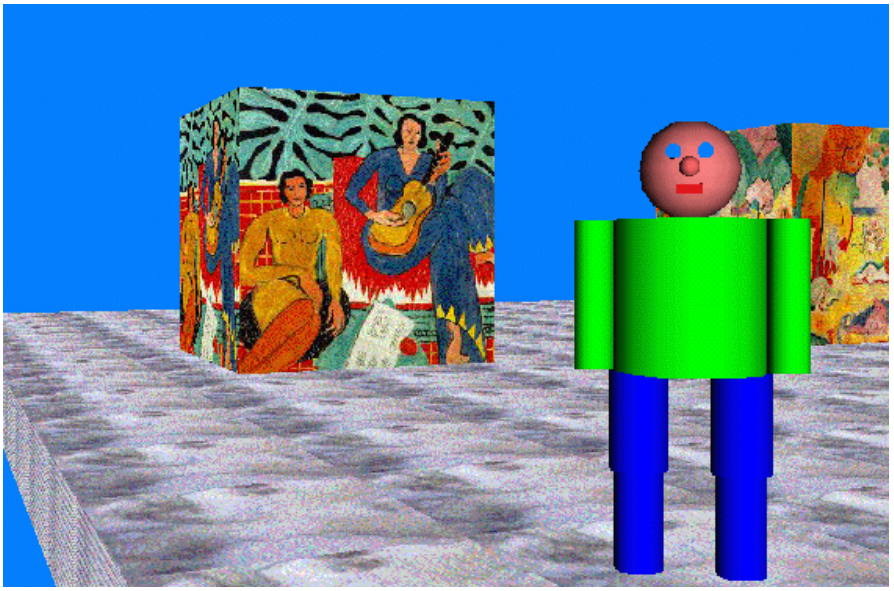
# AUTOMATISIERTE CHARAKTERE IN VIRTUELLEN WELTEN



von

Hauke Ernst

Informatik
Universität Bremen

zur Erlangung des akademischen Grades

## Diplom Informatiker

vorgelegte Diplomarbeit

Gutachter:

1. Professor Dr. rer. nat. Wolfgang Coy
2. Professor Dr.-Ing. Friedrich-Wilhelm Bruns

Bremen, den 24. Juli 1997

*Ich möchte allen danken,*

*die mich beim Entstehen dieser Arbeit*

*unterstützt haben.*

„Das Motiv" sagte die Konstruktion. „Das Motiv ist echt 'n Problem bei 'ner KI. Nicht menschlich, verstehste?"
„Tja ... na klar."
„Nix ist klar. Ich meine, 'ne KI ist kein Mensch. Aus der wirst du nie schlau. Ich bin auch kein Mensch, aber ich *reagiere* zumindest wie einer."

William Gibson (s. [GIB96], S. 163)

## Kurzfassung

Diese Arbeit thematisiert die Motivation, die Realisierung und die Anwendung von automatisierten Charakteren in virtuellen Welten. Jüngste Entwicklungen im Bereich der Virtuellen Realität machen es möglich, vernetzte virtuelle Welten aufzubauen, die gleichzeitig von mehreren Teilnehmern besucht werden können. Dabei wählt jeder Teilnehmer jeweils eine dreidimensionale Figur aus, die ihn den anderen Teilnehmern gegenüber repräsentiert. So kann jeder Benutzer die Aktionen der anderen verfolgen, aber auch mit ihnen in Dialog treten oder gemeinsame Aktionen durchführen.

In dieser Arbeit wird ein Konzept erarbeitet, wie das Verhalten von Charakteren in vernetzten virtuellen Welten, also Bewegungsabläufe und Textdialoge mit realen Teilnehmern, automatisch gesteuert werden kann. Die Realisierung der Ergebnisse erfolgt in einer Kopplung von VRML2 und Java, wobei die 3D-Darstellungen in VRML2 und die Mechanismen der Verhaltenssteuerung in Java definiert werden. Das Verhalten selber kann mit Hilfe einer Skriptsprache gesteuert werden, deren Interpretation Javaklassen übernehmen, die zusätzlich zur Skriptsprache in dieser Arbeit entwickelt werden. In besagter Skriptsprache lassen sich Verhaltensweisen in Prozeduren definieren und hierarchisch strukturieren, was durch ein Beispiel gezeigt wird. Weitergehende Ansätze können auf die zur Verfügung gestellte Basisfunktionalität unter Anwendung der Skriptsprache, aber auch durch eine objektorientierte Programmierschnittstelle, aufbauen. Die Programmierschnittstelle wird durch eine einfache Implementation des Schema-Konzeptes demonstriert.



## Abstract

This work deals with the motivation, the realisation and the application of automated characters in virtual worlds. Latest developments in the area of virtual reality make it possible, to set up networked virtual worlds, which can be visited simultaneously by several participants. Thereby each participant selects respectively a three-dimensional figure, which represents him to other participants. So each user can follow the actions of the other, but, in addition, he can carry on dialogs with other participants and perform common actions.

In this work a concept is worked out about the behaviour of characters in networked virtual worlds, that is movements and text dialogs with real participants, which can be automatically controlled. The realisation of the concept results in a tie-in of VRML2 and Java, whereby the 3D-presentations are defined in VRML2 and the mechanisms of behaviour are controlled in Java. The behaviour itself can be described with the help of a script language, which is interpreted by Java-classes, both subjects of development in this work. In this script language behaviour can be defined in procedures. Moreover behaviour can be hierarchically structured. The features of the script language are shown by an example. More extended approaches can set up on this platform by means of the script language, but also through an object-oriented program interface. The applicability of the program interface is demonstrated through a simple implementation of the scheme-concept.

INHALTSVERZEICHNIS

# 1 Einleitung

Als *Automaten* (griechisch αυτοματος → „Sich selbst bewegend“) bezeichnete man ursprünglich mechanische Gebilde, die als Reaktion auf ein Ereignis Tätigkeiten von Menschen oder Tieren nachahmten, indem sie einen festgelegten Plan schrittweise ausführten. Bereits im Altertum wurden Mechanismen gebaut, die die Bewegungen von Lebewesen nachbildeten. So schrieb Heron von Alexandria im ersten Jahrhundert n. Chr. über den Bau von Automatentheatern. Später - einhergehend mit der Entwicklung von Uhrwerken - entstanden feinmechanische Kunstwerke wie beispielsweise die künstliche Ente von Vaucanson (1738) oder der Schreiber von Droz (1760), aber auch verschiedene Uhren mit Figurenwerk. (vgl. dtv Brockhaus Lexikon 1984)

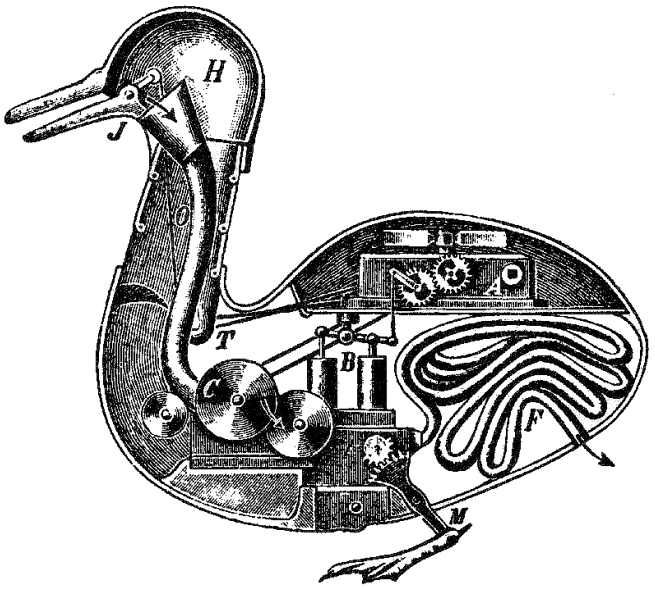


Abbildung 1: Künstliche Ente von J. de Vaucanson

Der Begriff *Charakter* (griechisch χαρακτηρας → „Das Eingeprägte“) bedeutet im weiteren Sinne die typischen Wesensarten und Merkmale einer Person oder Sache, wird aber auch für die Persönlichkeit, die inneren Werte, eine Rolle oder die äußere Erscheinung eines Menschen angewendet. (vgl. dtv Brockhaus Lexikon 1984)

Die Kombination der Worte *Automat* und *Charakter* im Titel meiner Arbeit „Automatisierte Charaktere in virtuellen Welten“ läßt sich dementsprechend auf verschiedene Weisen deuten:

1. eine Figur wird durch einen vorher festgelegten Plan gesteuert;
2. typische Merkmale und Eigenarten eines Menschen oder einer Rolle werden mit Hilfe eines Programmes nachgeahmt;
3. menschliche Tugenden wie ein Bewußtsein für Verantwortung oder soziales Handeln werden auf ein Programm übertragen.

Um es vorwegzunehmen: ich werde mich im Rahmen dieser Arbeit nur mit den Punkten 1 und 2 - also der Modellierung von Bewegungen und Rollen durch Handlungspläne - beschäftigen, da Punkt 3 (die Übertragung eines Bewußtseins für innere Werte auf ein Programm) meiner Ansicht nach weder erstrebenswert noch vorstellbar ist.

Abbildung 2: Ansicht einer virtuellen Realität
aus dem VRML2-Browser von Sony
„Community Place“

Durch die rasante Entwicklung der Computer bis heute sind mittlerweile auch aufwendige Berechnungen in Echtzeit möglich geworden. Eine solche Anwendung, die enorme Ansprüche an die Rechenleistung stellt, ist die dreidimensionale Darstellung von Szenen, die im Computer als Modelle repräsentiert sind. Für jeden Bildpunkt muß das Zusammenspiel von Perspektive, Licht und Schatten, Spiegelungen, Transparenz et cetera simuliert werden, damit Bilder entstehen können, die aus unserer Sicht realistisch und glaubhaft wirken. Systeme, die ein Operieren und Navigieren in derartigen von Computern simulierten, dreidimensionalen Wirklichkeiten oder künstlichen Welten erlauben, werden *Virtuelle Realitäten* (VR) genannt. Dabei wird dem Benutzer eine ständig aktualisierte Ansicht der Welt dargeboten, deren Perspektive er durch Steuerinstrumente manipulieren kann.

Wichtig ist, daß dem Betrachter der Bilder das Gefühl vermittelt wird, ein Teil der künstlichen Welt zu sein, und er so vom Betrachter zum Teilnehmer wird. Diese Illusion kann durch das Vorgaukeln von Sinnesreizen, die aufgrund von Erfahrungen in der realen Welt glaubhaft erscheinen, erzeugt werden.

Die Einbindung von Personen in virtuelle Realitäten kann durch spezielle Ein- und Ausgabegeräte, bei denen ein breiteres Sinnesspektrum als bei herkömmlicher Hardware einbezogen wird, gefördert werden. Zu nennen sind in diesem Zusammenhang insbesondere Datenhelme oder Brillen zur Präsentation von stereoskopischen Bildern mit einer Verstärkung des 3D-Eindruckes und Datenhandschuhe oder -anzüge zum gesten- beziehungsweise tastorientierten Navigieren und Operieren.

