatars in verschiedenen Phasen
(von links: Wireframe, Shaded und Texturiert)

Das IPO-System von Blender umfasst die Animation durch *key frames* und *motion curves*. Bei einer *key frame* Animation werden bestimmte Positionen der Objekte zu einer Zeit definiert und der Computer interpoliert dann die zwischen den Zuständen notwendige Bewegung. Dieser Animationstyp gleicht der Erstellung eines Comics, bei dem die Zeichner durch leicht veränderte Bilder eine Bewegung erzeugen. Die Animation durch *motion curves* ermöglicht eine bessere Kontrolle der Bewegungen. Hierbei gibt es jeweils für die Position, die Rotation und die Größe drei Kurven (XYZ). Diese Kurven bestimmen die Bewegung des Objektes, wobei in der Horizontalen die Zeit und in der Vertikalen der Wert der Veränderung aufgetragen werden. Durch die Zusammenfassung dieser beiden Animationsmethoden ergibt sich ein wesentlicher Vorteil. So können die einzelnen Positionen sehr leicht durch die *key frames* gesetzt werden und dann die Bewegung zwischen den Positionen genau über die *motion curves* gesteuert werden. Dies ist ohne großen Aufwand möglich, da in Blender zwischen den Animationsarten durch einfaches Umschalten gewechselt werden kann, ohne dass sich am Ergebnis etwas verändert.

Beim System der Pfad-Animation wird eine Linie in den drei dimensionalen Raum gezeichnet, an der sich das Objekt ausrichtet und entlang derer es sich bewegt. Die Bewegung wird über eine vorher definierte Zeitfunktion, die die Geschwindigkeit des Objektes bestimmt, gesteuert.

Wird die Animation mit Hilfe der *armature* erstellt, so wird nicht das Mesh direkt animiert, sondern das durch die *armature* erstellte Skelett. Dieses wird in das Mesh eingefügt. Anschließend wird das Mesh mit dem *armature* verknüpft, damit es bei Bewegungen dem *armature* folgt. Somit wird nur noch das *armature* animiert, und der Körper führt diese Bewegungen dann aus.

Wie schon zuvor beschrieben werden jedoch weder die *armature* noch die Pfad-Animation von der Game-Engine gegenwärtig unterstützt, weshalb auf die Animation durch das IPO-System zurückgegriffen wurde. Damit sich der Avatar verbeugt, müssen die verschiedenen Bestandteile des A-

vatars einzeln animiert werden. Der Anfang der Animation erfolgt durch die *key frame* Methode. Das heißt, dass der Avatar in der Grundposition steht und das erste Schlüsselbild erzeugt wird. In der Timeline muss nun das Ende der Animation festgelegt werden. Dazu wird um die entsprechende Anzahl an Frames die die Animation dauern soll, nach vorne gegangen und dort wird erneut ein Schlüsselbild angelegt. Bei der hier beschriebenen Verbeugung sind das 170 Frames, was einer Dauer von 6,8 Sekunden entspricht. Damit sich der Avatar aber auch bewegen kann, muss nun noch in der Mitte der Animationsdauer ein Schlüsselbild erstellt werden. Bei diesem Schlüsselbild muss dann die Extremposition des Avatars erstellt werden. Das bedeutet, dass an dieser Stelle der tiefste Punkt der Verbeugung liegt. Wird diese Animation ablaufen gelassen, so verbeugt sich der Avatar und kehrt in die Ausgangsposition zurück. Allerdings fällt dabei auf, dass die Bewegungen noch nicht flüssig sind und dementsprechend noch nicht real wirken. Damit dies erreicht wird, kommt die Methode der *motion curves* zur Anwendung. Dabei werden die vom Computer interpolierten Bewegungen zwischen den Schlüsselbildern manuell verändert. Im IPO-Editor wird dies realisiert durch Veränderungen an den Bezier-Kurven, die zwischen den Schlüsselbildern verlaufen. Diese Anpassungen müssen auch wieder einzeln für jedes zu animierende Teil des Avatars durchgeführt werden. Dabei muss ständig das Gesamtergebnis kontrolliert werden, damit sich die verschiedenen Körperteile auch synchron und in einer anatomisch korrekten Position bewegen (siehe Abbildung 25). Da die Animation der Körperteile relativ zu ihrer aktuellen Position und nicht als absolute Werte gespeichert wird, ist sie universell für alle im Spiel verwendeten Avatare einsetzbar. Bei der Animation der Lippenbewegung ist nach demselben Schema vorgegangen worden. Der einzige Unterschied besteht darin, dass nicht ganze Objekte verschoben wurden, sondern nur einzelne Vertices. Da Blender bei der IPO-Animation jedoch keinen Unterschied zwischen Objekten, Vertices oder Materialien macht, bleibt das Animationsprinzip das gleiche.

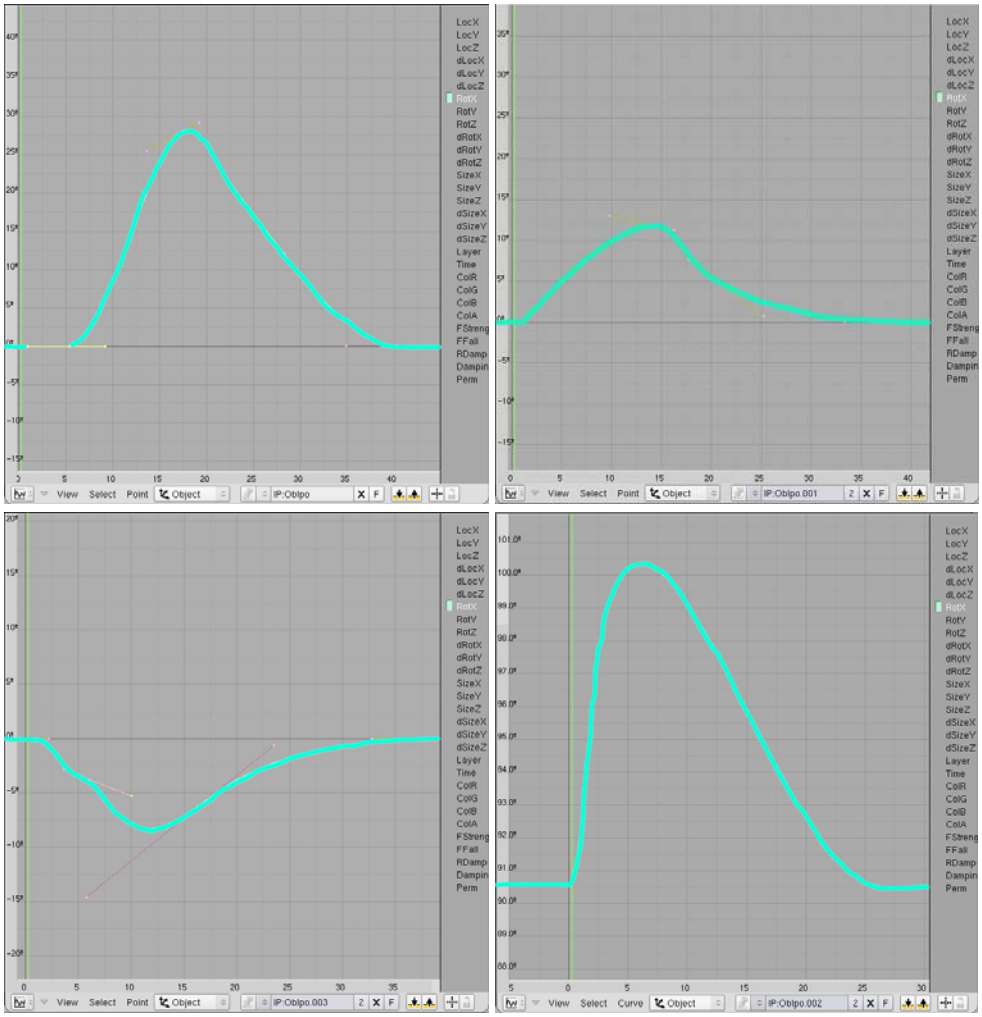

Abb. 25: Animationskurven des Charakters für Kopf, Oberkörper, Arme und Rock (von links oben nach rechts unten)

3.3 Programmieren der Game-Logic

Die Programmierung der Game-Logic (siehe Abb. 25) stellt einen der wichtigsten Bereiche dieser Diplomarbeit dar. Die Game-Logic ist von entscheidender Bedeutung für das Spiel, da sie die Möglichkeiten und den Umfang des Spiels festlegt. Aus diesem Grund ist bei der Erstellung der Game-Logic darauf zu achten, dass sie klar strukturiert ist, dadurch übersichtlich

bleibt und leicht um neue Aspekte erweitert werden kann. In diesem Abschnitt wird auf die allgemeinen Punkte eingegangen, die bei der Erstellung einer Game-Logic mit Blender zu beachten sind. Auf die Funktionsweisen der einzelnen Skripte wird dann in den folgenden Kapiteln eingegangen.

Jedem in Blender erstellten Objekt können mit Hilfe der *Realtime-Buttons* Logik-Elemente zugewiesen werden, die in der Summe aller Objekte die Game-Logic bilden. In der Ansicht der *Realtime-Buttons* werden alle Logic-Bricks gezeigt, die das Spiel enthält. Die Ansicht kann durch Abschaltung von Verknüpfungen, Verlinkungen und Auswahlmöglichkeiten vereinfacht werden. Sie umfasst drei Spalten, die *Sensors*, *Controllers* und *Actuators*. Die *Sensors* haben die Aufgabe, während des Spielverlaufs auf bestimmte, vorher definierte Ereignisse zu achten. Dafür gibt es verschiedene Arten von Sensoren, die zum Beispiel auf Tastatureingaben (*Keyboard*), Mausbewegungen (*Mouse*), Annäherungen des Spielers an vorher definierte Objekte (*Near*) oder aber auf bestimmte Eigenschaften (*Property*) reagieren. Die *Controllers* haben die Funktion einer Brücke zwischen den *Sensors* und den *Actuators*. Neben den booleschen Verknüpfungen *AND* und *OR* können hier auch Python-Skripte aufgerufen werden. Durch die *Actuators* werden dann Reaktionen auf die abgefangenen Ereignisse gestartet. So können unter anderem Animationen oder Soundfiles aufgerufen und abgespielt werden. Zusätzlich dazu ist es möglich, die Objekte durch *Properties* zu erweitern. Diese können als lokale Variablen der Objekte betrachtet werden, die für die gesamte Dauer des Spiels existieren und auf die auch aus Skripten zugegriffen werden kann. Als mögliche Typen stehen *Boolean*, *Integer*, *Float* und *String* zur Verfügung sowie ein *Timer*, der die seit dem Start vergangene Zeit angibt. Dabei ist jedoch nicht zwangsweise die Spieldauer gemeint, sondern die Zeit, die seit dem Entstehen des Objektes (es können auch Objekte während des Spiels neu erstellt werden) vergangen ist.

Auf Grund der umfangreichen Physik-Engine sind zahlreiche Einstellungen der Umgebung möglich, um so dem Anwender ein realeres Spielgefühl zu vermitteln. So ist es unter anderem möglich, die Dichte der Umgebung zu ändern, um so zum Beispiel den Unterschied zwischen der Luft und

dem Wasser in einem See zu simulieren. Da die Dichte von Wasser größer ist, wird sich der Spieler in diesem Medium durch den erhöhten Widerstand deutlich langsamer bewegen als in der Luft. Ein ähnlicher Effekt ist auch für die Beschaffenheit von Oberflächen möglich, wodurch dann zum Beispiel Eis oder aber Sand erzeugt werden. Es gibt außerdem die Möglichkeit mit anderen Objekten im Spiel zu interagieren, sofern diese dafür vorbereitet wurden. Der Spieler könnte dann beispielsweise eine Kiste aufheben oder einen Ball schießen. Alle Objekte sind mit einer Kollisionserkennung ausgestattet, wodurch verhindert wird, dass der Spieler oder bewegte Objekte sich durch andere Objekte hindurch bewegen.

Anhand dieser Möglichkeiten wurde die Game-Logic für EduTeCH entwickelt. Auf Grund des Umfangs und der sehr komplexen Struktur des Spiels müssen die einzelnen Bereiche gut strukturiert sein. Das Gesamtszenario kann dazu in die Bereiche Landschaft, Spieler, virtuelle Brille, Avatare und Spielkontrolle unterteilt werden, was jedoch nicht heißt, dass die einzelnen Bereiche von einander unabhängig sind, da es eine Vielzahl von Verknüpfungen untereinander gibt.

Für die Spielkontrolle sind drei Objekte außerhalb des eigentlichen Spielfeldes angelegt worden. Dabei handelt es sich um den *nearController*, den *textController* und den *modeController*.

Der *nearController* überprüft, wo sich der Spieler gerade befindet und welche Objekte in seiner Nähe sind. Dabei muss unterschieden werden zwischen permanent sichtbaren Objekten und solchen, die nur unter bestimmten Voraussetzungen (z.B. aktivierte virtuelle Brille) sichtbar sind. Nähert sich der Spieler einem nur temporär sichtbaren Objekt, so wird eine dafür angelegte Variable auf *true* gesetzt. Entfernt er sich wieder, so wird sie auf *false* zurückgesetzt. Um den entsprechenden Avatar, dem sich der Spieler nähert, eindeutig identifizieren zu können, wird durch das Skript *globalNear.py* die ID des Objektes ausgelesen und gespeichert.