Abbildung 3: Benutzerin mit Datenhelm und -handschuhen

(von McGreevy, Michael / Fisher, Scott, NASA Ames Research Center, Moffet Field, CA., aus [FOL93])

Das „Eintauchen“ in virtuelle Welten mit Hilfe spezieller Geräte wird als *Immersion* bezeichnet. Solche Geräte sind jedoch nur ein Aspekt von virtueller Realität. Nicht ebensogut, aber dennoch möglich ist auch der Einsatz von normalen Tastaturen, Mäusen und Monitoren, bei dem ein Teil der *Immersion* durch *Imagination*, also Vorstellungskraft, ausgetauscht werden muß.

Das Wesentliche an einem VR-System ist jedoch eher die Software, die vorgibt, welche virtuellen Welten modellierbar, welche Interaktionsformen realisiert sind und wie die Umsetzung der Szenen in Bilder erfolgt. Die Software bestimmt ebenfalls, inwiefern mehrere Teilnehmer gleichzeitig in virtuelle Realitäten eintreten dürfen, ob sie sich gegenseitig bei ihrem Handeln beobachten und auf welche Weise sie in Kontakt treten oder sogar gemeinsame Aktionen durchführen können. Diese Faktoren sind es, die über den praktischen Nutzen der VR als universelle Benutzungsschnittstelle für das Navigieren und Operieren in Daten, Simulationen oder Rechnernetzen und für kooperatives Arbeiten entscheiden.

Virtuelle Welten ohne andere Teilnehmer erzeugen mit der Zeit ein Gefühl von Einsamkeit; unbelebte virtuelle Städte sind wie Geisterstädte. Üblicherweise wird man daher in Systemen, die mehrere Benutzer zulassen, für die jeweils anderen als dreidimensionale Gestalt mit bestimmtem Bewegungsverhalten dargestellt, die sozusagen als Stellvertreter seiner selbst für die anderen sichtbar wird. Durch das Aussehen seiner sich vertretenden Figur, die *Avatar* genannt wird, und sein Verhalten in der virtuellen Welt prägt man bewußt oder unbewußt einen individuellen Charakter für seinen Stellvertreter, in dessen Rolle man beim Betreten der virtuellen Welt schlüpft.

Wie im wirklichen Leben gibt es aber auch in virtuellen Welten nicht nur die Rolle von Zuschauern und Konsumenten, vielmehr werden bestimmte Orte die Anwesenheit von Charakteren regelrecht benötigen, um funktionieren oder zur Geltung kommen zu können. Dabei denke ich an virtuelle Gegenstücke zu Verkäufern in Geschäften, zu Passanten und Musikern auf der Straße, aber auch an komplexe Datenverarbeitungssysteme (DV-Systeme) wie personifizierte Hilfesysteme in Gestalt von Museums- oder Stadtführern und Bibliothekaren, also Teile der Kulisse oder Funktionsträger, die basierend auf der Metapher Mensch realisierbar sind. Aus diesem Kontext heraus entstand für mich die Motivation, mich mit automatisierten Charakteren in virtuellen Welten zu beschäftigen. Mit Hilfe von programmgesteuerten Figuren möchte ich das Verhalten von Charakteren nachahmen und so Rollen besetzen, die ansonsten fehlen würden. Dies soll zu einer größeren Realitätsnähe virtueller Welten führen.

Diese Arbeit wird keine ultimativen Lösungen vorstellen, sondern stattdessen eine ausbaufähige Plattform zur Entwicklung und Erprobung von automatisierten Charakteren anbieten. Der Schwerpunkt soll sein, viele Möglichkeiten der Weiterentwicklung und Integration spezialisierter Module offenzuhalten. In diesem Zusammenhang habe ich eine Skriptsprache zur Verhaltenssteuerung entwickelt, die die Definition von reaktivem Verhalten vereinfachen soll. Darüberhinaus habe ich eine Programmierschnittstelle für Erweiterungen in Java implementiert und diese auch beispielhaft angewendet, indem ich auf ihrer Grundlage ein weitergehendes Konzept - auf der Basis von Schemata - eingebracht habe.

Zunächst wird diese Arbeit aber von den Grundlagen handeln, angefangen bei den Prinzipien virtueller Realität. Danach werde ich auf ausgewählte psychologische Modelle der menschlichen Verhaltenssteuerung eingehen, um Ideen für eigene Ansätze zu entwickeln. Es folgt eine Zusammenstellung verschiedener Methoden zur Animation komplexer Objekte, die mir Unterstützung bei der Defintion von Bewegungen liefern soll. Das dann anschließende Kapitel wird sich mit Agenten beschäftigen und gibt einen Überblick über den Stand der Forschung zu diesem Thema, das die Entwicklung von reaktiver Software mit verteilter Entscheidungskompetenz zum Schwerpunkt hat. Nach einer kurzen Vorstellung von Standardisierungsbemühungen im Zusammenhang mit virtuellen Welten im Mehrbenutzer-Betrieb und Avataren komme ich schließlich zu der Beschreibung der von mir entwickelten Konzepte und Software zur Steuerung von automatisierten Charakteren in virtuellen Welten. Daß dabei im Rahmen dieser Arbeit nicht alle Fragen beantwortet und alles technisch Machbare realisiert werden konnte, ist selbstverständlich. Deswegen ist der abschließende Ausblick denjenigen Dingen gewidmet, die aus Gründen der begrenzten Zeit zur Programmierung, dem Fehlen ausreichend schneller Hardware oder den Zwängen von Standards nicht untersucht werden konnten. Auch Ideen für mögliche Weiterentwicklungen sollen an dieser Stelle vorgestellt werden.

*Der Autor bittet um Vergebung, daß aus ästhetischen Gründen darauf verzichtet worden ist, ständig die Existenz weiblicher Ausprägungen von Menschen explizit im Text kenntlich zu machen.*

*Ebenso weise ich darauf hin, daß diese Arbeit nach der alten (bisher gültigen) Rechtschreibung verfaßt ist.*

# 2 Virtual Reality Systeme

## 2.1 Virtuelle Realität

Medien sind Kommunikationsmittel („Vermittelndes“) zur Verbreitung von Wissen - etwa durch Zeichen, Bilder, Rede, Druck, Film oder Rundfunk - in gesellschaftlichen Gruppen oder an ein großes Publikum (vgl. dtv Brockhaus Lexikon 1984). Sie übertragen Informationen auf ihr Publikum, indem sie Modelle in der Vorstellung dieser Menschen erzeugen. Die Modelle können raumzeitlicher, aber auch abstrakter Natur sein - wie zum Beispiel in vielen Fachbüchern. Das Medium selbst bestimmt und begrenzt die Formen der Modellentwicklung. Die Grenzen von Medien auszureizen oder durch Tricks zu überschreiten, um beim Empfänger eine möglichst störungsfreie (vgl. [SHA76]), reale und umfassende Vorstellung der zu übertragenden Sachverhalte zu erzeugen, ist seit jeher das Anliegen der Sender von Nachrichten mit Hilfe von Kommunikationsmitteln: Kinoproduzenten versuchen, die Flachheit der Leinwand durch breite Leinwände, ausgefeilte Illusionstricks und Rundumbeschallung zu überwinden, Buchautoren konstruieren durch detaillierte Beschreibungen von Akteuren und Gegebenheiten imaginäre Szenen. (vgl. Zielinski In: [RÖT93])

Das Hereinholen des Lesers in eine vom Autor konstruierte Phantasiewelt wird zum Beispiel in Michael Endes „Die unendliche Geschichte“ thematisiert. In diesem bekannten Roman wird erzählt, wie der Leser eines Buches sich plötzlich in der darin beschriebenen Welt mit allerlei phantastischen Kreaturen, Orten und Phänomenen wiederfindet. Dort erlebt er die Handlung aus seiner eigenen Perspektive, greift in die Handlung ein und interagiert mit den Akteuren, die ihn als jemand wahrnehmen, der er in Wirklichkeit gar nicht ist. Er als Leser (Zuschauer) ist in die Rolle eines Akteurs geschlüpft.

> *„... nicht mehr schauen oder glotzen, sondern tun / konzipieren / machen - das wäre der Modus, in dem sich dann auch der filmische Prozeß realisierte.“*
>
> (s. Zielinski In: [RÖT93], S.55)

Genau diese Idee wird bei der sogenannten *Virtuellen Realität* verfolgt: Der Zuschauer wird zum Akteur und wirkt bei der Gestaltung der Handlung mit. Dabei nimmt er sein Umfeld aus seiner persönlichen Perspektive wahr, die er durch Steuerungsmechanismen verändern kann. Mit Hilfe dieser Mechanismen kann der Benutzer durch die *virtuelle Welt* navigieren und erhält dabei ständig aktuali-

sierte, für ihn persönlich aufbereitete Beschreibungen seiner Weltsicht. In frühen Systemen dieser Art waren und sind diese Beschreibungen textuell, heutzutage verbreiten sich dank schnellerer Rechner Systeme mit Echtzeit-Graphikausgabe, die aus den dreidimensionalen Szenenbeschreibungen realistische Projektionen auf Monitoren oder speziellen Ausgabegeräten mit verbesserter räumlicher Illusion erzeugen. Zum Navigieren stehen heute neben Tastaturen, Joysticks und Mäusen auch Datenhandschuhe oder -anzüge zur Verfügung.

In künstliche Welten einzutauchen und die Grenzen der alltäglichen Erfahrung zu verschieben, ist keine neue Idee, vielmehr läßt sich der Wunsch danach durch die gesamte Menschheitsgeschichte zurückverfolgen. Auf unterschiedliche Weisen - etwa durch Panoramadarstellungen, durch perfekten Realismus, durch Zuhilfenahme von technischen Hilfsmitteln oder Drogen - sind immer wieder Grenzüberschreitungen, häufig von Künstlern, in andere Realitäten versucht worden. So ist es nicht verwunderlich, daß gerade Künstler sich neuerdings mit VR-Installationen beschäftigen; bietet die neue Technik doch die Möglichkeit, nicht nur vielfältige und dynamische künstliche Welten unmittelbar zu erschaffen, sondern auch *mitten in ihnen* zu sein, sie zu durchwandern und Interaktionsformen mit in das Kunstwerk einzubeziehen.

> *„Revolutionär ist nicht die Konstruktion künstlicher Welten, sondern die Möglichkeit, in sie einzutreten, also die normale Situation des passiven oder auch manipulativ eingreifenden äußeren Beobachters insoweit zu verändern, als dieser nun die künstlichen Welten von innen und als Teil ihrer beobachtet.“*
>
> (s. Rötzer In: [RÖT93], S. 9)

Mit solcherart interaktiven Computerinstallationen beschäftigen sich zum Beispiel Christa Sommerer und Laurent Mignonneau. Ihr Hauptaugenmerk liegt dabei auf der *„Verbindung von Kunst und lebendigen oder lebensähnlichen Systemen, wie etwa echtes Leben, Künstliches Leben und Virtuelles Leben.“*(s. [SOM97b])

Exemplarisch sei hier eine Installation namens A-Volve dieser beiden Künstler skizziert: Künstliche, von einem schnellen Graphikrechner in Echtzeit erzeugte Kreaturen werden mit Hilfe einer Spiegelkonstruktion in ein Wasserbecken projiziert, in denen sie sich bewegen und miteinander interagieren können. Benutzer des Systems ha-

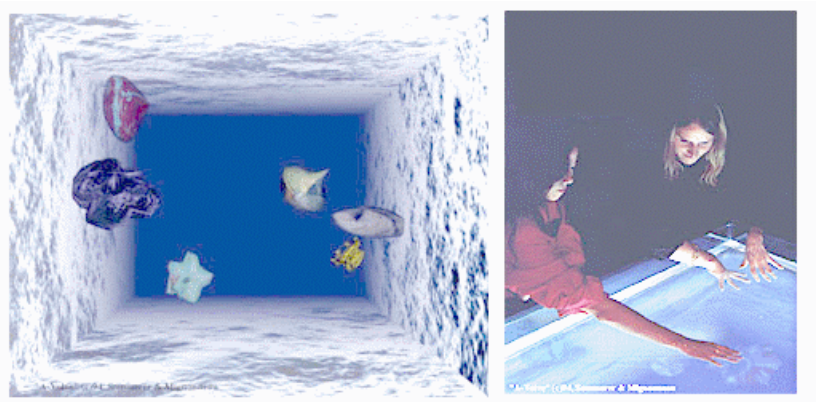


Abbildung 4: A-Volve, NTT-ICC 94. © Sommerer u. Mignonneau

ben die Möglichkeit, eigene Kreaturen an einem Touchscreen zu entwerfen und in das Wasserbekken einzuschleusen. In diesem „leben" sie dann für eine gewisse Zeit, wobei sie sich ihrer Form entsprechend fortbewegen. Die Form allein bestimmt, auf welche Weise und mit welcher Geschwindigkeit sich die Kreaturen fortbewegen. Die Geschwindigkeit hat für sie eine besondere Bedeutung, denn in dem Wasserbecken herrschen die harten Gesetze des Lebens: die Schnellen fressen die Langsamen, nehmen deren Energie auf und erhöhen damit ihre Chance, länger zu leben und sich zu vermehren. Der Fortpflanzungsmechanismus folgt den Prinzipien der Genetik, also der Neukombination von Eigenschaften nach den Regeln Mendels und der Herstellung neuer Eigenschaften durch die Mechanismen Mutation und Cross-Over. Die Bevorzugung einzelner Individuen - nämlich der schnelleren - realisiert eine einfache evolutionäre Entwicklung der Population. Die Betrachter beziehungsweise Benutzer des Systems können die Kreaturen beobachten, indem sie von oben in das Wasserbecken sehen. Dabei können sie mit ihren Händen im Wasser und an der Oberfläche durch bestimmte Gesten Einfluß auf das System nehmen, indem sie die Kreaturen festhalten, anlocken oder verjagen und ihnen auf diese Weise beim Jagen oder Fliehen zu Hilfe kommen. Wie auch in anderen, ähnlichen Installationen der beiden Künstler wird in dieser künstlichen Welt mit innovativen, natürlichen Interaktionsformen experimentiert und versucht, die Grenzen zwischen Realem und Irrealem zu verwischen. (vgl. [SOM97a], [SOM97b], [FAL97])

Durch die technische Vernetzung von Rechneranlagen ist die gleichzeitige Teilnahme mehrerer Zuschauer beziehungsweise Akteure möglich. Da VR verschiedenste herkömmliche Kunstformen und Medien vereinigen und eine globale Vernetzung realisieren kann, wird es auch *hypermediales Gesamtkunstwerk* genannt. VR als eigenständiges Medium ist gerade dabei, einen Platz in der Medienlandschaft zu finden: in der Kunst, aber auch in der Industrie, Forschung und Ausbildung zur Präsentation von 3D-Simulationen. Weitere Anwendungsfelder finden sich in der Unterhaltungsindustrie, nämlich für interaktive Filme und Spiele. Diese Entwicklung wird wahrscheinlich zu Lasten etablierter Einzelmedien wie beispielsweise dem Fernsehen, bei dem die wesentliche Interaktionsform im Ein- und Ausschalten von Kanälen besteht, geschehen, weswegen VR auch den sogenannten Substitutionsmedien („Ersetzungsmedien") zuzuordnen ist (vgl. [FAL95], S. 38). In Zukunft werden wir statt dessen an vorgegeben Szenarien teilnehmen können, selber als Akteur mit anderen Akteuren interagieren, Dinge verändern und die Handlung mitgestalten können. Die anderen Akteure werden programmgesteuerte Charaktere oder menschliche Teilnehmer sein; die Szenarien werden der Realität entnommen beziehungsweise entlehnt sein, aber auch völlig neue, abstrakte und irreale Erfahrungen sind denkbar. Es wird alles realisierbar, was man sich nur genau genug vorstellen kann, um es in glaubhafte Szenen umsetzen zu können. (vgl. Rötzer in [RÖT93], Zielinski in [RÖT93])

> *„In ihm werden sie mit anderen Menschen oder künstlichen Lebewesen interagieren, sich neue Identitäten zulegen oder kommunikative Erfahrungen machen können, die real nicht möglich wären."*
>
> (s. Rötzer In: [RÖT93], S.13)

Einerseits wird die Realität mit dem Virtuellen verschwimmen, andererseits werden die Menschen durch die Auseinandersetzung mit dem Virtuellen ein tieferes Verständnis für und vielleicht auch Bedürfnis nach realen Erlebnissen haben. Die Entwicklung im Freizeitverhalten in Richtung von Aktivitäten mit einem hohen Thrill-Wert wie Bungee-Jumping, Extrem-Rafting und ähnlichem ist bereits ein Indiz hierfür (vgl. Rötzer in [RÖT93]). In diesem Zusammenhang postuliert der Freizeitforscher Opaschowski die sogenannte Erlebnisgesellschaft:

> *„Der moderne Mensch sehnt sich nicht mehr nach Glück, sonder will Glück pausenlos erleben. Die Erlebnisinflation als massenhaft inszenierte Individualität ist vorprogrammiert."*
>
> (s. [OPA94], S. 22)

Das Kommunikationsmittel VR bietet sich ebenso als Plattform zum kooperativen Arbeiten in Rechnernetzen wie als alternative Benutzungsschnittstelle an. Diese beiden Aspekte sollen in den folgenden Abschnitten näher betrachtet werden, da sie den Kontext für diese Arbeit darstellen. Sie waren Gegenstand des studentischen Projektes NetzVision an der Universität Bremen, aus dem sich das Thema für meine Diplomarbeit ergeben hat (vgl. [NET96]).

## 2.2 Virtuelle Gemeinschaften

Virtuelle Gemeinschaften gehen aus einer Verknüpfung von Menschlichkeit und Technologie hervor. Computer und Rechnernetze stellen hierbei die Infrastruktur für ein neues Kommunikationsmedium - *Computer-mediated Communication (CMC)* - neben dem Telefon, dem Radio und dem Fernseher zur Verfügung. Kommunizieren Menschen mit Hilfe der CMC auf der Grundlage einer räumlichen Vorstellung, wird dieser vorgestellte (virtuelle) Raum *Cyberspace* genannt. (vgl. Rheingold In: [FAS94])

Novak definiert *Cyberspace* als:

> *„räumliche Visualisierung aller Informationen in allen globalen informationsverarbeitenden Systemen, die durch Verbindungen heutiger und zukünftiger Netzwerke realisiert wird und die eine vollständige Kopräsenz und Interaktion vieler Nutzer ermöglicht."*
>
> (s. Novak, M. (1991). In: [BEN91], S.255)

Treffen mehrere Menschen hinreichend regelmäßig in einem solchen Raum zusammen, bilden sich virtuelle Gemeinschaften als Gruppen von Menschen, die, während sie sich mit Worten oder ande-

ren Kommunikationsformen wie Gesten verständigen, soziale Beziehungen aufbauen. In virtuellen Gemeinschaften wird über alltägliche oder fachliche Themen diskutiert, Freundschaften werden geschlossen, gemeinsame Projekte werden durchgeführt oder Rollenspiele gespielt. Die meisten der zur Zeit existierenden virtuellen Gemeinschaften erzeugen die räumliche Illusion auf der Basis von Textbeschreibungen des aktuellen Ortes sowie der Möglichkeiten, an andere Orte zu gelangen. Die Teilnehmer können dann über Textbefehle den Ort wechseln und bekommen entsprechend eine neue Ortsbeschreibung.

```
Market Square [ N E S W ]
   You are standing on the market square, the famous Square of Midgaard.
A large, peculiar looking statue is standing in the middle of the square.
Roads lead in every direction, north to the temple square, south to the
common square, east and westbound is the main street.
A small sword lies here.
A note has been left here.
A Healer is standing here smiling happily.
His eyes shine in a soft blue![1]
```

Orte können als Treffpunkte für Leute, die zu bestimmten Themen diskutieren wollen, dienen. Außerdem werden mehrere Arten von Spielen angeboten, insbesondere MUDs (Multi User Dangeons).

> *„MUDs sind die Ausgangspunkte und ersten Implementationen des Cyberspace."*
> (s. Halbach In: [FAS94], S. 242)

In diesen Adventures (Abenteuerspiele) bewegen sich die Spieler in einer textuell beschriebenen virtuellen Welt, lösen gemeinsam oder gegeneinander Aufgaben, bestehen Abenteuer und sammeln dabei Erfahrungspunkte, die ihrer Spielfigur mehr Macht in Form einer Verbesserung ihrer Eigenschaften und Fertigkeiten verleihen. Die Teilnehmer können sich, beziehungsweise ihrer Spielfigur, eine Selbstbeschreibung hinzufügen, die die jeweils anderen Teilnehmer abrufen können. Durch diese Beschreibung, aber auch besonders durch Verhalten und Gespräche, kann der Spieler für seine Spielfigur eine Persönlichkeit mit konkreten Charaktereigenschaften nach seinem Belieben entwickeln, die ihn gegenüber den anderen Teilnehmern repräsentiert.

In virtuellen Gemeinschaften tritt in den Hintergrund, wer oder was man im wirklichen Leben (in real life) ist oder besitzt; die tatsächlichen räumlichen und zeitlichen Gegebenheiten verlieren an Bedeutung, da solche Gemeinschaften bei einer globalen Vernetzung aus Menschen[2] aus aller Welt bestehen können. Hervorgerufen durch den weltweit rapiden Anstieg von Menschen mit Netzzugang wird der Umgang in und mit virtuellen Gemeinschaften sowie die Implikationen der damit verbundenen sozialen Veränderungen an Bedeutung gewinnen. Schon heute verlagern viele Men-

[1] Aus dem MUD der Universität Bremen „Realms of Magic", b11.informatik.uni-bremen.de, Port 4000.

[2] allerdings nur solche mit Computer und Netzanschluß.

schen Teile ihres öffentlichen Lebens in Datennetze, treffen dort Freunde, diskutieren, besuchen ihre Bank oder Museen, gehen ihrem Beruf nach, kaufen ein oder recherchieren in Bibliotheken. Durch die schnell ansteigende Zahl der Nutzer und die vermehrte Nutzungsdauer werden aus Gemeinschaften Gesellschaften entstehen, die neuartige Regeln des Zusammenlebens erfordern. Neuartig deswegen, weil die Bürger beliebig über den Globus hinweg verteilt leben können, sich meist niemals persönlich (in real life) kennenlernen und - bei Bedarf - ihre wahre Identität geheimhalten können. (vgl. [FAS94], [LOE93], [GIB96])

Angesichts der immer schneller werdenden Rechnernetze und Computer mit 3D-Beschleunigung, die mehr und mehr auch Nutzern ohne Bereitschaft zum Eintippen von Befehlen zur Verfügung stehen, werden in Zukunft die heute noch vorherrschenden textbasierten virtuellen Welten durch solche mit intuitiv bedienbaren Benutzungsschnittstellen ersetzt werden, die ein Navigieren in dreidimensionalen, immersiven Welten in Echtzeit erlauben.