Der *textController* hat die Aufgabe, die in der virtuellen Brille eingeblendeten Buttons zu initialisieren sowie die entsprechenden Texte (Fragen, Antworten usw.) in die dafür vorgesehenen Textfelder zu schreiben. Dies geschieht mit Hilfe des Skripts *textControll.py*, das seine Anweisungen aus dem Skript *sePerson.py* bezieht, welches wiederum die aktuelle Frage bzw. Antwort von der Story-Engine übermittelt bekommt.

Während die gerade beschriebenen Kontrollobjekte lediglich einen Sensor besitzen, ist der Aufbau des *modeControllers* schon um einiges komplizierter. Dies gilt auch für seine Aufgabe, die darin besteht, aus dem Betrachtungs-Modus (der Spieler läuft nur in der Anlage umher) in den Auswahl-Modus (der Spieler hat die Brille aktiviert und kommuniziert mit dem Avatar) zu wechseln und umgekehrt. Der Wechsel zwischen den beiden Modi wird dabei wie folgt vollzogen: Im Betrachtungs-Modus steuert die Maus die Blickrichtung des Spielers. Wird dabei die `G`-Taste gedrückt, so wird die virtuelle Brille eingeblendet. Die Steuerung ändert sich aber noch nicht. Allerdings ist der Spieler nun in der Lage, auch die vorher verborgenen Objekte (z.B. die Avatare) zu sehen. Drückt der Anwender bei aktivierter Brille die linke Maustaste, so wechselt er in den Auswahl-Modus. Das bedeutet, dass die Maus nicht länger die Blickrichtung steuert, sondern den in der Mitte des Fensters eingeblendeten Mauszeiger. Ist ein Avatar in der Nähe, so kann ihn der Nutzer mit der Maus anwählen und so mit ihm kommunizieren. Auch die Beantwortung der Fragen erfolgt durch die Auswahl per Mauszeiger und die Bestätigung durch das Drücken der linken Maustaste. Durch das Drücken der rechten Maustaste kann der Anwender jederzeit aus dem Auswahl-Modus in den Betrachtungs-Modus zurückkehren.

Die Gruppe der Avatare gliedert sich in neun Charaktere, die sich auf die neun verfügbaren Bühnen verteilen. Ein Avatar besteht wie bereits erläutert aus mehreren Elementen (Kopf, Oberkörper usw.), um ihn animieren zu können. Daraus ergibt sich für die Logik jedoch das Problem, dass jedes dieser sechs Bauteile alle Properties, Sensors, Controllers und Actuators benötigt, um in der Game-Engine korrekt zu funktionieren. Das hat zur Folge, dass anstatt von neun Objekten nun 54 Objekte mit den Eigenschaften und

Verknüpfungen versehen werden müssen, da diese Version der Game-Engine für den Bereich der Game-Logic keine Möglichkeit der Gruppierung bietet. Jedes Element des Avatars erhält Sensoren, die es ermöglichen, dass eine Interaktion des Spielers mit ihm möglich wird. Um dies zu erkennen, verfügen die Elemente über verschiedene *Mouse-Sensoren*, die zum Beispiel ein mouseover-Event oder ein Drücken der linken Maustaste abfangen und daraufhin die Kommunikation starten. Zusätzlich werden weitere Sensoren benötigt, die den Status des Avatars überwachen und feststellen, ob der Spieler in der Nähe des Avatars ist und ihn somit sehen kann. Ist dies der Fall, so muss geprüft werden, ob der Avatar enttarnt wurde und dadurch nicht mehr angezeigt werden darf. Diese Informationen werden in ihrer Gesamtheit an das Skript `sePerson.py` weitergeleitet und dort verarbeitet. Auf die genaue Funktionsweise dieses Skriptes wird später genauer eingegangen (Kapitel 4.3.2 Ablauf des Spiels).

Die Gruppe, die unter dem Oberbegriff virtuelle Brille zusammengefasst ist, stellt einen elementaren Bestandteil des Spiels dar, da durch sie erst die Kommunikation mit den Avataren möglich wird. Sie stellt in der Virtual Reality-Anwendung quasi das Bindeglied zwischen der realen und der virtuellen Welt dar. Darum besitzt die Brille auch eine Vielzahl von Eigenschaften, die sich je nach Situation den Gegebenheiten anpassen. Grundvoraussetzung für die Interaktion mit einem Avatar ist, dass die Brille aktiviert wurde. Ist diese Bedingung erfüllt und hat der Spieler den Avatar angeklickt, so kommen die weiteren Eigenschaften der Brille zum Vorschein. Während der Avatar mit dem Benutzer redet, erscheinen mehrere Felder im unteren Teil der Brille. Dabei handelt es sich zum einen um Textfelder und zum anderen um Buttons zur Interaktion. Die Textfelder erhalten über den *textController* die auszugebenden Inhalte. Die Buttons bestehen lediglich aus einer durchsichtigen Fläche, die vor ein Textfeld gelegt wurde. Zusätzlich zu diesen Elementen enthält die Brille im oberen Bereich eine Anzeige für die bereits gesammelten Buchstaben sowie einen Zähler für den aktuellen Punktestand.

Abb. 26: Screenshot der GameLogic

Die Landschaft der Palastanlage tritt in der Game-Logic nur in sofern in Erscheinung, als dass sie als Kollisionsobjekt vorhanden ist. Damit wird verhindert, dass der Spieler durch die Umgebungsobjekte (Häuser, Bäume usw.) hindurchgehen kann beziehungsweise dass er beim Spielstart durch den Boden hindurch fällt.

Die in diesem Kapitel beschriebene Game-Logic stellt das Herzstück eines jeden Computerspiels da. Durch sie werden zum einen der Spielumfang und die Interaktionsmöglichkeiten festgelegt und zum anderen auch der Spielfluss sichergestellt. Der Spielspaß wird somit zu einem großen Teil von der Logik eines Spieles bestimmt, weshalb bei der Entwicklung eines Spiels ein großer Teil der Entwicklungszeit auf die Programmierung der Game-Logic entfallen sollte. Da der Spielfigur als Schnittstelle zwischen dem Nutzer und dem Spiel eine besondere Bedeutung zukommt, soll deren Aufbau und Funktion im folgenden Kapitel ausführlich veranschaulicht werden.

3.3.1 Steuerung des Spielers

Bei diesem Spiel wird die klassische Perspektive eines Ego-Shooters verwendet. Der Anwender sieht die virtuelle Umgebung durch die Augen des virtuellen Spielers. Dieser virtuelle Spieler besteht aus einem *Empty*, dem eine *Kamera* hinzugefügt wurde. Damit die Kamera immer über dem Empty bleibt, wurde sie mit Hilfe der *Parent-Funktion* an dieses gebunden. Bewegt der Anwender also später das Empty (entspricht dem Körper des Spielers), so bewegt sich die Kamera (entspricht den Augen des Spielers) ohne Verzögerung mit. Um zu verhindern, dass die Kamera in Dachüberstände oder Wände eintaucht, wurde der virtuelle Spieler noch mit einer Bounding-Box versehen, die eine Kollisionserkennung ermöglicht.

Um das Empty nun aber bewegen zu können, muss die entsprechende Logik für dieses Element angelegt werden. Bevor jedoch die Spielfigur bearbeitet werden kann, muss der virtuellen Welt eine Schwerkraft zugewiesen werden. Dies geschieht über die *World-Buttons*, mit denen über die

Auswahl *Sumo* die Schwerkraft aktiviert wird und deren Stärke über den nebenstehenden Regler variiert werden kann. Wechselt man danach bei markiertem Empty in die Realtime-Buttons, so findet man links oben den Eintrag *Actor*. Wählt man diesen aus, so erhält man weitere Einstellmöglichkeiten. Mit Hilfe des *Dynamik-Buttons* gibt man der Spielfigur ein Gewicht in der virtuellen Welt und somit unterliegt diese dann der dort herrschenden Schwerkraft. Zusätzlich sind dort weitere Einstellungen möglich, so lässt sich zum Beispiel die Größe des Objektes angeben (*Size*) oder die Elastizität einstellen.

Da die Steuerung des Spielers der eines Ego-Shooters entsprechen soll, wird die Bewegung mit Hilfe der Tastatur und die Blickrichtung über die Maus gesteuert. Dabei entspricht dem ↑ ein Schritt nach vorne, dem ↓ ein Schritt nach hinten, der ← hat einen Schritt seitwärts zur Folge und zwar nach links und der → dementsprechend dieselbe Bewegung nur nach rechts. Wird die Maus nach links bzw. nach rechts bewegt, so dreht sich der Spieler in die entsprechende Richtung. Eine Mausbewegung nach oben bzw. unten hebt bzw. senkt den Kopf (also die Kamera) des Spielers. Es existieren für den Spieler also die *Keyboard-Sensoren* ↑, ↓, ← und →, die das Drücken der entsprechenden Taste abfangen, sowie ein *Mouse-Sensor*, der auf *mousemove* reagiert. Wird eines der eben beschrieben Ereignisse registriert, so wird das an den entsprechenden *Controller* weitergeleitet. In diesem Fall wird dann immer das Python-Skript `player.py` (siehe Kapitel 9.1 Anhang A) aufgerufen.

Das Python-Skript `player.py` unterscheidet zwischen drei verschiedenen Arten von Bewegungen, die der Anwender steuern kann. Zum einen kann er den Spieler über das Palastgelände gehen lassen, zusätzlich ist es möglich, dass der Spieler den Kopf dreht. Neben diesen beiden Möglichkeiten kann im Auswahl-Modus auch noch der Mauszeiger durch das Skript gesteuert werden. Befindet sich der Spieler im Auswahlmodus, so sind keine Bewegungen mehr möglich. Es kann lediglich der Mauszeiger bewegt werden, bis dieser Modus durch Drücken der rechten Maustaste wieder beendet wird. Um immer die korrekte Funktion für die aktuelle Bewegungs-

form aufrufen zu können, führt das Skript verschiedene Abfragen durch. Dabei wird überprüft, ob sich die Maus bewegt und ob sich der Spieler im Auswahlmodus befindet. Den Ergebnissen entsprechend werden dann die jeweiligen Funktionen aufgerufen, die die Bewegungen steuern. Im Skript erfolgt das durch den folgenden Codeausschnitt:

```
if mousemove.isPositive():
  if not modeController.selectmode:
    init()
    look()
if modeController.selectmode:
  select()
if not modeController.selectmode:
  walk()
```

Wird eine Bewegung der Maus bemerkt und der Auswahlmodus ist nicht aktiviert, so werden die beiden Funktionen `init()` und `look()` aufgerufen. Die Funktion `init()` setzt die Position der Maus in die Mitte des Bildschirms. Dazu fragt sie die Höhe und Breite des Fensters über den `Rasterizer` ab und setzt diese Werte jeweils halbiert als Startwerte für x und y. Zusätzlich werden noch Steuerungsvariablen gesetzt, die in anderen Teilen des Skripts die Mausempfindlichkeit regulieren oder aber Mauspositionen zwischenspeichern. Diese Schritte sind Teil der Klasse `STB`, damit auch in anderen Funktionen des Skripts auf diese zugegriffen werden kann. Am Ende der Funktion wird dann noch die Position der Maus in die Mitte des Bildschirms gesetzt.

```
def init():
  class STB:
    initx = int(Rasterizer.getWindowWidth()/2)
    inity = int(Rasterizer.getWindowHeight()/2)
    mousex = 0.0
    mousey = 0.0
    sensitivity = 0.4
    mousefilter = 0.55
  GameLogic.STB = STB()
Rasterizer.setMousePosition(STB.initx, STB.inty)
```

Die Funktion `look()` steuert die Bewegungen des Kopfes, das heißt, ob der sich der Kopf hebt oder senkt bzw. ob er zur Seite gedreht wird. Um die Rotation des Spielers zu bestimmen, wird durch die Funktion `mouse()` die Bewegung der Maus berechnet und somit die Stärke der Drehung bestimmt. Die Funktion `look()` bestimmt anhand dieses Wertes dann die Rotation und ruft über den entsprechenden Frame der Animation `playerrotation` die neue Position des Spielers. Die Steuerung für das Heben bzw. Senken des Kopfes funktioniert nach einem ähnlichen Prinzip, jedoch mit dem Unterschied, dass die Position nicht durch eine Animationskurve sondern direkt durch einen *Motion-Actuator* bestimmt wird.

```
def mouse():
  nmousex = mousemove.getXPosition() – GameLogic.
            STB.initx
  nmousey = mousemove.getYPosition() – GameLogic.
            STB.inity
  GameLogic.STB.mousex = GameLogic.STB.mousex +
            (float(nmousex) - GameLogic.STB.mousex)*
            GameLogic.STB.mousefilter
  GameLogic.STB.mousey = GameLogic.STB.mousey +
            (float(nmousey) - GameLogic.STB.mousey)*
            GameLogic.STB.mousefilter

def look():
  from Rasterizer import setMousePosition
  mouse()
  player.rot -= -(GameLogic.STB.mousex * GameLogic.
                  STB.sensitivity * 0.25)
  playerrotation = controller.getActuator ('player
                   rotation')
  GameLogic.addActiveActuator (playerrotation, 1)

  upndown = controller.getActuator ('upndown')
  player.pitch -= (GameLogic.STB.mousey * 0.0005)
  if player.pitch < -1:
    player.pitch = -1
  if player.pitch > 1:
```

```
    player.pitch = 1
upndown.setDRot(player.pitch, 0.0, 0.0, 1)
GameLogic.addActiveActuator (upndown, 1)
```

Bei der Funktion `select()` ist die Vorgehensweise ähnlich der bei `look()`. Der eingeblendete Mauszeiger wird allerdings durch die Maus gesteuert. Erst wenn der Mauszeiger den Bildschirmrand berührt, verändert sich die Position des Spielers. Außerdem ist eine Vor- oder Seitwärtsbewegung durch die Pfeiltasten in diesem Modus nicht möglich.