## 2.3 Metaphern

Graphische Benutzungsschnittstellen basieren meist auf der sogenannten Schreibtisch- oder Desktop-Metapher, wobei auf dem Desktop Objekte wie Papierkorb oder Ordner versammelt sind, deren Eigenschaften direkt manipuliert werden. Dagegen liegen VR-Systemen Raum und Zeit als Metaphern zugrunde. Räumliche Tiefe ist bei der Gestaltung von Benutzungsschnittstellen vor allem deswegen nützlich, weil sich mit ihrer Hilfe eine größere Zahl von Objekten auf gleicher Fläche übersichtlich unterbringen läßt. Dieser Effekt entsteht dadurch, daß sich überlappende oder überschneidende Objekte subjektiv trotzdem so erscheinen, als wären sie vollständig präsent. In VR-Systemen ergibt sich zusätzlich die Möglichkeit, um Objekte, die gerade die Sicht verdecken, herumzunavigieren, damit man Objekte im Hintergrund oder am Rande näher betrachten kann. Da bei VR-Systemen der Benutzer direkt in einer auf ihn wirkenden Szene handelt und nicht nur beobachtend an einem Gegenstand hantiert, ist der entdeckende Charakter des Operierens stärker ausgeprägt. (vgl. [EBE94], S.30ff und S.136-139)

Im besonderen eignen sich VR-Systeme als Plattform für kooperatives Arbeiten, da die Aktionen der anderen Teilnehmer „live“ mitverfolgt werden können. In diesem Zusammenhang bietet VR ein intuitiv bedienbares Gerüst für verschiedene Kommunikationsformen und für den Austausch sowie das gemeinsame Bearbeiten von Dokumenten. Die reale räumliche Entfernung der Projektteilnehmer tritt dabei zugunsten der Geschwindigkeit der Netzwerkverbindung in den Hintergrund. (vgl. [NET96])

# 3 Modelle menschlicher Verhaltenssteuerung

Will man künstliche Akteure in virtuellen Welten mit einem Verhalten ausstatten, das vom Standpunkt menschlicher Betrachter aus glaubhaft erscheint, so ist es angebracht, verhaltenspsychologische Modellansätze zum Verhalten des Menschen selbst in die Untersuchungen dieser Arbeit einzubeziehen. Die dort verwendeten Erklärungsmodelle finden nicht auf neuronaler Ebene statt, sondern abstrahieren von den Einzelheiten der Speicherung von Wissen und Fertigkeiten sowie der konkreten Kontrolle über Muskeln. In den beiden folgenden Abschnitten über motorische Programme und die Schematheorie werden zwei weit verbreitete Konzepte vorgestellt, die Ideen bei der Steuerung automatisierter Charaktere liefern können.

## 3.1 Motorische Programme

In der einschlägigen Diskussion über die Herstellung von Bewegungen beim Menschen herrscht heute die Meinung vor, daß vor der Bewegung selbst eine Art Programm zusammengestellt wird, das den Bewegungsablauf eines längeren Zeitraumes vorgibt. Für diese These sprechen vor allem Versuche, die belegen, daß koordiniertes Verhalten auch dann möglich ist, wenn die sensorische Rückmeldung gestört ist. Außerdem hat sich aus Reaktionszeit-Messungen eine systematische Abhängigkeit der Latenzzeit zwischen einer Bewegungsaufforderung und seiner Durchführung von der Art der Bewegung ergeben, woraus man folgert, daß der komplette Bewegungsablauf vor der Ausführung - jedenfalls in einem gewissen Rahmen - vorprogrammiert wird. Solche motorische Programme enthalten Kommandos über spezifische Muskelbewegungen, aber auch Pläne auf abstrakterer Ebene wie die räumlich-zeitliche Charakteristik von Bewegungen unabhängig von den beteiligten Muskelgruppen.

Motorische Programme können aus in der Vergangenheit geübten Fertigkeiten gewonnen oder neu zusammengestellt werden. Fertigkeiten sind in diesem Sinne Programme, die sich bewährt haben und im Gedächtnis gespeichert werden, um sie in Zukunft schneller zur Verfügung zu stellen. Durch Üben können bestimmte Bewegungsklassen mit variablen Anteilen abstrahiert werden, die mit konkreten Parametern belegbar sind. Inwieweit diese Abstraktion stattfinden kann, hängt insbesondere vom angewendeten Lernverfahren und -aufwand ab. Motorische Parameter sind zum Beispiel Kraft, Schnelligkeit, Richtung oder räumliche Ausdehnung einer Bewegung.

Versuche haben ebenfalls nahegelegt, daß motorische Programme hierarchisch strukturiert vorliegen. An der Spitze der Hierarchie befinden sich die sogenannten *generalisierten Programme*, die den Bewegungsablauf auf einer groben, von Muskelkommandos abstrahierten Ebene vorgeben; auf der untersten Stufe befinden sich konkrete Anweisungen für Muskelkontraktionen. Die Art der Hierarchiebildung wird durch das Lernverfahren bestimmt. Die Strukturierung der motorischen Programme auf die beschriebene Weise bewirkt eine signifikante Speicherersparnis und eine Sicherung der Konsistenz des motorischen Verhaltens. (vgl. [ZIE88] S. 371-388, [NEU89] S. 32-49, [PRI84], [SCH82] S. 285-333)

## 3.2 Die Schematheorie

Die Schematheorie von R. A. Schmidt (1975) versucht zu erklären, wie Mechanismen der Verhaltenssteuerung repräsentiert und gelernt werden, so daß diese auch unter veränderten Bedingungen erfolgreich angewendet werden können. Das Konzept des *motorischen Programmes* wird dabei in ein umfassendes Modell zur Verhaltenssteuerung integriert und um Überlegungen zur Handlungsauswahl und Adaption erweitert.

> *„Ein Schema ist das Charakteristikum einer Population von Objekten und besteht aus einem Satz von REGELN, die als Anweisungen für die Erzeugung eines Prototypen dieser Population (dem Konzept) dienen."*
>
> (s. [EVA67], S. 87)

Menschen entwickeln durch Übung Regeln (Schemata) über ihr eigenes Verhalten. Ein solches Schema enthält die Ergebnisse und Parameter aller vorangegangenen Ausführungen dieses Schemas (Welche Parameter haben zu welchen Konsequenzen geführt? Wurde das Ziel erreicht?), so daß im erneuten Anwendungsfall dieses Wissen über vergangene Situationen zur Wahl der Parameter herangezogen werden kann.

Gelernte Verhaltensakte können auch unter neuen Bedingungen ausgeführt werden, also muß es Mechanismen geben, die verallgemeinerbare Relationen beziehungsweise Konzepte extrahieren und Verhaltensakte an die aktuelle Situation anpassen. Die Verhaltensakte sind in flexiblen, generalisierten Verhaltensschemata gespeichert, die von den Kontrollmechanismen geeignet parametrisiert werden. Können anhand bestehender Schemata nicht die gewünschten Ergebnisse erzielt werden, so werden Subschemata, also Variationen vorhandener Schemata, mit spezifischen Eigenschaften hergestellt.

Motorische Handlungsschemata enthalten folgende Informationen:

- Anfangsbedingungen
- Handlungsspezifikationen für das motorische Programm
- Sensorische Konsequenzen erzeugter Bewegungen
- Ergebnisse der Handlungsausführung

(vgl. [SCH82], S. 493 u. 592-601 und [SCH90])

### 3.2.1 Anfangsbedingungen

Nach jeder Handlung werden die Anfangsbedingungen, die zur Planung verwendet worden waren, gespeichert. Dies sind Informationen über den Anfangszustand des Muskelsystems und der Umwelt, zum Beispiel Position der Körperteile und des Körpers im Raum sowie visuelle und auditive Information über den Zustand der Umwelt. Die gespeicherten Ausgangsbedingungen können bei einer erneuten Handlungsplanung wiederverwendet werden, indem sie in Relation zu den jeweils erreichten Resultaten gesetzt werden. (vgl. [SCH90], S. 29)

### 3.2.2 Handlungsspezifikation

Die Handlung selbst besteht in der Ausführung eines motorischen Programmes für die Herstellung von Muskelkommandos. Diese Programme können durch Variation von Parametern modifiziert werden. Reaktionszeit-Versuche haben ergeben, daß alle Bewegungsparameter einer Handlung schon in der Planungsphase dieser Handlung spezifiziert werden und der Bewegungsablauf damit weitestgehend festgelegt wird. (vgl. [SCH90], S. 30)

### 3.2.3 Sensorische Konsequenzen

Während und nach der Handlung werden Rückkopplungsreize aus der Umwelt sowie von der Bewegung selbst hergestellte sensorische Informationen gesammelt und abgespeichert. Diese Daten werden benötigt, um zu bewerten, ob das angestrebte Resultat wirklich erreicht wurde. (vgl. [SCH90], S. 30)

### 3.2.4 Ergebnisse der Handlungsausführung

Nach Abschluß einer Handlung werden die gesammelten sensorischen Konsequenzen in Kenntnis des Resultates in Bezug zum ursprünglich beabsichtigten Ergebnis gesetzt. Dabei ist die Genauigkeit der Ergebnisinformation direkt abhängig von der Fülle der Rückkopplungsinformation. Bei fehlendem Feedback kann keine Ergebnisinformation ermittelt und abgespeichert werden, falls nicht das Ergebnis der Handlung schon von vornherein vorhersehbar ist. (vgl. [SCH90], S. 30)

### 3.2.5 Schemabildung

Werden mehrere Bewegungen des gleichen Typs gemacht, können Beziehungen zwischen den Anfangsbedingungen, der Handlung, der Wahrnehmung und dem Ergebnis abstrahiert werden. Die Stärke dieser Beziehungen erhöht sich durch jede Bewegung desselben Typs. Diese Beziehungen für einen bestimmten Bewegungstypen definieren ein Schema.