Das Gehen des Spielers wird durch die Klassen `walk` und `move` gesteuert. Wird die Klasse walk aufgerufen, so startet zunächst einmal `move` und initialisiert die nötigen Sensoren, um so die Bewegung abfragen zu können. Anschließend wird die Orientierung des Spielers im Raum bestimmt und als Richtungsvektor an `walk` zurückgegeben. Hier wird die Orientierung des Spielers verarbeitet und an den für die Bewegung verantwortlichen *Motion-Actuator* weitergeleitet, der dann den Spieler in die entsprechende Richtung bewegt.

3.3.2 Ablauf des Spiels

Der Spielablauf der Virtual Reality-Anwendung entspricht von seiner Ausrichtung dem normalen Ablauf eines Besuches in der Tempelanlage Gyeongbok Palace. Das bedeutet, dass der Spieler sich zum einen frei auf dem gesamten Palastgelände bewegen kann und sich zum anderen an bestimmten Stellen mit zusätzlichen Informationen versorgen kann. Dabei werden weder Einschränkungen in Bezug auf die Wahl der verwendeten Wege auf dem Plastgelände noch bei der Nutzung von verschiedenen Informationsquellen vorgenommen. Es bleibt also dem Anwender überlassen, ob er die ihm angebotenen Informationen und das damit verbundene Spiel annimmt, oder ob er sich ganz auf die optischen Reize bei der Begehung der Landschaft beschränkt.

Entscheidet sich der Nutzer das vom Avatar angebotene Spiel anzunehmen, so erhält er auf diesem Weg zusätzliche Informationen zur Geschichte und Entwicklung des koreanischen Alphabetes. Die Avatare des Spiels erscheinen auf neun Bühnen, die sich über das gesamte Areal des Palastes verteilen (siehe Abb. 27). Auf Grund der im nachfolgenden Kapitel beschriebenen Story-Engine ist der Nutzer nicht an eine bestimmte Reihenfolge gebunden, sondern kann die Bühnen in beliebiger Reihenfolge aufsuchen. Um die Avatare sehen zu können, ist es notwendig, dass der Spieler die virtuelle Brille aktiviert (`G`-Taste).

Ist die Brille aktiviert und befindet sich der Nutzer auf einer Bühne, so erscheint der Avatar und durch Klicken der linken Maustaste gelangt der Anwender in den `Select-Modus`. Nun ist es möglich, den Avatar mit dem Mauszeiger anzuklicken und somit die Unterhaltung zu starten. Der Avatar verbeugt sich und beginnt mit dem Anwender zu sprechen. Nach dieser Einführung, in der dem Spieler verschiedene historische Informationen gegeben werden, wird ihm eine erste Frage gestellt. Auf Grund der zuvor vermittelten Informationen muss der Spieler entscheiden, ob er dem Avatar vertraut oder nicht. Misstraut er dem Avatar, weil dieser ihm falsche Informationen gegeben hat, so kann er den Avatar demaskieren. Dies ist jedoch nur möglich, wenn der Spieler mindestens zwanzig Punkte auf seinem Konto hat. Ist dies nicht der Fall, kann er entweder die Unterhaltung beenden oder aber dem Avatar trotzdem vertrauen. Sind genügend Punkte vorhanden, so werden die zwanzig Punkte abgezogen und der Avatar offenbart seinen wahren Charakter. Entpuppt er sich als Feind, so löst der Avatar sich in Rauch auf und verschwindet. Ist er jedoch ein Freund, so beklagt er sich über das fehlende Vertrauen des Spielers und beendet die Unterhaltung. Anschließend verschwindet auch er.

Sollte der Spieler dem Avatar jedoch vertrauen, so stellt dieser ihm eine Frage und gibt dazu zwei mögliche Antworten vor. Sollte sich der Spieler nicht sicher sein, welche Antwort die richtige ist, so kann er die Hilfe auswählen, um weitere Informationen zu erhalten, die ihm dann die Beantwortung der Frage ermöglichen sollen. Hat der Spieler sich für eine Antwort

entschieden, so hängt der weitere Verlauf davon ab, ob er es wie zuvor vermutet mit einem Freund oder aber doch mit dessen Gegenspieler zu tun hat. Wurde die Frage von einem Freund gestellt, so erhält der Spieler bei richtiger Beantwortung der Frage einen Buchstaben sowie weitere zwanzig Punkte für sein Demaskierungs-Konto. Sollte er die Frage falsch beantwortet haben, so teilt ihm der Avatar die richtige Lösung mit. Kam die gestellte Frage jedoch von dem feindlichen Avatar, so erhält der Spieler bei der korrekten Antwort zwar die zwanzig Punkte gutgeschrieben, er verliert allerdings einen der bereits gesammelten Buchstaben an den Avatar. Sollte die Antwort der Frage falsch gewesen sein, so erfährt der Spieler die richtige Antwort auf die Frage und verliert ebenfalls einen der gesammelten Buchstaben. In allen Fällen endet anschließend die Unterhaltung mit dem Avatar.

Abb. 27: Übersicht der Palastanlage mit den verschiedenen Bühnen

Der Spieler kann sich dann zu einer anderen Bühne begeben und dort nach dem gleichen Schema mit dem Avatar kommunizieren. Auch das mehrfache Besuchen einer Bühne durch den Spieler ist möglich. Er wird jedoch nicht immer auf derselben Bühne einen Freund beziehungsweise einen Feind treffen, da die Gesinnung des Avatars immer wieder neu durch eine Zufallsfunktion bestimmt wird. Der Spieler kann sich solange frei in der Tempelanlage bewegen und die verschiedenen Bühnen aufsuchen, wie das Spiel andauert. Erreicht die Story-Engine jedoch das Ende der Geschichte, so wird beim nächsten Besuch einer Bühne das Ende des Spiels durch die Einblendung eines Abschlussbildes angezeigt. Der Spieler kann dann das Spiel durch drücken der `ESC`-Taste verlassen. Der hier beschriebene Ablauf des Spiels wurde zum besseren Verständnis auch in einem Ablaufdiagramm (siehe Abb. 28) dargestellt.

Die Steuerung des beschriebenen Spielablaufes ist eine der Aufgaben, die von dem Python-Skript `sePerson.py` durchgeführt werden. Das Skript unterscheidet dazu zwischen verschiedenen Zuständen, in denen sich das Spiel befinden kann. Die genaue Funktionsweise des Skriptes kann im Anhang 8.1 aus dem Quellcode entnommen werden.

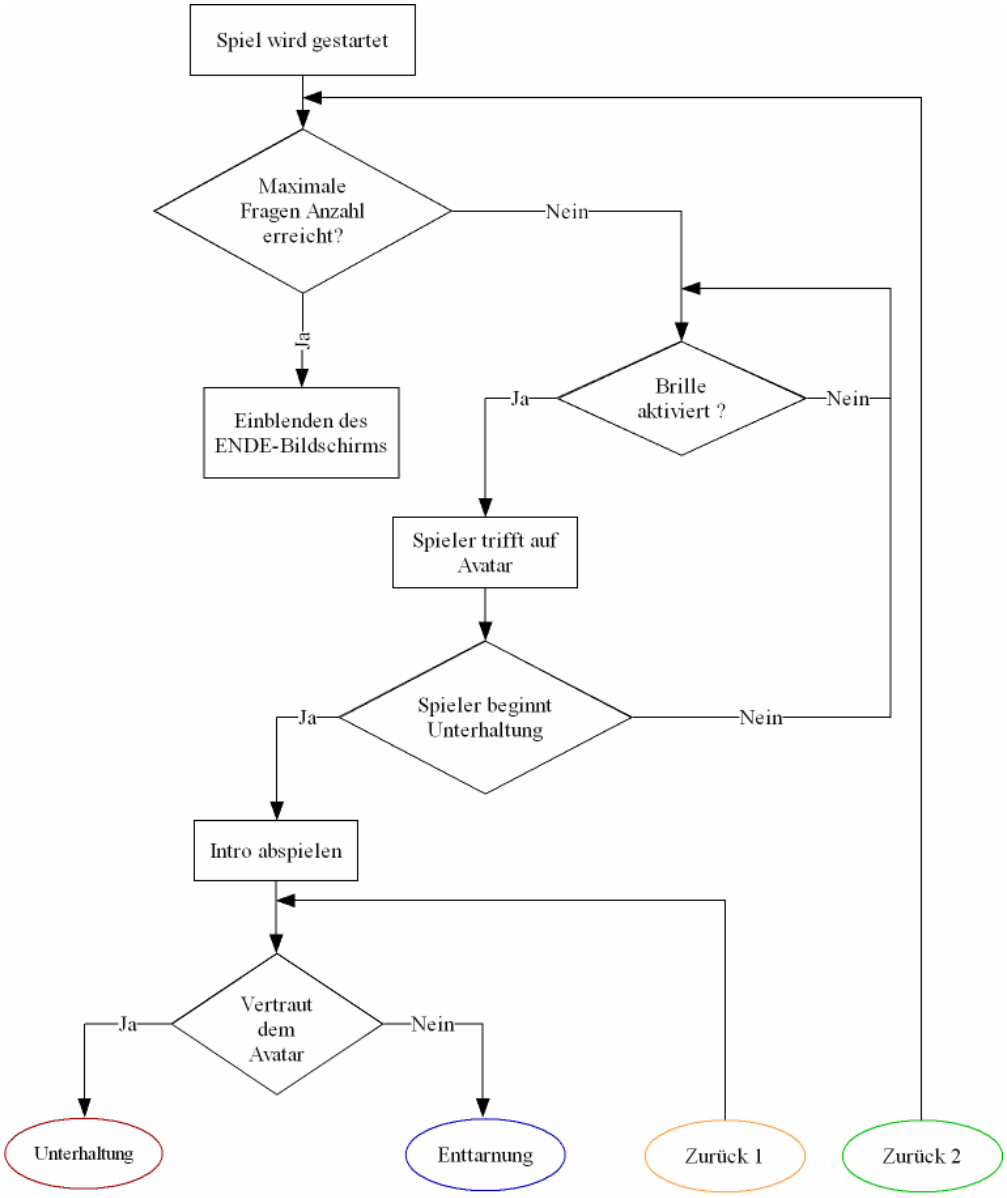

Abb. 28a: Ablaufdiagramm des Spieles

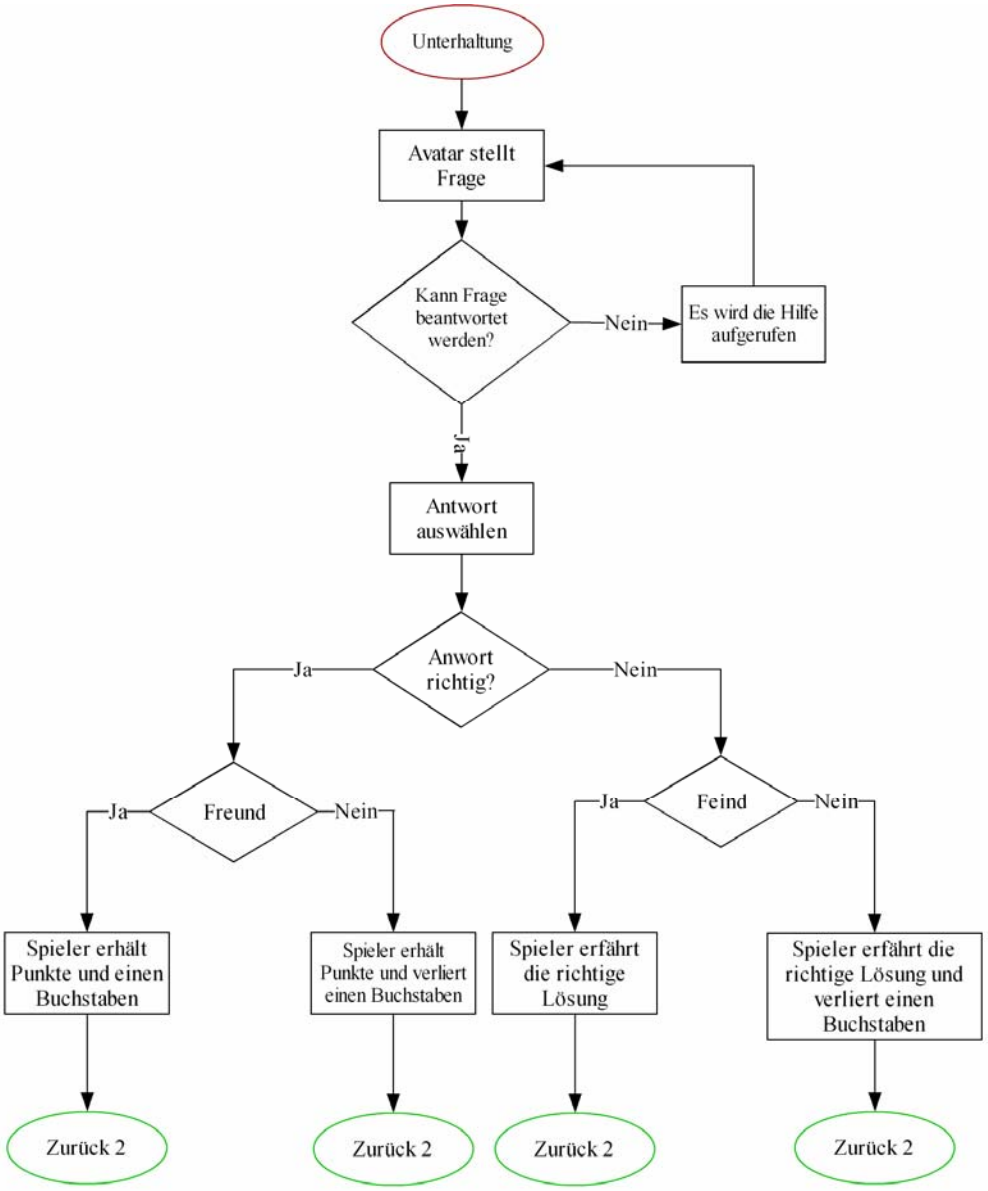

Abb. 28b: Ablaufdiagramm des Spieles

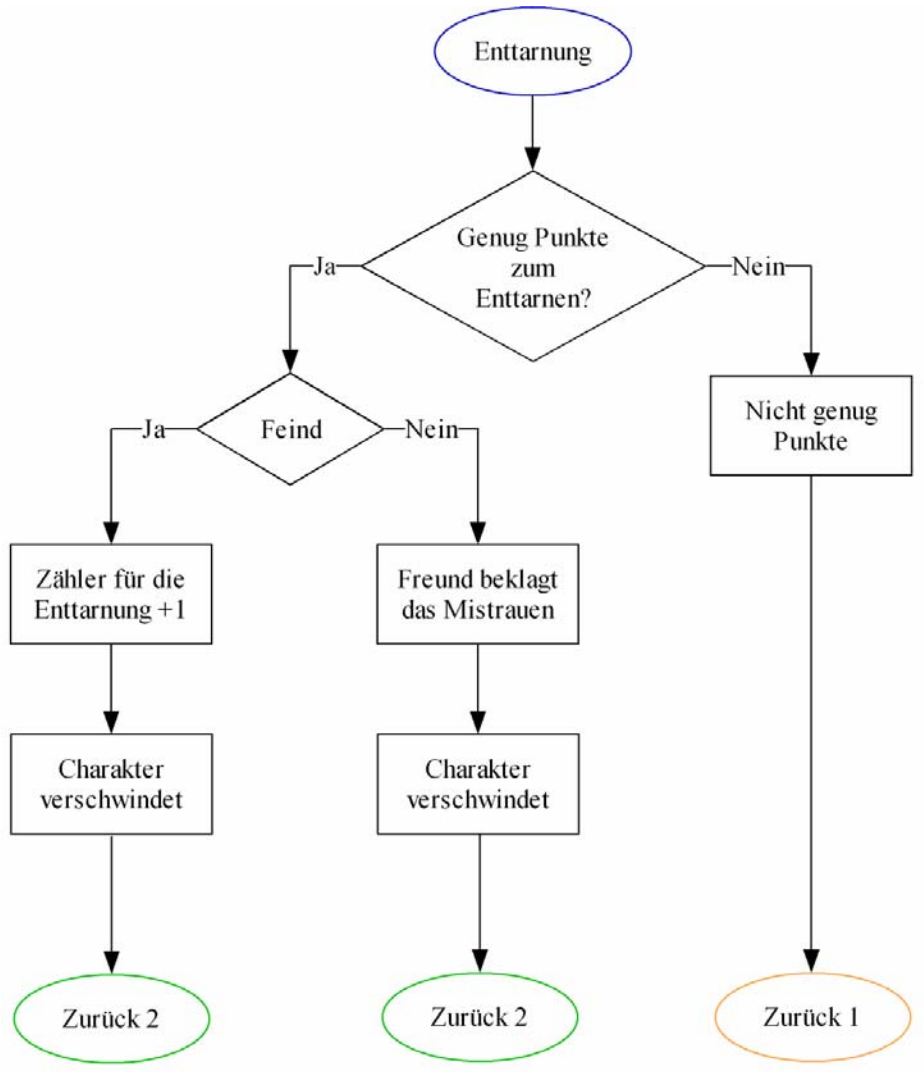

Abb. 28c: Ablaufdiagramm des Spieles

3.3.3 Integration der Story-Engine

„Die Story-Engine ist ein Suchalgorithmus der anhand von Bedingungen (Kontexte) auf Grundlage der vorhergehenden Szene eine neue Szene berechnet und die entsprechende Szenennummer zurückgibt. Die Daten für die Suche entnimmt sie zwei XML-Dateien. Einmal gibt es eine Datei, welche das Storymodell enthält und eine zweite, die suspenseML, welche die Szenennummern mit den entsprechenden Bedingungen enthält. Das Storymodell ist ein Ablaufplan dafür, welche Szenentypen nacheinander abgespielt werden sollen. Außerdem können noch kontextgesteuerte Schleifen enthalten sein. Die suspenseML enthält parallel zu den Szenen im Storymodell zu jedem Szenentyp eine oder mehrere Szenen (genau genommen muss nicht zu jedem Szenentyp unbedingt eine Szene existieren, solange dieser Typ nicht gespielt werden muss). Diese enthalten Bedingungen, welche erfüllt sein müssen, um diese Szene abspielen zu können (needed) und Bedingungen, die gesetzt werden, wenn diese Szene abgespielt wird (set). Jede Story besitzt eine globale Kontextliste, in der die aktuellen Bedingungen, die für die Story gelten, gespeichert sind (sie ist also sozusagen das Gedächtnis). Ist die Story-Engine bei der letzten Szene angelangt, gibt sie statt der Szenennummer ein 'finished' zurück“ [Schaar03b].

Die Kommunikation des Spiels mit der Story-Engine erfolgt durch das Python-Skript sePerson.py. Der Datenaustausch zwischen den Programmen findet über eine Socket-Verbindung statt. Zu diesem Zweck wurde im Skript die Funktion `storyEngine()` definiert. Sie baut eine Socket-Verbindung zu einem vorher festgelegten Port auf. Damit auch die vor dem Spiel gestartete Story-Engine auf diesen Port zugreifen kann, wurde die Portnummer in einer externen Textdatei gespeichert. Das Starten der Story-Engine erfolgt durch den Aufruf von `import seStory.py` in der Python-Konsole des Betriebssystems. Über die Socket-Verbindung gibt das Spiel die Entscheidungen des Anwenders an die Story-Engine, die diese dann verarbeitet und die Reaktion darauf an das Spiel zurückgibt.

```
def storyEngine (friend, loc, seSecondCall,
                 gameFinished)
  import time
  seConditions=(str(friend)+'*'+str(loc)+'*'+
                str(seSecondCall)+'*'+
                str(gameFinished))
  HOST = 'localhost'
  file = open(portPathSE, 'r')
  PORT = int(file.readline())
  file.close()
  s = socket.socket(socket.AF_INET,
                    socket.SOCK_STREAM)
  s.connect((HOST, PORT))
  s.send(seConditions)
  data = s.recv (1024)
  ...
```

Die von der Story-Engine zurückgegebenen und in `data` gespeicherten Daten werden im Skript weitergeleitet. Dies geschieht jedoch nur, wenn die Story-Engine kein Ende-Signal übermittelt. Wird ein Ende-Signal übermittelt, so wird dies direkt in der Funktion verarbeitet und der `modecontroller.gameFinished` auf den entsprechenden Wert gesetzt. Auf dem Bildschirm wird in diesem Fall ein Bild eingeblendet, das dem Anwender das Ende des Spiels anzeigt. Gleichzeitig werden alle anderen Funktionen gesperrt, damit der Anwender das Spiel nur noch beenden kann.

```
  ...
  elif data == 'finished':
    print "finished-EndOfStoryEngineData"
    modeController.gameFinished = 3

  else:
    return int(data)
```

3.3.4 Soundausgabe

Die Soundausgabe des Spiels erfolgt aus Python heraus. Dies ist notwendig, da die interne Funktion, die Blender zur Verfügung stellt, nur für die Ausgabe von vorher festgelegten Audiofiles benutzt werden kann. Da bei EduTeCH die Soundfiles aber erst zur Laufzeit durch die Story-Engine bestimmt werden, kann im Vorfeld keine Reihenfolge festgelegt werden, in der sie abgespielt werden. Auf Grund der Tatsache, dass die Sound-Actuators lediglich Soundfiles und keine Skripte unterstützen, kann das entsprechende Soundfile auch nicht an den Actuator übergeben werden. Aus diesen Gründen erfolgt die Soundausgabe über Python.

Zwar besitzt Python Module zur Soundausgabe, allerdings ist lediglich das Windowsmodul gut einsetzbar. Um das Spiel auch unter Windows mit Sound spielen zu können, muss dementsprechend eine Überprüfung des Betriebssystems durchgeführt werden. Wird eine Windowsversion als solche erkannt, so werden die Soundfiles über das Modul `winsound` ausgegeben. Soll die Ausgabe gestoppt werden, so wird ein leeres Wave-File an das Modul übergeben.

```
def playSound (soundFile)
  if (os.name == “nt“ or os.name == “dos“ or os.name
      == “cw“ or):
    import winsound
    winsound.PlaySound(soundFile, 1)

def stopSound():
  if (os.name == “nt“ or os.name == “dos“ or os.name
      == “cw“ or):
    import winsound
    winsound.PlaySound(soundPathLeer, 1)
```

Die Soundausgabe unter Linux ist im Gegensatz dazu recht schwierig. Es gibt zwar auch hier ein Soundmodul, allerdings lässt sich bei `ossaudiodev` die Soundausgabe nicht abbrechen und das Spiel ist blockiert, bis das

gesamte Wave-File abgespielt ist. Weiterer Nachteil dieser Methode ist, dass das Audiomodul erst ab der Version 2.3 in Python integriert ist und somit nicht mit der alten Game-Engine in Blender 2.25 funktioniert, da dort die Pythonversion 2.0 verwendet wird. Ein anderer Ansatz war die Ausgabe der Soundfiles über ein externes Soundprogramm, welches die Daten über eine Socket-Verbindung geschickt bekam. Nachteil dieser Version ist, dass vor dem Spielstart ein Programm geöffnet und mit einem bestimmten Port verbunden werden musste. Da dies nicht von jedem Nutzer erwartet werden kann, wurde dieser Ansatz wieder verworfen. Da Python jedoch auch die Möglichkeit besitzt, externe Programme zu starten und diese auch zu steuern, wurde es möglich, die Soundfiles durch das KDE-Programm *noatun* auszugeben. Dazu wurde das Programm durch den Befehl `os.popen ("noatun")` aus Python geöffnet und mit den notwendigen Daten versehen. Einziger Nachteil dieser Lösung ist, dass der KDE-Desktop unter Linux zwingend erforderlich ist, da ansonsten das erforderliche Programm nicht existiert.

```
else:
  if commands.getoutput("dcop | grep noatun") == "":
    os.popen ("noatun")
  noatune = commands.getoutput("dcop | grep noatun")
  commands.getoutput("dcop "+ noatune +
                     "noatun addFile"+soundFile +"1")
```

3.4 Zusammenfassung

Dieser Abschnitt der Diplomarbeit befasste sich mit der Entwicklung der Virtual Reality Anwendung für das Projekt EduTeCH. Dabei wurde zunächst auf die vorher definierte Zielsetzung dieses Bereiches eingegangen, um dann die Modellierung der virtuellen Umgebung und der Avatare zu beschreiben. Dabei wurde auch auf die Texturierung der Modelle und die Animation der Avatare eingegangen und deren Bedeutung für den realistischen Gesamteindruck dargestellt. Im Anschluss daran wurde die für das Spiel notwendige Game-Logic erklärt und die sich daraus ergebenden Arbeitsschritte vorgestellt. Dazu gehören die Steuerung des Spielers, der Spielablauf, die Integration der Story-Engine und die Soundausgabe. Nach Abschluss dieser Arbeiten liegt ein lauffähiges Spiel vor.

Nachdem in diesem Kapitel die entwickelte Virtual Reality Anwendung erläutert wurde, befasst sich der nun folgende Abschnitt mit der daraus erstellten Augmented Reality Anwendung.

4. Anforderungen an die AR Anwendung

4.1 Zielsetzung der AR Anwendung

Die Zielsetzung für die Augmented Reality Anwendung ist es, das in Kapitel 4 beschriebene Virtual Reality System in ein mobil nutzbares Augmented Reality System umzuwandeln. Dabei werden neben einer Erweiterung der Anwendung auch Anpassungen des bestehenden Systems an die neu definierten Anforderungen nötig.

Die folgenden Abschnitte befassen sich folglich mit notwendigen Abänderungen, die an der VR-Anwendung durchgeführt werden müssen. Außerdem wird beschrieben, was notwendig ist, um die erforderlichen Location Based Service Geräte einzubinden und effektiv nutzen zu können.

4.2 Anpassungen in der VR-Anwendung

Damit eine funktionierende AR-Anwendung aus dem VR-Spiel wird sind diverse Änderungen in verschieden Bereichen notwendig (siehe Abb. 30 und 31). Zum einen befindet sich der Anwender bei der AR-Anwendung im realen Palastumfeld, weshalb auf die Anzeige der real vorhandenen Bausubstanz verzichtet werden kann. Zum anderen muss die Steuerung des Spielers verändert werden, da die AR-Version nicht mehr über Tastatur und Maus sondern lediglich über zwei Tasten verfügt, die auf der oberen Seite der AR-Brille liegen (siehe Abb. 29).