Die Beziehung, die auch Erinnerungsschema oder Recallschema genannt wird, sorgt dabei für die Zuordnung, unter welchen Bedingungen welche Parameter spezifiziert werden müssen, um ein bestimmtes Ziel zu erreichen. Das Wiedererkennungsschema (Recognitionschema) enthält die Relation zwischen den propriozeptiv und exterozeptiv erlebten Wahrnehmungen und dem Erfolg der Anwendung. Je vielfältiger die Übungssituationen, um so effektiver wird das Schema gelernt. Während der dabei ablaufenden Abstraktion wird festgelegt, inwieweit die Bewegung beziehungsweise das motorische Programm durch Parameter variiert werden kann. (vgl. [SCH90], S. 31-32)

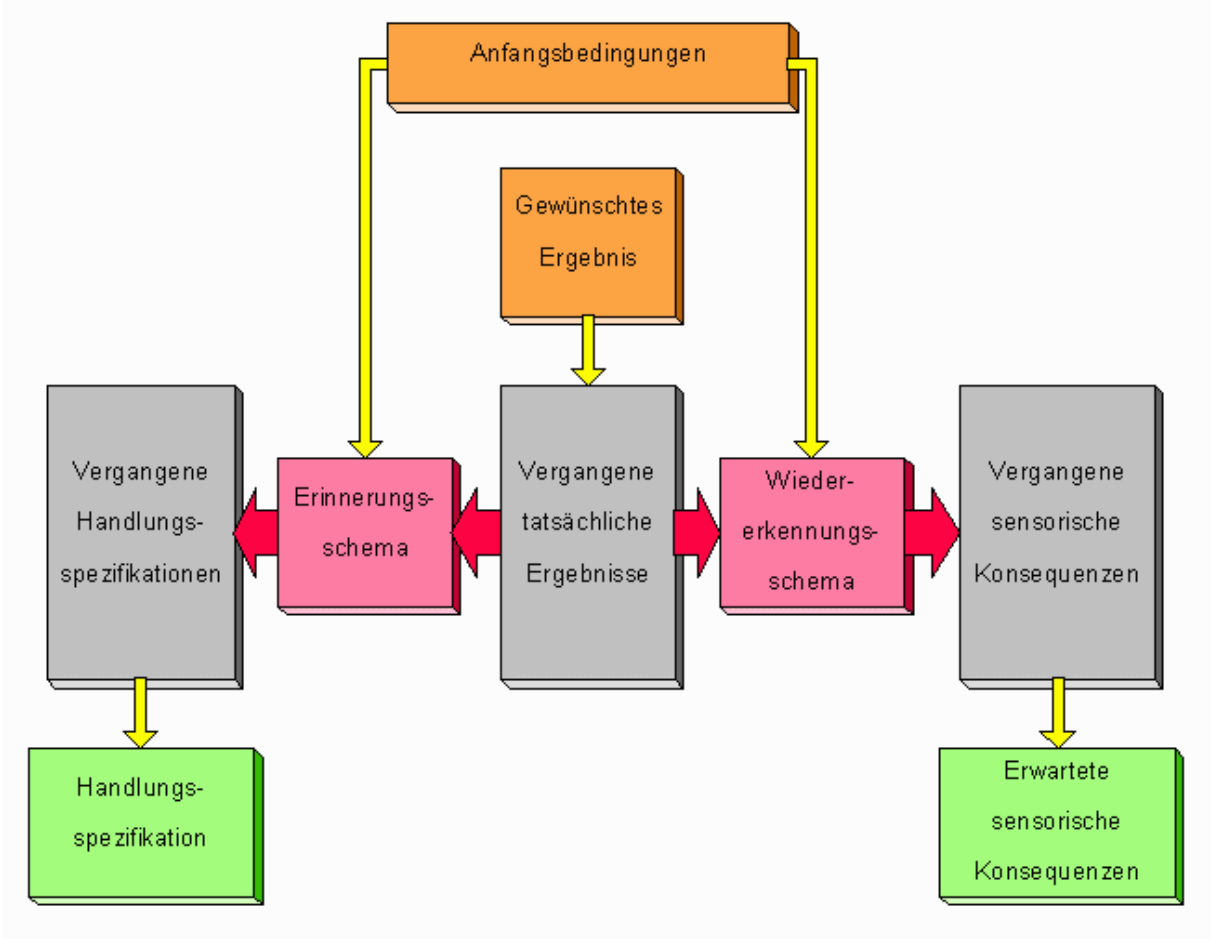



Abbildung 5: Erinnerungs- und Wiedererkennungs-Schema
(vgl. [SCH90], S. 31)

### 3.2.6 Herstellung von Bewegungen

Ist für einen auszuführenden Bewegungstyp schon ein Schema vorhanden, wird anhand des gewünschten Ergebnisses, der aktuellen Anfangsbedingungen und dem Erinnerungsschema ermittelt, welcher Parameter wie zu spezifizieren ist, um das gewünschte Ergebnis zu erzielen. Dabei muß zur erfolgreichen Bewegungsherstellung die konkret vorliegende Kombination von gewünschtem Ergebnis und Anfangsbedingungen vorher noch nicht geübt worden sein, da das Erinnerungsschema auf der Grundlage früherer erfolgreicher Parameterspezifikationen eine Interpolation herstellen kann. Auf diese Art und Weise kann das erlernte Schema auf immer neue Situationen angewendet werden und zu neuartigen Bewegungen beziehungsweise allgemein Handlungen führen. (vgl. [SCH90], S. 32 u. 38-40)

### 3.2.7 Rückkopplung

Während der Planung einer Handlung wird gleichzeitig ermittelt, welche sensorischen Konsequenzen bei Ausführung der Handlung zu erwarten sind. Grundlage hierfür sind früher erlebte sensorische Konsequenzen in Beziehung mit den dazugehörigen Ausgangsbedingungen und dem angestrebten Resultat. Man unterscheidet die propriozeptive und exterozeptive sensorische Rückkopplung: die propriozeptive ist die Wahrnehmung der körpereigenen Nervenzellen in Muskeln und Gelenken, die exterozeptive bezieht sich auf Reize der Sinne (visuelle, auditive) aus der Umgebung. Durch den Vergleich von antizipierter und tatsächlicher Wahrnehmung können während und nach Bewegungen Fehler bestimmt werden. Die Einbeziehung von Fehlerrückkopplung kann genutzt werden, um bestehende Schemata zu verbessern und so die Genauigkeit von Bewegungen zu erhöhen oder um neue Schemata auszuprägen. Mit zunehmendem Lernfortschritt verringert sich die Bedeutung der Rückkopplung. (vgl. [SCH90], S. 37-38 u. 25-26)

> *„Die NOTWENDIGKEIT peripheren Feedbacks kann als umgekehrt proportional zur Fähigkeit des ZNS[3] betrachtet werden, jeden wichtigen Aspekt der kommenden Bewegungen vorausschauend zu bestimmen."*
>
> (s. MacNeilage, MacNeilage (1973); zit. n.: [SCH90], S. 25)

[3] Anmerkung des Autors: ZNS für Zentrales Nervensystem

### 3.2.8 Zusammenfassendes Modell

Zusammenfassend ergibt sich folgendes Modell für die Schematheorie:

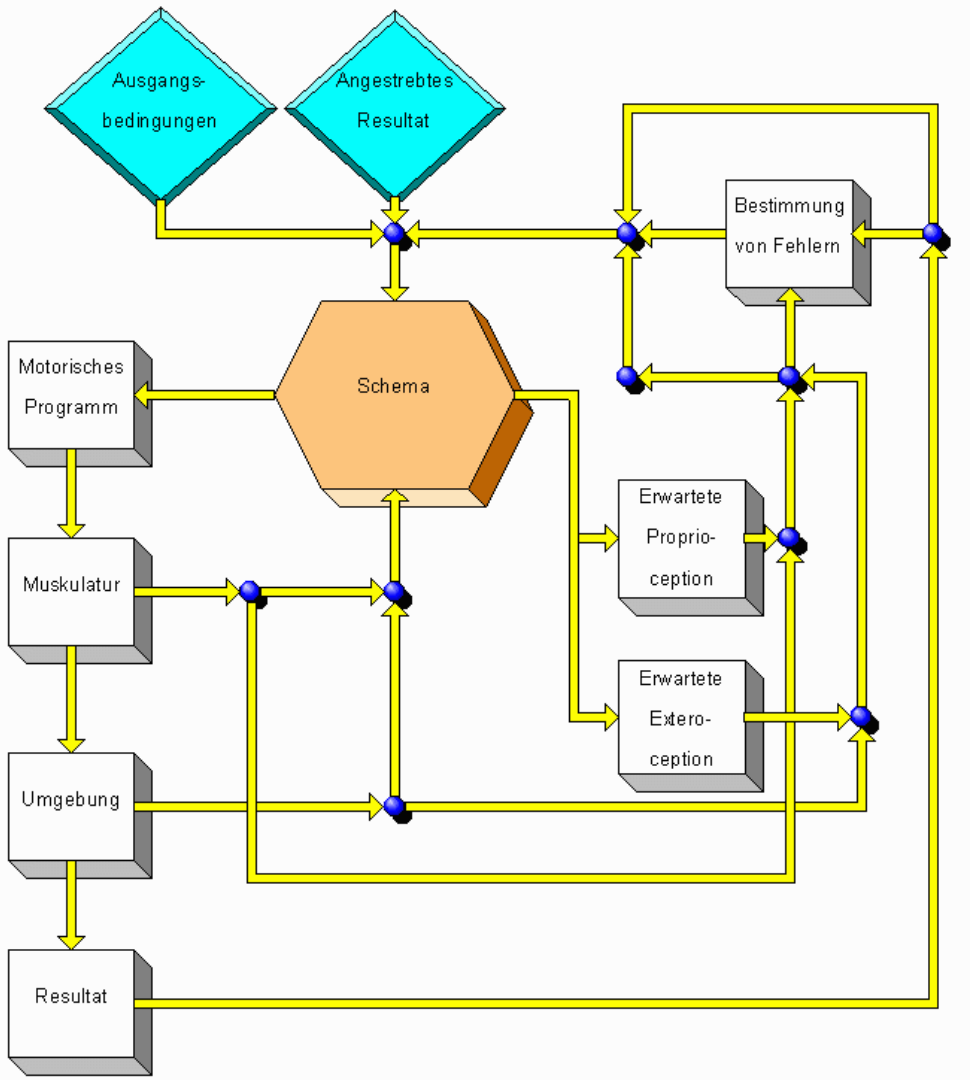



Abbildung 6: Schematheorie

Schaubild zur Schematheorie frei nach [HOF93], S. 191)

# 4 Komplexe Animation von 3D-Objekten

In diesem Abschnitt sollen verschiedene, bekannte Ansätze zur 3D-Computeranimation behandelt werden. Die Repräsentationsformen der Bewegungsabläufe reichen dabei von komplett Keyframe[4]-bezogenen Spezifikationen bis zu ökonomischeren, symbolischen Beschreibungen. Welche dieser Repräsentationsformen gewählt werden sollte, hängt im Wesentlichen davon ab, ob die Animation im voraus planbar ist oder ob sie sich erst zur Laufzeit als Ergebnis einer Simulation und/oder Benutzereingabe ergibt. Die im folgenden vorgestellten Techniken sind dabei grundsätzliche Ideen, die in konkreten Anwendungen oft in Kombinationen auftreten. (vgl. [WAT92], S. 339-394, [FOL93], S.1057-1081)