Die in der AR-Anwendung dargestellten Objekte beschränken sich auf die in der Realität nicht vorhandenen Gebäude sowie die zum Spiel notwendigen Avatare. Das bedeutet, dass der größte Teil der VR-Anwendung ausgeblendet werden kann, da der Nutzer diese auch durch sein Head-Mounted Display sehen kann. Lediglich die Avatare werden eingeblendet, sobald sich der Spieler ihnen nähert. Dabei ist besonders darauf zu achten, dass die

Überblendung der Objekte exakt erfolgt, damit der Avatar weder im Boden versinkt noch in der Luft schwebt. Alle nicht benötigten Objekte werden als invisible gesetzt und fallen somit aus der Berechnung heraus, was die Datenmengen erheblich reduziert. Diese Änderungen sind jedoch im Gegensatz zur Steuerung relativ problemlos zu realisieren.

Abb. 29: Tasten zur Steuerung der AR-Anwendung an der AR-Brille

Bei der Steuerung des Spiels gibt es hingegen eine Menge an Problemen zu lösen, da hier die Möglichkeiten deutlich beschränkter sind als im VR-Spiel, wo neben der Maus mit drei Tasten und zwei Bewegungsachsen auch noch die Tastatur mit über 100 Tasten zur Verfügung steht. Deshalb werden insbesondere bei der Steuerung des Spielers andere Geräte benötigt, die eine Steuerung ermöglichen, ohne dass der Anwender selbst etwas dafür tun muss. Die Blickrichtung wurde in der VR-Anwendung durch die Maus gesteuert. Da diese nun nicht mehr vorhanden ist, übernimmt der Head-Tracker diese Aufgabe. Wie dieses Gerät eingebunden wird und wie es funktioniert wird im Kapitel 4.3.3 genauer erklärt.

Bleibt noch das Problem der Bewegung des Spielers, die in der VR-Version durch die Pfeiltasten der Tastatur gesteuert wurde. Die Bewegung

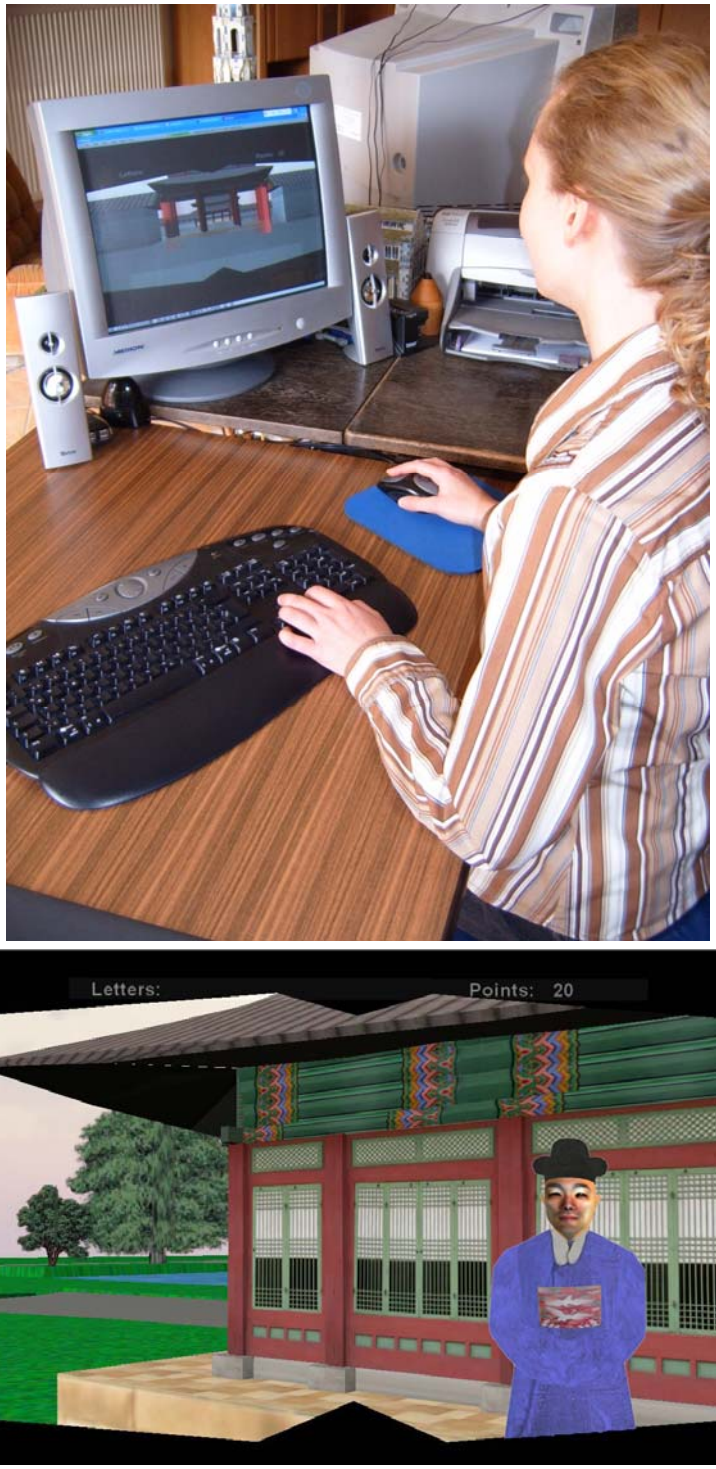

Abb.30: Equipment für die VR-Anwendung (oben) und Ansicht des Spielers (unten)

Abb.31: Equipment für die AR-Anwendung [Wohlfahrt04] (oben)
und Ansicht des Spielers (unten)

des Anwenders entspricht in der AR-Version denen des virtuellen Spielers, weshalb es nahe liegend war, diese mit Hilfe eines GPS-Systems auf das Spiel zu übertragen. Dazu müssen dem virtuellen Spielfeld dieselben Koordinaten zugewiesen, die das reale Palastgelände hat. Dadurch kann dann immer die exakte Position auf dem Spielfeld bestimmt werden (genauere Erläuterungen hierzu folgen in Kapitel 4.3.4).

Die virtuelle Brille wurde in der VR-Anwendung durch die Taste ’G’ aktiviert. Dies kann bei AR entfallen, da lediglich virtuelle Elemente angezeigt werden und die Brille durch den Anwender auf- oder abgesetzt werden kann. Für das Aktivieren des Auswahlmodus sowie die Auswahl der einzelnen Antworten wurde bislang die linke Maustaste verwendet. In der AR-Anwendung wird dies durch eine der drei an der Brille angebrachten Tasten übernommen. Dies gilt auch für das Verlassen des Auswahlmodus, der durch die rechte Maustaste gesteuert wurde.

Nach diesem Überblick über die Unterschiede zwischen dem VR-Spiel und der AR-Anwendung und den daraus resultierenden Anpassungen sollen die folgenden Abschnitte die notwendigen Geräte vorstellen und deren Eindung in das Projekt verdeutlichen.

4.3 Einbindung der Location Based Services Geräte

4.3.1 Überblick über die Geräte

Um eine AR-Anwendung ansprechend realisieren zu können, sind diverse Location Based Services Geräte notwendig. Dabei ist vor allem darauf zu achten, dass die eingesetzten Geräte möglichst klein und leicht sein sollten, um die Bewegungsfreiheit des Anwenders nicht unnötig einzuschränken. Des Weiteren müssen die Geräte eine möglichst hohe Messgenauigkeit haben, damit die AR-Anwendung nicht durch falsche Daten beeinträchtigt wird. Da für dieses Projekt lediglich auf bereits vorhandene Hardware zurückgegriffen werden konnte, befinden sich nicht mehr alle eingesetzten Geräte auf dem neusten Stand der Technik. Da sie jedoch alle für ein ähnliches AR-System angeschafft wurden, erfüllen sie die zuvor definierten Ansprüche und können bedenkenlos eingesetzt werden.

Das eingesetzte Head-Mounted Display kann mit einem Fernglas verglichen werden. Der Anwender hält es in der Hand und führt es bei Bedarf vor die Augen. Auf Grund der halbtransparenten Bauweise sieht er dann sowohl die virtuelle Projektion als auch die reale Welt. Diese Methode ist zwar im Vergleich zu den modernen Head-Mounted Displays, die den Nutzer eher an eine Sonnenbrille erinnern, recht umständlich, weil der Anwender immer etwas in der Hand halten muss. Trotzdem ist die Methode gut geeignet.

Der verwendete Tracker ist sehr klein und verfügt über eine schnelle Datenübermittlung bei hoher Abtastrate, wodurch auch schnelle Bewegungen gut erfasst werden können. Lediglich die bei Gyroskopen auftretende Drift führt bei längerer Anwendung zu einem Messfehler. Dieser kann jedoch durch Drücken der Kalibrierungstaste reduziert werden, da dann die Ausgangswerte des Trackers wieder hergestellt werden. Um die Blickrichtung des Anwenders jederzeit korrekt ermitteln zu können, wird der Tracker auf dem Head-Mounted Display befestigt.

Durch die Verwendung eines GPS-Empfängers soll die exakte Positionsbestimmung des Anwenders im Raum ermöglicht werden. Um immer eine optimale Verbindung zu den Satelliten zu gewährleisten, wird der GPS-Empfänger an einer außen liegenden Stange des Rucksacks befestigt. Durch Anpeilen von mindestens drei Satelliten kann dann die exakte Position errechnet werden und damit die Position des Spielers in der AR-Anwendung angepasst werden.

4.3.2 Einbindung des Head-Mounted Displays

Das in dieser Arbeit eingesetzte Gerät ist streng genommen kein Head-Mounted Display, sondern gehört in die Gruppe der AR-Glasses. Unterschied zwischen diesen beiden Kategorien ist, dass ein Head-Mounted Display am Kopf des Anwenders befestigt ist, während die AR-Glasses vom Nutzer in der Hand gehalten werden und wie ein Fernglas vor die Augen gehalten werden. Für dieses Projekt wurde das in der Abteilung vorhandene Sony Glasstron verwendet (Abb. 32).

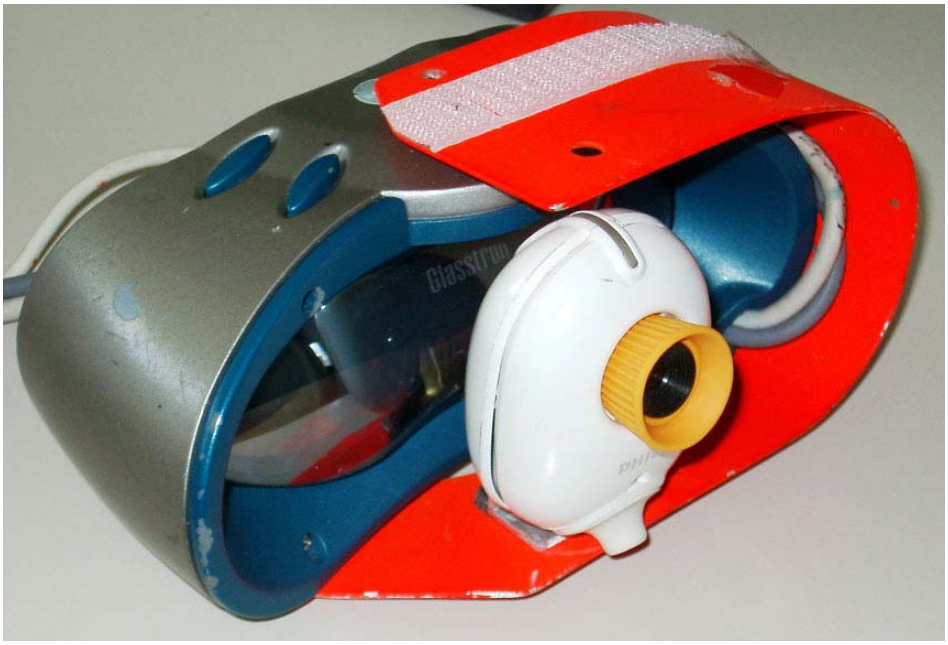

Abb. 32: Sony Glasstron mit Halterung für den Tracker und zwei Tasten zur Interaktion

Das Sony Glasstron verfügt über eine Auflösung von maximal 800x600 Pixel und wird über ein externes Steuergerät mit dem PC verbunden (Abb. 33). Als Anschlussmöglichkeiten steht ein RGB- bzw. ein S-Video-Eingang zur Verfügung. Zusätzlich zum Videosignal können auch die Audiodaten an das Gerät übertragen werden. Da die Sony Glasstron über ein integriertes Steuermenü verfügt, ist die Anpassung der Bilddaten an die Brille einfach zu realisieren. Es ist dabei unter anderem möglich, Grundeinstellungen wie die Helligkeit, den Kontrast oder die Lautstärke des Audiosignals zu regulieren. Zusätzlich sind jedoch auch Korrekturen der Bildposition oder eine Reduktion der Lichtdurchlässigkeit der Brille möglich. Dadurch wird es ermöglicht, dass der Anwender auf die vorliegenden Wetterverhältnisse reagieren kann, indem er bei starker Sonneneinstrahlung die Lichtdurchlässig-

Abb. 33: Steuergerät der AR-Glasses

keit verringert oder bei Bewölkung erhöht. Die Stromversorgung des Displays wird durch einen am Steuergerät angebrachten Akku sichergestellt. Für die in der Anwendung notwendige Interaktion verfügt das Sony Glasstron über zwei Tasten, die vom System wie Maustasten behandelt werden.

Der Anschluss des Gerätes ist unproblematisch. Es muss lediglich darauf geachtet werden, dass die eingestellte Auflösung nicht größer als 800x600 ist, da größere Auflösungen nicht mehr dargestellt werden können. Sollten anschließend noch Anpassungen notwendig sein, so können diese direkt über das Steuergerät der Brille vorgenommen werden.

4.3.3 Einbindung des Trackers

Die Einbindung des Trackers in das System ist von zentraler Bedeutung, da erst durch dieses Gerät die richtige Ausrichtung der Kamera in der Augmented Reality Anwendung gewährleistet wird. Der Tracker ist eine unverzichtbare Voraussetzung für eine funktionierende und vor allem ansprechende AR-Anwendung.

Der in diesem Projekt verwendete Tracker 'InterTrax^2' der Firma InterSense (www.isense.com) ist laut Firmenangaben der kleinste high Performance Head-Tracker der Welt (Abb. 34). Er wird über eine USB-Schnittstelle an den Rechner angeschlossen und ist ein 3DOF-Sensor (**D**egree **O**f **F**reedom), der die Rotation um die drei Raumachsen mit Gyroskopen erfasst. Da dieser Sensor lediglich drei der für einen AR-Anwendung erforderlichen sechs Freiheitsgrade bestimmt, muss es durch ein weiteres System ergänzt werden, dass die Translation des Anwenders überwacht und somit die genaue Positionierung im Weltkoordinatensystem ermöglicht. In diesem Projekt wird dies durch ein GPS-System erfolgen. „Externe Geräte für das Bezugssystem sind nicht nötig. Ein Gyroskop misst die Rotationsbeschleunigung um eine Achse, durch die Kombination von drei orthogonal zueinander angeordneten Gyroskopen werden alle drei Raumachsen erfasst“

[Evers01]. Da diese Sensoren in Chip-Technologie hergestellt werden können, ist mit dieser Technik die Herstellung besonders kompakter Geräte möglich.

Abb. 34: Head-Tracker der Firma InterSense (Länge: 7cm)

Um die Daten des Tracker verwenden zu können, ist eine Anpassung der Treiber an das verwendete Betriebssystem notwendig. Unter Linux wird das Trackermodul durch den Befehl `modprobe itrax` in das System integriert. Damit auch auf die vom Tracker gelieferten Daten zugegriffen werden kann, ist eine Anpassung der Datei `tracker.c` notwendig. Aus ihr wird anschließend das Modul erstellt (`tracker.so`), über das Python die Daten des Trackers abfragt. Durch die Auswertung dieser Daten wird dann die Steuerung des Charakters ermöglicht. Die Anpassungen in der `tracker.c` -Datei ermöglichen es, in einem Pythonskript die notwendigen C-Funktionen anzusprechen und so die Daten des Trackers auszulesen. Damit dies möglich ist, muss daraus ein Python-Modul entstehen, das in C geschrieben ist. Damit der Pythoninterpreter das Modul erkennt, wird das Prinzip der Shared Library bzw. Shared Object verwendet. „Das in C geschriebene Modul wird in eine nachladbare Bibliothek übersetzt und in ein Verzeichnis gestellt, das in `sys.path` enthalten ist. Python sucht beim Import eines Moduls auch immer nach einer nachladbaren Bibliothek mit dem Namen des Moduls. Wird diese Bibliothek gefunden, wird sie initialisiert und im aktuellen Namensraum zur Verfügung gestellt“ [Loewis01].

Das Modul kann aus einer oder mehreren Dateien bestehen und besitzt eine festgelegte Struktur. Das bedeutet, dass jedes Modul bestimmte Funktionen enthält, die notwendig sind, um den Quellcode richtig interpretieren zu können. Durch den Aufruf von `#include <Python.h>` werden dem Modul die Prototyp-Definitionen aller durch Python zur Verfügung gestellten Funktionen bekannt gemacht. Außerdem muss eine Funktion definiert werden, die das Modul initialisiert. „Diese Funktion hat einen festgelegten Namen, der aus dem Namen des Moduls mit dem Präfix `init` gebildet wird. Für das Modul `tracker` muss also die Funktion `inittracker` definiert werden. Eine Funktion dieses Namens wird vom Pythoninterpreter gerufen, wenn das erste Mal `import tracker` ausgeführt wird. Diese Funktion erwartet keine Argumente und hat als Rückgabewert `void`“ [Loewis01]. Durch Aufruf der Funktion `Py_InitModule` wird das Modul in der Funktion bekannt gemacht. Die Funktion erwartet dabei folgende Argumente:

- den Namen des Moduls als String und
- ein Feld mit Elementen des Typs `PyMethodDef`.

Im zweiten Argument steht die Definition aller Funktionen, die im Modul enthalten sein sollen. Um eine Funktion zu definieren, sind folgende Informationen notwendig:

- Wie heißt die Funktion?
- Welche C-Funktion soll durch diese Funktion gerufen werden?

Die aufzurufende C-Funktion muss dabei der folgenden Signatur entsprechen [Loewis01]:

```
static PyObject * Name(PyObject* self, PyObject* args)
```

Daraus ergibt sich folgende Grundstruktur für das zu erstellende Python-Modul (das gesamte Modul kann in Kapitel 9, Anhang A nachgeschlagen werden):

```
…
#include <Python.h>
…
static PyObject * open(PyObject* self, PyObject* args)
{
   return Py_None;
}
…
static PyObject * close(PyObject* self, PyObject* args)
{
   return Py_None;
}
…
static PyMethodDef moduleMethods[]=
{
   {“open_tracker“, open},
   …
   {“close_tracker“, close},
   {NULL, NULL}
};

void inittracker()
{
   (void) Py_InitModule(“tracker“, moduleMethods);
}
```

Da der Tracker aus Blender heraus nur einmal geöffnet wird, gibt die Funktion `open` die Prozessnummer zurück, die dann bei den Abfragen der Werte für X-, Y- und Z-Achse wieder an das Modul übergeben wird. Die Abfrage der Daten erfolgt durch eine Funktion mit folgendem Aufbau:

```
static PyObject * getX(PyObject* self, PyObject* args)
{
   int device;
   PyArg_ParseTuple(args,“i“,&device);
   read(device, &data, sizeof(struct trackerposition));
   return PyInt_FromLong((long)data.tenthdegree[0]/10.0);
}
```

Der Tracker liefert für die Drehung um die Z-Achse Werte zwischen 0° und 360°, was dementsprechend einer Drehung des Spielers um die eigene Achse entspricht. Dem Heben bzw. Senken des Kopfes entsprechen die Tracker Werte für die Y-Achse. Dieser beträgt beim Geradeaus-Schauen 0° und verändert sich beim Nach-oben-Sehen bis auf maximal 180° und beim Nach-unten-Blicken auf bis zu -180°. Analog dazu verhält sich das Schräghalten des Kopfes, wobei das Abkippen nach rechts den positiven, das Abkippen nach links den negativen Wertebereich widerspiegelt (jeweils bis zu 180°). Ist das Ende des Spiels ereicht, so wird auch der Tracker durch den `close`-Aufruf geschlossen.

Nachdem sowohl die Funktionsweise als auch die Integration des Trackers beschrieben wurde, fehlt noch die exakte Bestimmung der Position des Spielers im Raum, die durch den im folgenden Abschnitt beschriebenen GPS-Empfänger gewährleistet werden soll.

4.3.4 Einbindung des GPS-Empfängers

Zu Beginn des Projektes war für die Positionsbestimmung des Anwenders in der AR-Anwendung der Einsatz eines GPS-Systems geplant (siehe Abb. 35). Auf Grund der in den letzten Jahren verübten Terroranschläge hat sich die Koreanische Regierung dazu entschlossen, die Palastanlage zur Sicherheitszone zu erklären. Das bedeutet, dass das Mitführen der für die Satellitenortung notwendigen GPS-Empfänger nicht länger gestattet ist. Somit ist der Einsatz dieser Technik für dieses Projekt nicht mehr realisierbar und wurde von der Projektleitung ersatzlos gestrichen. Da die Umrechnung der Satellitendaten auf die Erdkoordinaten jedoch ein sehr komplexes Verfahren ist und die dafür vorgesehene Unterstützung anderer Abteilungen nicht mehr in Anspruch genommen werden konnte, würde die Integration der GPS-Ortung den Rahmen dieser Diplomarbeit bei weitem überschreiten. Aus diesem Grund soll lediglich beschrieben werden, was notwendig wäre, um die Satellitenortung noch nachträglich zu integrieren.

Die Vorgehensweise bei der Integration eines GPS-Empfängers ist mit der bei der Einbindung des Trackers zur Blickrichtungsbestimmung vergleichbar. Den ersten Schritt stellt die Anpassung der Treiber auf das verwendete Betriebssystem sowie die verwendete Software dar. Dabei wird zunächst geprüft, welche Treiber bereits vorhanden sind und in wie weit man diese übernehmen kann. Anschließend müssen dann eventuelle Änderungen an den Treiberdateien vorgenommen werden.

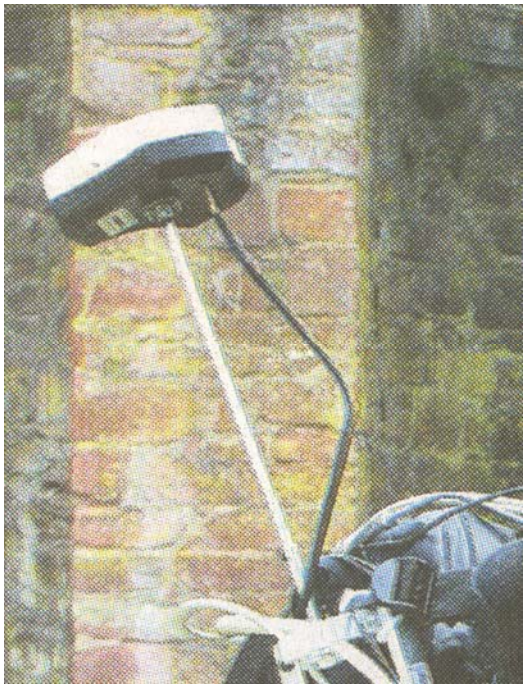

Abb. 35: GPS-Empfänger am Rucksack befestigt [Wohlfahrt04]

Sind diese Vorbereitungen abgeschlossen und kann der GPS-Empfänger vom Betriebssystem angesprochen werden, so müssen die zurückgegebenen Daten noch so verarbeitet werden, dass diese auf das Erdkoordinatensystem übertragbar sind. Dabei muss zum Beispiel beachtet werden, dass die in Blender modellierte Landschaft einer flachen Scheibe entspricht. Das reale Palastgelände unterliegt jedoch der Erdkrümmung und somit ergeben sich Messfehler, die durch eine Software herausgerechnet werden müssen.

Damit die Daten auch aus der Game-Engine abgefragt werden können, muss ein Python-Modul programmiert werden, das vom Aufbau dem Modul für die Trackerdaten entspricht und die korrigierten Daten zur Positionsbestimmung im Spiel nutzen kann. Zur Positionsbestimmung des Spielers auf dem virtuellen Palastgelände müssen den Eckpunkten des Gebietes die entsprechenden GPS-Daten zugeordnet werden. Dadurch wird es möglich, den vom GPS-Empfänger gelieferten Datensatz einer bestimmten Stelle auf dem virtuellen Spielfeld zuzuordnen und somit die Position sowie auch die Bewegung des Spielers richtig darzustellen.

Nach der erfolgten Implementierung des GPS-Empfängers wäre es dringend erforderlich, einen Funktionstest im Palastgelände durchzuführen. Dabei müsste überprüft werden, ob es Problemstellen gibt, an denen keine genaue Positionsbestimmung mehr möglich ist. Solche Probleme können sich zum Beispiel in schmalen Bereichen zwischen zwei Gebäuden, in überdachten Durchgängen oder aber den Säulengängen ergeben, da sich hier der sichtbare Himmelsausschnitt bis auf ein Minimum reduzieren wird und somit unter Umständen nicht mehr die erforderliche Anzahl von drei Satelliten zur Verfügung steht, um die Position zu bestimmen. Da sich die Anzahl der maximal verfügbaren Satelliten während eines Tages verändert, sollte auch überprüft werden, ob sich eventuell durch geringe Verfügbarkeit Probleme ergeben können. (siehe Abb. 36).

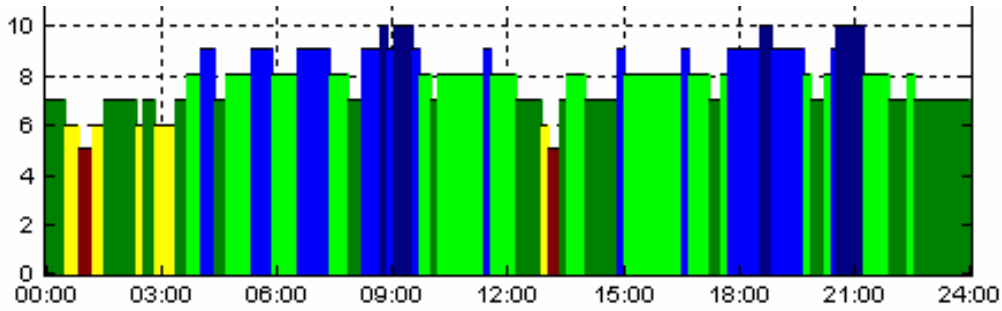

Abb. 36: Anzahl der sichtbaren Satelliten (über 10° über dem Horizont) für München am 23.11.02 (X-Achse: Uhrzeit, Y-Achse: Anzahl der verfügbaren Satelliten)

Sollten sich auch im freien Gelände durch Bäume oder Gebäude Probleme ergeben, so müssten entweder geeignete Verfahren entwickelt werden, um diese Beeinträchtigungen zu minimieren, zum Beispiel durch Berücksichtigung der vorherigen Position und Bewegungsrichtung. Falls dies nicht möglich wäre, müsste auf alternative, eventuell stationäre, Ortungsverfahren zurückgegriffen werden.