## 4.1 Hierarchiebildung

Animationen sind in der Regel sehr komplexe Bewegungen von ebenfalls sehr komplexen Objekten. Ein Objekt mit sechs Bewegungs-Freiheitsgraden (DOF für Degrees of Freedom) über fünf Sekunden bei 30 Bildern pro Sekunde zu animieren, bedeutet zum Beispiel, 9000 Zahlen angeben zu müssen. Um diesem Aufwand zu entgehen, ist es erforderlich, Animationssysteme mit Mechanismen zur Hierarchiebildung auszustatten. Jede Hierarchieebene bildet einen parametrisierten Input auf einen parametrisierten Output ab, der wiederum als Input für die darunterliegende Ebene gilt. Auf der untersten Ebene existieren nur noch die Primitiven, die direkt dem Renderer (Modul zur graphischen Ausgabe) übergeben werden können. Durch die Hierarchiebildung können Bewegungsabläufe auf beliebig abstraktem Niveau spezifiziert werden, und der Computer übernimmt die Übersetzung in Primitiven und die Berechnung der Parameter beziehungsweise Vektoren. Hierarchiebildung kann zum Beispiel durch das Konzept der Prozedur erreicht werden, das die Gruppierung anderer Prozeduren unter Einbeziehung von Parameterwerten erlaubt. Als nützlich hat sich auch das Konzept der Akteure erwiesen: im Sinne der objektorientierten Programmierung werden Bewegungen nicht separat von den Objekten betrachtet, sondern sind diesen als Eigenschaft zugeordnet. So „weiß“ zum Beispiel ein Pferd-Objekt, wie es beim Galopp seine Beine zu koordinieren hat. Bei der Animation etwa einer Reiterszene braucht man sich dann nicht mehr um Pferde-Interna zu kümmern. ([WAT92], S. 339-394, [FOL93], S.1057-1081)

[4] Keyframe = englisch für Schlüsselbild

## 4.2 Interpolation von Schlüsselszenen

Keyframe-Systeme haben ihren Namen von der traditionellen Animationsproduktion, die zuerst von Walt Disney eingesetzt wurde. Diese besteht darin, daß fähige Zeichner den Ablauf der Animation und das Aussehen der Akteure durch das Herstellen markanter Schlüsselbilder (keyframes) festlegen, während weniger begabte Zeichner auf dieser Grundlage die Zwischenbilder (inbetweens) produzieren. Die Emulation dieser Technik mit Hilfe des Computers war die Idee der frühen Animationstools, bei denen das Zeichnen der Zwischenbilder durch vom Computer berechnete Interpolationen ersetzt wurde. Das Problem und die Herausforderung bei dieser Technik ist, möglichst wenig Schlüsselbilder herstellen zu müssen und trotzdem die richtigen Interpolationen zu finden. Hierfür gibt es verschiede Ansätze, wobei die meisten auf Splines basieren. ([WAT92], S. 339-394, [FOL93], S.1057-1081)

## 4.3 Motion Tracking

Um schnell oder in Echtzeit Animationen-Akteure mit natürlichen Bewegungen auszustatten, können Bewegungen realer Schauspieler als Vorbild herangezogen werden. Die automatische Erfassung von Bewegungen dieser Schauspieler wird Motion Tracking genannt. Grundsätzlich gibt es verschiedene Methoden, um Motion Tracking zu realisieren. Bisher ist das Aufbringen von Indikatoren, deren räumliche Position durch Sensoren ermittelt werden kann, auf markanten Stellen des Körpers am weitesten verbreitet. Dieses Thema wurde u.a. von Ginsberg und Maxwell untersucht, die Motion Tracking verwendet haben, um eine graphische Marionette zu steuern. Motion Tracking kann helfen, schwer modellierbare Bewegungsabläufe beschreibbar zu machen. In diesem Zusammenhang ist der Aufbau von Bewegungsbibliotheken/-datenbanken im Gespräch. (vgl. Ginsberg u. Maxwell In: [BAD86], S.303-311, [WAT92], S. 339-394, [FOL93], S.1057-1081)

## 4.4 Kinematik und Inverse Kinematik

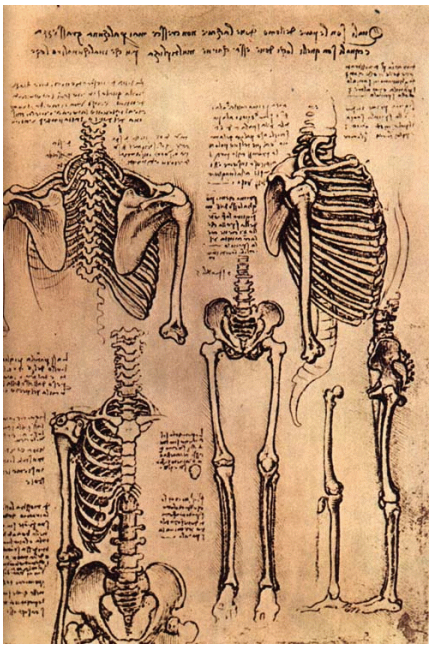


Abbildung 7: Anatomische Zeichnung von Leonardo da Vinci

(North Wing - Leonardo. http://cellini.leonardo.net/museum/north.html am 20.5.1997)

Kinematik ist die Lehre von Bewegungen ohne Rücksicht auf die sie verursachenden physikalischen Kräfte. In diesem Zusammenhang werden geometrische und zeitliche Beziehungen zwischen Positionen, Geschwindigkeiten und Beschleunigungen bewegter Körper untersucht, insbesondere im Rahmen der Getriebelehre und Robotik. Eine wichtige Kenngröße ist hier die Anzahl der Freiheitsgrade (Degrees of freedom, *DOFs*), beispielsweise die Anzahl der beweglichen Gelenke eines Roboterarmes. Die Kinematik stellt unter anderem Regeln bereit, mit denen bei gegebenen Gelenkzuständen beziehungsweise Bahnen des besagten Roboterarmes die Positionen oder Bahnen des freien Endes (etwa der Greifhand) ermittelt werden können. Das freie Ende eines solchen Armes wird End-Effector genannt. Bei der Inversen Kinematik ist die Problemstellung genau umgekehrt: Bei vorgegebener Zielposition oder Bahn des End-Effectors sollen die Winkel der Gelenke berechnet werden. Je mehr DOFs dabei zu betrachten sind, um so komplexer ist natürlich das Problem. In den seltensten Fällen gibt es genau eine Lösung, sondern entweder gar keine oder viele, aus denen eine bestimmte ausgewählt werden kann. Zur Einschränkung des Lösungsraumes werden dem System üblicherweise zusätzliche Zwänge (*Contraints*) beigefügt, die zum Beispiel die Minimierung des Energieaufwandes oder die Begrenzung des Bewegungsspielraumes von Gelenken bewirken.

Verfahren aus der Inversen Kinematik werden beim Herstellen von Computeranimationen angewandt, da mit ihrer Hilfe - etwa bei der Steuerung einer menschenähnlichen Figur - nicht alle Winkel der Gelenke vorgegeben werden müssen, sondern entsprechend der Contraints des Körperbaus berechnet werden können. Der Aufwand des Animationsdesigns kann so entscheidend verringert werden; die Kontrolle über die zu bewegenden Figuren kann dadurch auf einer zielgerichteteren Ebene stattfinden. Allerdings ist der Berechnungsaufwand bei einer großen Anzahl von DOFs sehr hoch, womit dieses Verfahren erst in letzter Zeit bei Echtzeitanwendungen zum Einsatz kommen konnte. (vgl. [WAT92], S. 339-394, [FOL93], S.1057-1081, [BAD95])

## 4.5 Physikalisch basierte Animation

Im Gegensatz zur Kinematik werden in der Dynamik solche Bewegungen untersucht, die durch physikalische Kräfte, wie zum Beispiel die Schwerkraft, bewirkt oder beeinflußt werden. Für den Animator bedeutet dies, daß er viele Bewegungen nicht selber spezifizieren muß, da sie sich direkt aus den wirkenden Gesetzen ergeben. So würde etwa ein Gummiball wegen der Schwerkraft automatisch einen schiefen Tisch herunterrollen, dann fallen, je nach Fußboden wegen seiner Elastizität einige Male auftrumpfen und schließlich reibungsbedingt liegenbleiben. Der Preis für die derart gewonnene Vereinfachung für das Animationsdesign ist, daß die physikalischen Eigenschaften der Umwelt und der Objekte modelliert werden müssen. Wie zur Kinematik gibt es auch zur Dynamik ein inverses Gegenstück. (vgl. [FOL93], S.1057-1081, Murthy u. Raibert In: [BAD86], [BAD95])

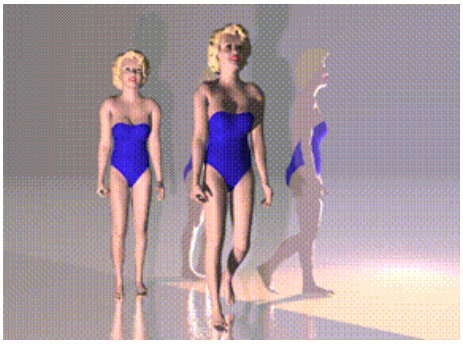


Abbildung 8: Thalmanns Marilyn Monroe Simulation

Der Einsatz der Gesetze der Dynamik sind Beispiele für physikalisch basierte Animation. Ebenfalls in dieses Gebiet fallen spezielle Simulationen der Bewegungen von Haut, Muskeln, Haaren und Kleidung. Bekannte Pioniere sind hier die Schweizer Nadja Magnenat-Thalmann und ihr Ehemann Daniel, die seit Jahren versuchen, perfekte Modelle virtueller Akteure zu schaffen, wobei die präferierte Versuchsperson Marilyn Monroe ist. Die wesentlichen Probleme sind, Haaren (durchschnittlich 150.000!) und Kleidung unter den Einflüssen von Wind, Gravitation und Kollision ein natürliches Aussehen und Bewegungsverhalten zu geben; die perfekte Animation menschlicher (oder tierischer) Fortbewegung erfordert die Einbeziehung der jeweiligen anatomischen und physiologischen Eigenschaften. Neuerdings beschäftigen sich die Thalmanns auch mit der aufgabenorientierten (☞ 4.7 Aufgabenorientierte Animation) und autonomen (☞ 5 Autonome Agenten) Verhaltenssteuerung ihrer Akteure. (vgl. [SPE95], [THA93])

Solcherart komplexe Simulationen sind heute nicht in Echtzeit durchführbar, vielmehr rechnen oft ganze „Farmen“ von Graphikrechnern wochenlang an Animationssequenzen für Spezialeffekte in Spielfilmen oder Werbesendungen. Das ESPRIT-Projekt HUMANOID, bei dem auch die Thalmanns mitarbeiten, hat es sich zum Ziel gesetzt, ein paralleles Echtzeit-System mit vereinfachten, aber schnelleren Algorithmen für die Simulation virtueller Menschen zu entwickeln. (vgl. [THA95])

## 4.6 Animation mittels neuronaler Netze

Neuronale Netze sind parallele Verarbeitungseinrichtungen oder Programmiermodelle, die von Teilstrukturen der Informationsverarbeitung in Lebewesen abgeleitet sind. Besondere Eigenschaften sind ihre Anpassungs- und Lernfähigkeit. Neuronale Netze bestehen aus Verarbeitungselementen (Neuronen) mit jeweils einem Ein- und einem Ausgang. Die Verarbeitungselemente erzeugen mittels einer Wichtungsfunktion aus einem Eingangsmerkmal ein Ausgangsmerkmal. Ein neuronales Netz „lernt", indem die Abweichungen zwischen tatsächlichem und gewünschtem Muster die Wichtungsfaktoren adaptieren. Neuronale Netze werden unter anderem in Bereichen der künstlichen Intelligenz und zur Sprach- und Bilderkennung eingesetzt.