Eine endgültige Entscheidung über das zu verwendende Ortungsverfahren müsste bei entsprechenden Funktionstests vor Ort getroffen werden, da die lokale Verfügbarkeit auch an Problemstellen gewährleistet sein muss. Sollte dies nicht der Fall sein, so würde sich dies unter Umständen negativ auf den Spielfluss auswirken, da dann die exakte Position des Spielers nicht mehr bestimmt werden kann.

4.4 Zusammenfassung

In diesem Teil der Arbeit wurde die Weiterentwicklung der erstellten Virtual Reality Anwendung hin zu einer Augmented Reality Anwendung beschrieben. Dazu wurden zunächst der veränderte Ansatz und die sich daraus ergebenden Modifikationen erläutert. Im Anschluss wurden die notwendigen Geräte vorgestellt und deren Einbindung in das System dargestellt. Dabei wurden auch die entstehenden Probleme sowie die möglichen Lösungen beschrieben, wobei größere Probleme lediglich beim GPS auftraten. Diese waren jedoch nicht technisch bedingt, sondern entstanden auf Grund politischer Entscheidungen. Einen Überblick über das verwendete Equipment zeigt Abbildung 37, in der das verwendete Gesamtsystem im Einsatz zu sehen ist.

Abb. 37: AR-Equipment im Einsatz [Wohlfahrt04]

Das folgende Kapitel befasst sich mit der Erstellung von Installationsskripten, die eine einfache Installation unter dem verwendeten Betriebssystem ermöglichen.

5. Erstellen eines Ebuilds für Gentoo

5.1 Definition Ebuild

Unter dem Begriff Ebuild werden bei der Linux-Distribution Gentoo jene Installationsskripte verstanden, die Informationen über jedes einzelne, bei der Installation eines Programms benötigte Programmpaket enthalten. „Sie enthalten zum einen Paketinfos für den Benutzer wie Beschreibung, Homepage und Lizenz der Software und zum anderen Systemanweisungen zum Patchen, Kompilieren, Herunterladen und Installieren“ [GenLib].

5.2 Allgemeine Bestimmungen

Ein Ebuild besteht normalerweise aus fünf Abschnitten, die in jedem Ebuild vorkommen. Der Inhalt der einzelnen Bereiche kann jedoch von Ebuild zu Ebuild stark variieren, weshalb hier auch nur auf die Funktion der Abschnitte und nicht auf deren exakten Aufbau eingegangen wird.

Den ersten Abschnitt in einem Ebuild bildet der Standard-Header. Er beinhaltet Angaben zum Copyright, den Autoren sowie Informationen zur Datei.

Im zweiten Teil des Ebuilds beginnt dann das eigentliche Skript. Dort wird das Quellverzeichnis definiert. Dies geschieht normalerweise durch `S=${WORKDIR}/{P}`, wobei sich `{P}` aus dem Paketnamen und der Version zusammensetzt. Da diese Informationen aus dem Dateinamen abgeleitet werden können, kann auf die Definition von `{P}` verzichtet werden. Anschließend folgen die Beschreibung des Pakets in »`DESCRIPTION`«, ein Verweis auf die herunterzuladenden Dateien in »`SRC_URI`« und »`HOMEPAGE`« gibt die URL für das entsprechende Paket an. Zusätzlich werden noch der Lizenzinhaber des Skriptes »`LICENSE`«, eine Versionskontrolle »`SLOT`«

um mehrere Versionen eines Programms parallel installieren zu können, die unterstützte Rechnerarchitektur »KEYWORDS« und eventuell notwendige USE-Variablen »IUSE« angegeben.

Der dritte Abschnitt listet die Runtime Dependencies auf. Das sind die Pakete, die installiert sein müssen, um das Programm ausführen zu können. „Den Paketnamen können Operatoren vorangestellt werden, um die spezielle Version eines Paketes zu definieren. Möglich sind »=«, um nur die angegebene Version zu verwenden, »>=« um alle Versionen ab der angegebenen zuzulassen sowie analog dazu »<=«, »>«, »<« und »~« für die neueste verfügbare Version“ [Raschba.02]. Zusätzlich kann dieser Abschnitt Use-Abfragen enthalten, die überprüfen, ob das Programm mit speziellen Optionen übersetzt werden soll.

Im vierten Abschnitt sind die Build Dependencies aufgelistet. „Diese Abhängigkeiten sind nur zum Übersetzen, nicht aber zum Betrieb des Programms von Nöten“ [Raschba.02].

Den Abschluss bilden die Ebuild-Funktionen. „In eigenen Ebuild-Skripten sind die folgenden Funktionen erlaubt“ [Raschba.02]:

- `src_compile`
- `src_install`
- `src_unpack`
- `pkg_setup`
- `pkg_preinst`
- `pkg_postinst`
- `pkg_prerm`
- `pkg_postrm`
- `pkg_config`

5.3 Umsetzung für das Projekt

Anhand dieser eben beschriebenen Bestimmungen wurde auch das für EduTeCH geschriebene Ebuild erstellt. Dabei wurden jedoch auch einige Bereiche durch zusätzliche Angaben erweitert. Im Folgenden soll nun das von Oliver Schneider erstellte Ebuild-Skript erläutert werden.

Am Anfang von jedem Skript steht der Ebuild-Header (Zeile 1-3).

```
01 # Copyright 1999-2004 Gentoo Technologies, Inc.
02 # Distributed under the terms of the GNU General
   Public License v2
03 # $Header: $
```

Im Anschluss an die Header-Daten folgt das eigentliche Skript. Hier werden neben der Beschreibung des Projektes (Zeile 5) auch die Homepage (Zeile 6) und das von dort herunterzuladende Verzeichnis (Zeile 7) angegeben. Die Lizenz des Skriptes liegt beim ZGDV (Zeile 9). Da nicht mehrere Versionen des Programms installiert werden sollen, wird die entsprechende Variable auf Null gesetzt. Da das vorliegende Programm nur für den Einsatz auf PC-Systemen entwickelt wurde, werden lediglich x86-Systeme unterstützt (Zeile 11). Da keine speziellen USE-Variablen für die Installation notwendig sind, bleibt das entsprechende Feld leer (Zeile 12).

```
05 DESCRIPTION="Edutainment Technologies for Cul-
   tural Heritage in Asia"
06 HOMEPAGE="http://www.zgdv.de/"
07 SRC_URI="${P}.tar.bz2"
08
09 LICENSE="ZGDV"
10 SLOT="0"
11 KEYWORDS="x86"
12 IUSE=""
```

Die Runtime Dependencies geben an, welche Programme benötigt werden, um die EduTeCH-Anwendungen ausführen zu können. Dazu zählen

unter anderem »virtual/x11« für die graphische Oberfläche, »kde-base /kdemultimedia« für die Soundausgabe, »dev-lang/python-2.2« für die Story-Engine, »app-misc/itrax« für den Head-Tracker, sowie diverse andere für das Programm Blender.

```
14 RDEPEND="virtual/x11
15      media-libs/libsdl
16      media-libs/jpeg
17      media-libs/libpng
18      >=media-libs/freetype-2.0
19      >=media-libs/openal-20020127
20      >=media-libs/libsdl-1.2
21      >=media-libs/libvorbis-1.0
22      >=dev-libs/openssl-0.9.6
23      >=dev-lang/python-2.2
24      =dev-lang/python-2.0.1
25      kde-base/kdemultimedia
26      app-misc/itrax"
```

Der Abschnitt der Build Dependencies entspricht im vorliegenden Fall den Runtime Dependencies, weshalb lediglich der Verweis auf diese angegeben wird.

```
28 DEPEND=${RDEPEND}
```

Den Abschluss bildet die Ebuild-Funktion `src_install`. Sie legt die erforderlichen Verzeichnisse an (Zeilen 31 – 33 und 35) und legt die Zugriffsrechte fest (Zeile 34). Anschließend können die notwendigen Daten in die Verzeichnisse kopiert werden (Zeile 36 und 37). In Zeile 38 wird das Verzeichnis definiert, in das das für den Tracker notwendige Modul installiert wird (Zeile 39). Zum Abschluss wird jedem Archiv des Paketes das Verzeichnis mit den Ports, die für die Kommunikation über Sockets notwendig sind, mitgeteilt (Zeile 41).

```
30 src_install() {
31      dodir /usr/lib/python2.3
32      dodir /etc/env.d
33      dodir /opt/EduTeCH/
34      diropts -m0777
35      dodir /opt/EduTeCH/ports
36      cp -a ${S}/EduTeCH ${D}/opt/
37      cp -a ${S}/StoryEngine/*
        ${D}/usr/lib/python2.3/
38      exeinto /usr/lib/python2.0
39      doexe ${S}/tracker.so
40      echo PATH=/opt/EduTeCH >
        ${D}/etc/env.d/60edutech
41      touch ${D}/opt/EduTeCH/ports/.keep
42 }
```

6. Fazit

Das Ziel der Diplomarbeit war die Entwicklung einer interaktiven Storytelling AR-/VR-Spiel- und Lernumgebung. Zum Abschluss der Arbeit sollen die verschiedenen Bereiche zusammengefasst und ein Überblick über mögliche Erweiterungen gegeben werden.

Zunächst wurden die für das Verständnis der Arbeit notwendigen Grundlagen erläutert. Dabei wurde auf die unterschiedlichen Hardware-Anforderungen von AR- und VR-Anwendung eingegangen und die dafür benötigten Geräte vorgestellt. Außerdem wurden die verschiedenen Hardware-Lösungen miteinander verglichen und begründet, warum das jeweilige Gerät für das vorliegende Projekt verwendet wurde. Neben der Hardware wurde auch die verwendete Software sowie die verwendete Programmiersprache vorgestellt.

Im Anschluss daran wurde die Entwicklung der Virtual Reality-Anwendung beschrieben. Zielsetzung dieser Anwendung ist es, dem Benutzer einen virtuellen Rundgang durch die koreanische Tempelanlage zu ermöglichen. Um dies zu erreichen, waren verschiedene Arbeiten notwendig. Es wurden 3D-Modelle des Palastgeländes und des Charakters erstellt und diese anschließend texturiert. Nachdem diese Grundlagen geschaffen waren, musste die Logik des späteren Spiels programmiert werden. Der Steuerung des Spiels kommt dabei eine große Bedeutung zu, da der Benutzer sich so real wie möglich durch die Tempelanlage bewegen soll. Neben der Steuerung des Spielers wird auch der Ablauf des Spiels durch die Game-Logic gesteuert. Unterstützt wird sie dabei durch die verwendete Story-Engine, die für die Steuerung der Geschichte verantwortlich ist. Auch die Soundausgabe des Spiels erfolgt über die Game-Logic.

Nachdem die VR-Anwendung fertig gestellt war, musste sie an die neuen Anforderungen, die durch den Augmented Reality-Einsatz entstanden, angepasst werden. In diesem zweiten großen Problemfeld der Diplomarbeit wurden zunächst die notwendigen Veränderungen herausgearbeitet, die sich

durch den geänderten Anwendungsbereich ergeben haben. Dabei kann die Game-Logic übernommen werden und muss lediglich in einigen Bereichen erweitert werden, wodurch die Anwendung insgesamt sehr flexibel bleibt. Die durchzuführenden Änderungen waren notwendige Folge der geänderten Hardware. Das Bild wird hierbei auf AR-Glasses übertragen und überlagert dadurch das Bild der realen Welt. Die Blickrichtung des Spielers wird durch einen auf den AR-Glasses montierten Tracker gesteuert. Die Position des Spielers in der Palastanlage sollte mittels eines GPS-Gerätes ermittelt werden, was jedoch auf Grund der in Kapitel 4.3.4 geschilderten politischen Entscheidungen und Sicherheitsmaßnahmen nicht möglich war. Deshalb wurde lediglich die prinzipielle Vorgehensweise bei der Integration eines solchen Gerätes beschrieben und auf die Implementierung verzichtet.

Zum Abschluss der Arbeit wurde ein Installationsskript für das verwendete Linux-Betriebssystem erstellt, das die einfache Installation des Spiels ermöglicht.

Auch nach Abschluss dieser Diplomarbeit sind noch Verbesserungen bzw. Erweiterungen in verschiedenen Bereichen der Anwendung denkbar. So könnte beispielsweise die Optik des Spiels noch durch bessere Texturen deutlich aufgewertet werden. Die in dieser Arbeit verwendeten Texturen wurden von Mitarbeitern in Seoul gemacht, die keine Erfahrung im Fotografieren von Texturen hatten. Dadurch waren die gelieferten Bilder nur begrenzt für die Texturierung geeignet. Erweiterungen des Spiels wären im Bezug auf die Animationen des Charakters, die Anzahl der verwendeten Gebäude oder den Umfang der Geschichte denkbar. Dabei muss jedoch immer die Leistungsfähigkeit der Game-Engine berücksichtigt werden und notfalls die Spielfläche in mehrere kleine Gebiete unterteilt werden. Da die Game-Engine ständig verbessert wird, wäre auch eine Portierung des Spiels auf die aktuelle Version denkbar, um so von neuen Features profitieren zu können und den Leistungszuwachs auszunutzen. Um das Spiel auch richtig als AR-Anwendung betreiben zu können, wäre die Integration eines GPS-Empfängers notwendig. Zusätzlich könnte noch eine Live-CD erstellt werden, die ein komplettes Betriebssystem inklusive dem erstellten Spiel ent-

hält. Diese CD würde es dem Nutzer ermöglichen, die Anwendung unabhängig vom verwendeten Betriebssystem auf nahezu jedem PC-System zu verwenden.