Diese Lern-Eigenschaft neuronaler Netze kann bei der Animation zur Automatisierung unkreativer, sich wiederholender, komplexer Aufgaben eingesetzt werden. Beispielsweise kann das Eingangsmerkmal aus einem Sprachmuster und das Ausgangsmerkmal aus den Parametern einer Mundanimation bestehen, die dieser Tonaufnahme entsprechen. Das Netz kann lernen, welche Laute zu welchen Mundstellungen gehören, und die Mundbewegungen zu neuen Sprachaufnahmen selbsttätig generieren. ( vgl. [PRI94], S. 99-111)

## 4.7 Aufgabenorientierte Animation

Aufgabenorientierte oder zielgerichtete Animationssysteme versuchen die Arbeitsweise zwischen Regisseur und Schauspieler nachzuempfinden. Mit Hilfe von Anweisungen kann der Animator einer Figur vorgeben, „was sie tun soll". Denkbar sind zum Beispiel Befehle wie: "Nimm das Glas und stelle es auf den Tisch" oder „gehe links durch die Tür". (vgl. [SPE95])

Vorausgesetzt werden entsprechend vordefinierte Bewegungseinheiten. Diese Einheiten sind üblicherweise wegen der potentiellen Komplexität der auszuführenden Bewegungen hierarchisch strukturierbar.

Um aus vorgegebenen Zielen oder Aufgaben konkrete Bewegungen abzuleiten, ist ein Zuordnungsprozeß notwendig, der Handlungen mit zu erwartenden Resultaten in Verbindung bringt. David Zeltzer von der Computer Graphics Research Group der Universität Ohio schlägt hierfür den Einsatz wissensbasierter Verfahren aus der klassischen KI vor. Dann nämlich sei es möglich, automatisch von globalen Zielen auf Teilziele zu schließen und so Handlungen sinnvoll zusammenzustellen. (vgl. Zeltzer In: [BAD86], S. 318-324)

## 4.8 Verhaltensgesteuerte Animation

1987 wurde in dem computeranimierten Film „Stella & Stanley – Breaking the Ice“ zum ersten Mal der Ansatz verhaltensgesteuerter Animation verfolgt. Der Animationsvorgang wurde für Fische und Vögel automatisiert, indem man für die einzelnen Individuen verhaltensspezifische Bewegungsmuster, wie zum Beispiel das Schwimmen beziehungsweise Fliegen in einem Schwarm, implementierte. Die synthetischen Tiere agieren und reagieren in ihrer computergenerierten Umwelt, basierend auf der für sie definierten Verhaltensgrundlage. Die Arbeit des Animators beschränkte sich auf die Animation eines Leittieres und auf die Gestaltung der dramaturgischen Bewegungsabläufe.

Die Animationsarbeit wird also erleichtert, indem die Gesamtbewegung auf einfache, autonome Objekte (☞ 5 Autonome Agenten) aufgeteilt und so strukturiert wird. Für einfache Modelle wird die Verhaltensdefinition dadurch wesentlich erleichtert. Für differenziertere Anwendungen wird jedoch das sich aus den vielfältigen Wechselwirkungen zwischen den Individuen ergebene semantische Netzwerk sehr komplex. Agenten sind zur Zeit Gegenstand zahlreicher Forschungsarbeiten. ([MAE95a], [MAE95b])

Abbildung 9: Stärkesimulation bei Norman Badlers „Jack“

Norman Badler vom *Center for Human Modeling and Simulation* an der Universität von Pennsylvania beschäftigt sich in seinen Forschungen mit virtuellen Darstellern in militärischen und zivilen Echtzeit-Simulationen unter besonderer Berücksichtigung von zielgerichtetem Verhalten sowie anatomischen und physiologischen Eigenschaften. In diesem Zusammenhang entstand das produktreife System namens „Jack“, bei dem das Verhalten von Figuren als Resultat konkurrierender Ziele (zum Beispiel Erreichen eines Zielortes ⇔ Kollisionsvermeidung) entsteht. „Jack“ kann sich selbständig sogar in unebenem und unbekanntem (virtuellem) Gelände bewegen und dabei durch Verlagerung seines Schwerpunktes oder durch Schritte das Gleichgewicht halten. Natürlich wirkende Bewegungsabläufe werden mit Hilfe detaillierter Kraftsimulation erreicht, bei der das Bewegungspotential von Figuren auf der Grundlage eines Muskel- und Skelettmodells ermittelt wird. ([BAD95], [TRA96])

# 5 Autonome Agenten

## 5.1 Einordnung

Die Erforschung künstlicher, autonomer Agenten wird gemeinhin als Schnittmenge der Wissenschaftsfelder *Artificial Life* (AL) und *Artificial Intelligence* (AI) angesehen. Während sich AL mit der Modellierung natürlicher Lebensformen beschäftigt, um zum Beispiel Evolutionsmechanismen oder Ökosysteme besser zu verstehen, versucht AI, Computer mit Intelligenz nach menschlichem Vorbild auszustatten. Das gemeinsame Interesse besteht in der Synthese von künstlichen Geschöpfen mit menschenähnlicher Intelligenz. Diese sollen sich nach Möglichkeit selbständig in ihrer Umgebung zurechtfinden, sich sinnvoll sowie zielbewußt verhalten und eventuell auch neue Verhaltensweisen dazulernen. (vgl. [MAE95a], [MAE95b])

> *„Autonomous agents are computational systems that inhabit some complex, dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks that they are designed for."*
>
> (s. [MAE95a])

Autonome Agenten können in unterschiedlichen Ausprägungen auftreten: als Roboter, die sich in der realen Welt aufhalten, als zwei- oder dreidimensionale animierte Figuren in computergenerierten Umgebungen, oder als sogenannte *Knowbots*, die als körperlose Helfer Softwaresysteme bewohnen. Jede dieser Ausprägungen hat eigene typische Einsatzbereiche, beispielsweise werden autonome Roboter zur Arbeit in für Menschen schwer zugänglichen oder gefährlichen Gegenden eingesetzt, animierte Figuren wer-

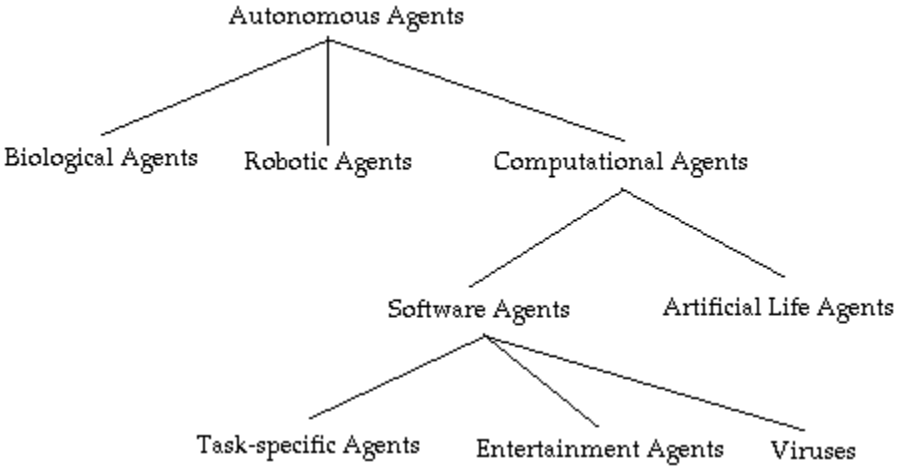



Abbildung 10: Klassifikation Autonomer Agenten
(vgl. [FRA96])

den in Simulationsprogrammen verwendet und *Knowbots* machen sich in komplexen Softwaresystemen als *Interface Agents* zur Unterstützung der Benutzer nützlich. Während Agenten mit computergenerierter Visualisierung früher vorwiegend in militärischen Anwendungen wie Gefechtssimulationen auftraten, dringen sie heute in zivile Bereiche und - im Besonderen - in die Unterhaltungsindustrie in Form von Darstellern in Filmen, virtuellen Welten oder Computerspielen ein (vgl. [MAE95a]). Gerade wenn Echtzeit-Interaktion mit der Umwelt gefordert ist, sind agentenbasierte Ansätze erfolgversprechend, da sie aufgrund einer spezifischen Programm-Architektur zur Beschreibung reaktiven Verhaltens besonders geeignet sind.

## 5.2 Architektur

Agenten-basierten DV-Systemen liegt eine charakteristische Architektur zugrunde: keine zentrale Entscheidungsinstanz ist für die Erfüllung funktionaler Anforderungen zuständig, vielmehr wird die Entscheidungskompetenz auf eigenständige Module (Agenten) aufgeteilt, die jeweils für einen kleinen Teil der Gesamtaufgabe Experten sind. Diese Module laufen als parallele Prozesse auf einem oder mehreren Rechnern und kommunizieren miteinander und mit ihrer Umwelt durch möglichst einfache Botschaften. Innerhalb der Module wird das Verhalten mit Hilfe von Regeln festgelegt, die immer dann aufgerufen werden, wenn Botschaften von anderen Modulen oder von Sensoren eintreffen. Während der Abarbeitung der Regeln können wiederum Botschaften an andere Agenten - etwa Aufträge zur Bewältigung von Teilaufgaben - geschickt werden, so daß semantische Netze von Wechselwirkungen zwischen den Agenten entstehen. Um das Verhalten von Agenten vielfältig zu gestalten, sind redundante Regeln sinnvoll, das heißt, daß für gegebene Situationen mehrere alternative Regeln verfügbar sind. Aus diesen Regeln kann dann eine zufällig ausgewählt und zur Anwendung gebracht werden oder, wenn eine Regel fehlgeschlagen ist, eine zweite versucht werden. Je mehr Redundanz eingebaut ist, um so robuster und interessanter wird das Verhalten des Agenten. (vgl. [MAE95a], [MAE95b])

Diese Eigenschaften - nämlich Vielfältigkeit und Robustheit - sowie die Illusion von emotionalem und sozialem Verhalten sind gerade beim Einsatz von Agenten in virtuellen Welten wichtiger als ein hoher Grad an Intelligenz; entscheidend ist die Glaubwürdigkeit der Handlungen.

Ausgehend von der speziellen Architektur lassen sich folgende Anforderungen an autonome Agenten aufstellen (vgl. [MAE95b]):

- Wahrnehmung der Umwelt, die unter Umständen ständigen Veränderungen ausgesetzt sein kann,

- Auswahl von Handlungen, die sinnvoll sind, um in der jeweils aktuellen Situation einem bestimmten Ziel näherzukommen,
- Kontrolle von Bewegungen abhängig von der auszuführenden Handlung,
- Anpassung von Verhaltensweisen im Sinne einer Optimierung auf der Grundlage von gemachten Erfahrungen,
- Kommunikation mit anderen Agenten und menschlichen Akteuren.

Von diesen Anforderungen werfen das Auswählen von Handlungen und das Lernen aus Erfahrung (Anpassung) die größten Probleme auf und sollen in den folgenden Abschnitten näher beleuchtet werden.

## 5.3 Auswahl von Handlungen

Bei Agenten, die mehrere, oft konkurrierende und über die Zeit veränderliche Ziele verfolgen und dabei jeweils verschiedene denkbare Aktionen im Repertoire haben, ist die Auswahl sinnvoller Handlungen unter Einbeziehung von Sensordaten nicht trivial. „Sinnvoll" meint dabei, daß die Aktion dem Erreichen von Zielen dienen soll. Besonders die unvorhersehbare Dynamik der Umwelt, die über Sensoren wahrgenommen wird, welche möglicherweise zu wenig, zu ungenaue oder widersprüchliche Daten liefern, und die Veränderungen der Ziele abhängig von internen sowie externen Zuständen machen es unmöglich, immer die optimale Handlung im Sinne der Ziele zu finden. Zu beachten ist auch, daß diese Entscheidung bei endlicher Speicherkapazität und Rechenleistung in Echtzeit erfolgen muß, also etwa im Rahmen einer „Schrecksekunde", wie man es von einem Menschen erwarten würde.