Durch den sehr variablen Aufbau des Spiels sind diverse Einsatzgebiete für die entwickelte Anwendung möglich. Eine Anpassung an andere Örtlichkeiten ist durch den Austausch der 3D-Modelle problemlos möglich. Gleiches gilt auch für die verwendete Geschichte, die an neue Inhalte angepasst werden kann. Die Game-Logic kann in andere Projekte übernommen und bei Bedarf auch verändert oder erweitert werden. Durch diese Eigenschaften stellt die entwickelte Anwendung ein sehr variables System dar, das für unterschiedlichste Projekte eingesetzt werden könnte.

7. Literaturverzeichnis

adLexicon *Dieser Artikel basiert auf dem Artikel Avatar aus der freien Enzyklopädie Wikipedia und steht unter der GNU Lizenz für freie Dokumentation.*
http://avatar.adlexikon.de/Avatar.shtml
letzter Zugriff: Januar 2005

Anthes02 C. Anthes: *Virtual Reality im CAVE WS 2002/03 – Der CAVE und andere VR Geräte*
http://www.gup.uni-linz.ac.at/vrcave
letzter Zugriff: November 2004, erstellt 05.11.2002

Azuma01 R. T. Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, Blair MacIntyre: *Recent Advances in Augmented Reality*, IEEE Computer Graphics and Applications, 0272-1716/01,
http://www.cs.unc.edu/~azuma/cga2001.pdf,
letzter Zugriff: Dezember 2002, erstellt 2001

Azuma97 R. T. Azuma: *A survey of Augmented Reality*, Presence: Teleoperators and Virtual Environments 6, 4(August 1997), Seiten 355-385,
http://www.cs.unc.edu/~azuma/ARpresence.pdf,
letzter Zugriff: November 2004, erstellt 1997

Azuma99 R. T. Azuma: *The Challenge of Making Augmented Reality Work Outdoors,* In *Mixed Reality: Merging Real and Virtual Worlds,* 1999. Kapitel 21 Seiten 379-390,
http://www.cs.unc.edu/~azuma/ismr99.pdf
letzter Zugriff: November 2004, erstellt 1999

Bild01 *Beispielbild für eine AR-Anwendung*
http://www.ordnancesurvey.co.uk/oswebsite/images/userImages/misc/research/augmented_small.jpg
letzter Zugriff: März 2005

Bild02 *Beispielbild für eine AR-Anwendung*
http://www.zgdv.de/zgdv/departments/z2/Z2Abt/AR/AR.jpg
letzter Zugriff: März 2005

Braun03 N. Braun: *Nonlinear Storytelling: Programmierter, interaktiver Narrationsansatz für kontinuierliche Medien*
http://elib.tu-darmstadt.de/diss/000497/Dis_Norbert.pdf
letzter Zugriff: November 2004, erstellt 2003

Brown C. M. Brown: *Beispielbild für eine AR-Anwendung*
http://www.cs.rochester.edu/u/brown/Images/brain_ar.gif
letzter Zugriff: November 2004

Davies D. Davies, D. Robbins, P. Gavin, K. T. Kalleberg, J. P. Davis, M. Frysinger: *Gentoo Ebuild/Entwickler HOWTO*
http://www.gentoo.de/doc/de/gentoo-howto.xml
letzter Zugriff: November 2004

Evers01 J.-F. Evers: *Entwicklung einer orientierungssensitiven Kamera zur Erstellung von Panoramabildern*
http://www.mip.informatik.uni-kiel.de/~evers/Dipl-Arbeit_Evers.pdf
letzter Zugriff: Januar 2005, erstellt 25.06.2001

Fischbach99 R. Fischbach: *Wie Python arbeitet*
in: iX Magazin für professionelle Informationstechnik 11/1999, S. 184
http://www.heise.de/ix/artikel/1999/11/184/
letzter Zugriff: November 2004, erstellt 1999

Geiger01 C. Geiger, B. Kleinjohann, C. Reimann, D. Stickling, W. Rosenbach: *Interaktive VR/AR-Inhalte auf mobilen Endgeräten*, C-LAB, Paderborn, http://webster.fhs-hagenberg.ac.at/staff /haller/giocg/12_geiger.pdf, letzter Zugriff: November 2004, erstellt 2001

Geiger01a C. Geiger, V. Paelke, C. Reimann, W. Rosenbach: *Structured Design of Interactive Virtual and Augmented Reality Content*, C-Lab, Paderborn, http://www.c-lab.de/web3d/VEWorkshop/ Christian_Geiger.pdf, letzter Zugriff: November 2004, erstellt 2001

Geiger02 C. Geiger: *Design Issues of Virtual and Augmented Reality Authoring*, Workshop Proceedings Production Process of 3D Computer Graphics Applications-Structures, Roles and Tools:ACM SIGGRAPH and Eurographics Campfire, Shaker Verlag, ISBN 3-8322-0241-2, S. 51-58, Aachen 2002

Geiger02a C. Geiger: *Tutorial: Augmented Reality auf der Simulation & Visualisierung 2002 in Magdeburg,* http://www.fhs-hagenberg.ac.at/staff/haller/mmp6_2002/ 07vrApplications_1.pdf, letzter Zugriff: November 2004, erstellt 01.10.2002

GenLia *Diese Lexikon-Artikel stehen alle unter der GNU FDL (GNU Freie Dokumentationslizenz). Wikipedia ist eine mehrsprachige Enzyklopädie, deren Inhalte frei nutzbar sind.* http://www.computerbase.de/lexikon/Gentoo-Linux letzter Zugriff: November 2004

GenLib *Diese Lexikon-Artikel stehen alle unter der GNU FDL (GNU Freie Dokumentationslizenz). Wikipedia ist eine mehrsprachige Enzyklopädie, deren Inhalte frei nutzbar sind.*
http://www.computerbase.de/lexikon/Ebuild
letzter Zugriff: November 2004

Goebel03 S. Göbel, A. Hoffmann, O. Schneider: "*EduTeCH*",
http://www.zgdv.de/zgdv/departments/z5/Z5Projects/EduTeCH
letzter Zugriff: November 2004, erstellt 2003

Haller02 M. Haller: *Multimediaprogrammierung und Sensorik 6, Virtual Reality, Thema: Virtual Reality (Einführung),*
http://www.fhs-hagenberg.ac.at/staff/haller/mmp6_2002/04vrIntroduction_1.pdf,
letzter Zugriff November 2004, erstellt 01.10.2002

Haller02a M. Haller: *Multimediaprogrammierung und Sensorik 6, Virtual Reality, Thema: Virtual Reality (Hardware)*,
http://www.fhs-hagenberg.ac.at/staff/haller/mmp6_2002/05vrHardware_1.pdf,
letzter Zugriff: November 2004, erstellt 01.10.2002

Heilbrun A. Heilbrun, B. Stracks: *Was heißt „virtuelle Realität"? Ein Interview mit Jaron Lanier. In: M. Waffender (Hrsg.): Cyberspace: Ausflüge in virtuelle Wirklichkeiten.*
Rowohlt: Reinbeck bei Hamburg, 1991, S. 67-87

Herzig03 C. Herzig, J. Shin: *EduTeCH – Edutainment Technologies for Cultural Heritage in Asia*
http://www.inigraphics.net/press/topics/2003/issue5/5_03a10.pdf
letzter Zugriff: November 2004, erstellt Mai 2003

Himstedt96 T. Himstedt: *Python: objektorientierte Scriptsprache fürs World Wide Web*, in : iX Magazin für professionelle Informationstechnik 3/1996, S. 144
http://www.heise.de/ix/artikel/1996/03/144/
letzter Zugriff: November 2004, erstellt 1996

Ihringer J. Ihringer: *Skripten zur Vorlesung Experimentalphysik I Das Auge*
Uni Tübingen, Institut für Angewandte Physik
http://www.uni-tuebingen.de/uni/pki/skripten/V8_2_4Auge.DOC
letzter Zugriff: November 2004

Krenzel04 R. Krenzel: *Repräsentation von realen Objekten in Outdoor Augmented Reality Systemen*, Diplomarbeit an der FH Gießen-Friedberg, 2004
http://info.bib.fh-giessen.de/opus/volltexte/2004/3160/
letzter Zugriff: November 2004, erstellt 2004

Krömk02 Krömker: Skript: *Virtual Reality & Augmented Reality*, Fachbereich: Informatik, Johann Wolfgang Goethe-Universität Frankfurt Sommersemester 2002,
http://www.gdv.informatik.uni-frankfurt.de/
index.php?m=2&sm=3,
letzter Zugriff: November 2004

Leberl01 C. Leberl: *Stereopsis 1*
http://fcggpc41.icg.tu-graz.ac.at/grabner/BACG_Online/
Kapitel-13-Stereopsis/sld022.htm
letzter Zugriff: November 2004, erstellt 29.01.2001

Loewis01 M. von Löwis, N. Fischbeck: *Python 2 – Einführung und Referenz der objektorientierten Skriptsprache*, 2. Auflage
Addison-Wesley, ISBN: 3-8273-1691-X, 2001

Pammer00 L. Pammer: *Python-Tutorial*
http://www.ssw.uni-linz.ac.at/Teaching/Lectures/Sem/2000/Pammer/index.htm
letzter Zugriff: November 2004, erstellt 2000

Raschba.02 T. Raschbacher: *Die Technik von Gentoo Linux – Formwandler* im Linux-Magazin Ausgabe 09/2002
http://www.linux-magazin.de/Artikel/ausgabe/2002/09/gentoo/gentoo.html
letzter Zugriff: Januar 2005, erschienen September 2002

Roosendaal03 T. Roosendaal, C. Wartmann: *The Official Blender Gamekit, Interactive 3-D for Artists*
No Starch Press, 2003, ISBN: 1-59327-004-6

Roosendaal04 T. Roosendaal, S. Selleri: *Blender 2.3 Guide, free 3D creation suite for modelling, animation and rendering*
No Starch Press, 2004, ISBN: 1-59327-041-0

Ruegge99 I. Rügge: Alternativen zum Desktop: *Virtual, Augmented und Real Reality, Frauenarbeit und Informatik* 19/1999
Seite 17 - 22
http://www.tzi.de/~ruegge/PDF-Dateien/GI-R19-99-RR.pdf
letzter Zugriff: November 2004, erstellt 1999

Schaar03 F. Schaar: Programmieren einer Story-Engine in Python für interaktive Geschichten in Augmented Reality Projekten Praktikumsbericht, erstellt 2003

Schum96 T. Schumann : *Das Grundprinzip des GPS*, TU-Ilmenau,
http://ikmcip1.e-technik.tu-ilmenau.de/~traut/gps_www/gps_prz.htm,
letzter Zugriff: November 2004, erstellt 1996

Sorantin04 E. Sorantin: *Virtuelle Leberchirurgie - Neue Wege zu alten Problemen*
http://liverplanner.icg.tu-graz.ac.at/modules/LSPSI/images/arinteraction2.png
letzter Zugriff: November 2004, erstellt 27.04.2004

Verbre98 E. Verbree, L. Verzijl , M.-J. Kraak: *Integrated 3D-GIS and VR: Use of Virtual Reality and 3DGIS within the Planning Process Concerning the Infrastructure,* 10th Colloquium of the Spatial Information Research Centre
http://divcom.otago.ac.nz/sirc/webpages/Conferences/SIRC98/98Abstracts/98Verbee/Verbree.pdf,
letzter Zugriff: November 2004, erstellt 1998

Walker99 J. Walker: *Piecing together and tearing apart: finding the story in afternoon.* In Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots, pages 111 – 117. ACM, 1999

Wartmann00 C. Wartmann: *Das Blender Buch*, 2. Auflage
dpunkt.verlag, 2000 , ISBN: 3-89864-100-7

Wohlfahrt04 R. Wohlfahrt: *Sieht ja alles aus wie früher!*
in: Frankfurter Allgemeine Sonntagszeitung
erschienen: 09.05.04

8. Anhänge

8.1 Anhang A: Skripte der Diplomarbeit

9.2 Anhang B: Übersichten zur Game-Logic und zum Spielaufbau

9.3 Anhang C: Inhalt der CD-Rom zur Diplomarbeit

Die CD-Rom, welche dieser Diplomarbeit beiliegt, enthält die in dieser Arbeit erstellte Anwendung in zwei Ausführungen. Eine Version ist für die Installation unter Linux, die andere für die Installation unter Windows gedacht. Jeder Version liegen alle notwendigen Programme und Dateien sowie eine detaillierte Installationsanweisung bei. Zusätzlich zum Spiel enthält diese CD auch eine digitale Version dieser Diplomarbeit.

Verzeichnisstruktur der CD-Rom:

- **Ordner Linux**
  Installationsanleitung und alle notwendigen Dateien

- **Ordner Windows**
  Installationsanleitung und alle notwendigen Dateien

- **Ordner Diplomarbeit**
  Digitale Version dieser Diplomarbeit