In den meisten Fällen ist es aber gar nicht notwendig, immer den optimalen Weg zum Ziel zu finden; vielmehr reicht es aus, irgendeinen Weg zum Ziel zu finden, wenn dieser nur einigermaßen adäquat erscheint. Dabei muß vermieden werden, daß der Agent sich für keine der Alternativen entscheiden kann, ständig zwischen Handlungen wechselt oder in Endlosschleifen gerät.

Bei den Zielen differenziert man eine Reihe von Arten: zu erreichende oder zu vermeidende Zustände sowie die Erfüllung von Bedürfnissen, Aufgaben, Motivationen oder Bedingungen, wobei meist mehrere Ziele konkurrieren und mit unterschiedlichen Prioritäten versehen sind. Die Repräsentation dieser Ziele kann implizit oder explizit realisiert sein. Implizit bedeutet in diesem Zusammenhang, daß die Ziele beim Design des Agenten nicht ausdrücklich formuliert werden, sondern

nur in der Vorstellung des Entwicklers existieren, während bei expliziter Repräsentation der Ziele diese konkret bezeichnet werden. Architekturen, die mit impliziten Zielen arbeiten, haben gegenüber expliziten den Nachteil, daß sie fest einprogrammiert und damit statisch sind. Doch lassen sich auch mit Hilfe von expliziten Formen der Repräsentation niemals alle Ziele beschreiben, insbesondere, wenn mit expliziter Formulierung lediglich der Sollzustand von Variablen gemeint ist, wie in den meisten Anwendungen.

In diesem Zusammenhang sei auf die intensiven Forschungen von Barbara Hayes-Roth, Philippe Morignot und anderen vom *Knowledge Systems Labroratory* der Stanford University verwiesen, die sich ausgiebig mit der zielgerichteten Steuerung autonomer Agenten nach menschlichem Vorbild anhand von Motivationsprofilen beschäftigen. Im Vordergrund steht dabei die Modellierung von Persönlichkeit für Akteure in virtuellen Umgebungen. Bei ihrem Ansatz sind Charakterzüge und die Befriedigung verschieden stark ausgeprägter Bedürfnisse das wesentliche Kriterium zur Handlungsauswahl. (vgl. [HAY94], [HAY95a], [HAY95b], [HAY95c], [HAY95d], [HAY96a], [HAY96b], [HAY96c], [HAY96d])

Will man verschiedenartige Agentenimplementationen vergleichen, so müssen die Charakteristika des konkreten Anwendungsfalls (Umgebung, Aufgabe und Art des Agenten) berücksichtigt werden. Zum Beispiel erfordern Umgebungen, in denen die Kosten von Fehlentscheidungen hoch sind, ein höheres Maß an Antizipation als solche, in denen durchaus mehrere Alternativen ausprobiert werden können. Andere Einflußfaktoren, die bei der Betrachtung eine Rolle spielen, sind die Vielfalt und Art der verfügbaren Sensoren sowie die erforderliche Reaktionszeit des Agenten.

Die Auswahl der jeweils nächsten Handlung geschieht bei einem Agenten als Reaktion auf eine eingehende Botschaft. Das Gesamtverhalten eines agentenbasierten Systems ergibt sich also aus der Summe des Verhaltens aller einzelnen Agenten sowie dem Entscheidungs-Netzwerk, das die einzelnen Agenten untereinander jeweils hemmend oder aktivierend verbindet. Diese semantischen Netze werden bei größeren Anwendungen relativ komplex und sind schwer zu überblicken, so daß meist spezielle Werkzeuge zur Herstellung von Agentensystemen eingesetzt werden. Sie unterscheiden sich bezüglich der Unterstützung zum Aufbau der Netzwerke, und zwar unter folgenden Aspekten (vgl. [MAE95b]):

- Werden Intensitäten von Verbindungen unterstützt?
- Können Verbindungen von explizit formulierten Zielen oder global gespeicherten Zuständen abhängen?
- Sind Mechanismen zur (hierarchischen) Strukturierung des Netzwerkes vorhanden?

- Gibt es Werkzeuge, die von einzelnen Verbindungen abstrahieren und Entwicklern die Spezifikation der Zusammenhänge auf höherer Ebene, zum Beispiel mit Hilfe eines Modells zur Reduktion von Zielen in Teilziele, erlauben?

## 5.4 Lernen aus Erfahrung

Im letzten Abschnitt sind Probleme und Lösungsansätze der Handlungsauswahl vorgestellt worden. Im folgenden wird geklärt, wie das über die Sensoren aufgenommene Feedback aus der Umwelt zur stetigen Verbesserung des Verhaltens genutzt werden kann. Solche Mechanismen werden bei nichttrivialen Anwendungen nötig, da es oft zu komplex ist, die Verhaltensweisen aller denkbaren Situationen per Hand und im vorhinein zu programmieren, insbesondere wegen der sich ständig ändernden Umwelt. Das Problem des Lernens aus Erfahrungen ist, bei gegebenen Zielen, Handlungen und Sensordaten eine Optimierung der Handlungsauswahl im Sinne der Ziele zu erreichen, und zwar basierend auf bisher gemachten Erfahrungen, also dem Erfolg oder Mißerfolg früherer Handlungen. Der Lernmechanismus sollte den Eingriff eines „Trainers“ möglich, aber nicht nötig machen, das heißt er sollte im wesentlichen unüberwacht und autonom stattfinden, aber die Eingabe von Erstwissen oder die manuelle Korrektur von Gelerntem erlauben. Erfahrungen sollten nicht nur in expliziten Trainingsphasen aufgezeichnet werden, vielmehr sollten alle Handlungen zur Weiterentwicklung beitragen können. Dabei ist eine Filterung nicht relevanter oder unplausibeler (zum Beispiel gestörter) Sensordaten nötig, um die entstehende Datenflut in den Griff zu bekommen.

Zusätzlich zum eigentlichen Lernmechanismus werden weitere Algorithmen benötigt, etwa zur Bewertung des Erfolges einer Handlung und die Aufteilung dieser Bewertung auf Teilhandlungen, oder zur Auswahl von alternativen Handlungen, um diese im Experiment auszuprobieren. Im allgemeinen bringen die einzelnen Lernverfahren spezielle Verfahren zur Handlungsauswahl mit. Es lassen sich drei Klassen von Architekturen zur Implementation der Handlungsauswahl mit Lerneffekten unterscheiden (vgl. [MAE95b]):

| Name | Überlegung | Handlungsauswahl | Lernmechanismus | Experimentier-strategie |
|---|---|---|---|---|
| **Reinforcement Learning** | Eine Matrix enthält für jedes Situation-Handlung-Paar einen vorraussichtlichen Erfolgswert. | Auswahl derjenigen Handlung mit dem größten zu erwartenden Erfolgssignal. Dabei werden auch die wahrscheinlich nächsten Handlungen einbezogen. | Je nach empfangendem Erfolgssignal wird der entsprechende Wert in der Matrix erhöht oder erniedrigt. | Bei einem bestimmten Prozentanteil wird nicht die optimale, sondern eine zufällige Handlung ausgewählt. |
| **Classifier Systems** | Eine Liste von Regeln (Classifier) mit jeweils einer Anwendungsbedingung, einer Handlung und einem Stärkewert bestimmen die Handlungsauswahl. | Aus denjenigen Classifiers, deren Bedingung zur aktuellen Situation paßt, wird proportional zu deren Stärke einer ausgewählt. | Empfängt der Agent ein Erfolgssignal bzgl. einer Handlung, wird die Stärke derjenigen Regeln, die zu der Ausführung dieser Handlung beigetragen haben, angepaßt. | In bestimmten Zeitintervallen werden die Regeln mit den geringsten Stärkewerten durch Mutationen und Crossovers erfolgreicher Regeln ersetzt.[5] |
| **Model Builders** | Ein kausales Modell aus Bedingungen, Handlungen und ihren wahrscheinlichen Resultaten wird in Form von Schemata aufgebaut. | Durch Abgleich der Situation und der Ziele mit den wahrscheinlichen Resultaten der Schemata wird das geeignetste Schema ausgewählt und die zugehörige Handlung(sfolge) ausgeführt. | Veränderungen der Umwelt werden während der Handlungsausführung aufgezeichnet und in die Liste der Resultate des aktiven Schemas einbezogen. Wenn in einem Schema Inkonsistenzen entstehen, werden neue Schemata mit differenzierteren Bedingungen abgespalten. | Mehrere Ansätze sind möglich. In vorhandenen Implementationen werden Experimente basierend auf Zufall, Zielen oder Aufmerksamkeit durchgeführt. |

[5] Classifier Systems verwenden zum Erstellen alternativer Verhaltensweisen „Genetische Algorithmen“, die nach dem Vorbild der Genetik positive Merkmale neu kombinieren und gleichzeitig Mechanismen zur Abstraktion zur Verfügung stellen. (vgl. [WES96])

Den verschiedenen Verfahren zur Handlungsauswahl und zum Lernen haften jeweils gewisse Vor- und Nachteile an, die vor einem Einsatz unter Berücksichtigung der Anforderungen abgewogen werden sollten. So können beispielsweise Reinforcement Learning und Classifier Systems nur mit konstanten Zielen umgehen. Model Builders können zwar, da sie eine Strukturierung mit Hilfe von Schemata vornehmen, mit variierenden Zielen umgehen, doch ergeben sich wegen des Berechnungsaufwandes für den ständigen Abgleich von Situationen, Zielen und vergangenen Resultaten Einschränkungen für den Echtzeitbetrieb.

# 6 Multiuser-Welten und Avatare

Der Standard zur Beschreibung virtueller Welten ist derzeit VRML2 (Virtual Reality Modeling Language). Verteilte Welten sind wie bei HTML (Hypertext Markup Language) auf der Basis des HTTP-Protokolls (Hypertext Transmission Protocol) und URLs (Uniform Resource Locator) realisierbar. Seit der Version 2 ist es im Rahmen des Standards möglich, Objekte mit Hilfe neuer Mechanismen sowie einer Programmierschnittstelle zu Java und JavaScript mit Verhalten auszustatten. Leider fehlen dem Standard zur Zeit noch wesentliche Vorgaben für den Mehrbenutzerbetrieb und der Repräsentation von Avataren, was zur Folge hat, daß wegen der Nachfrage nach Mehrbenutzerwelten die Hersteller von VRML-Browsern hierfür eigene Konzepte entwickeln und so einen gewissen Wildwuchs entstehen lassen. Dem entgegenzuwirken, hat sich die Initiative Universal Avatars verschrieben. Diese Gruppe erarbeitet in erster Linie Vorschläge zur Standardisierung von Avataren, um diese unabhängig vom verwendeten Browser-Produkt verwenden zu können. Außerdem soll ermöglicht werden, die Eigenschaften des Avatars auch beim Wechseln der Welt über eine URL „mitzunehmen“, indem Formate für solche Eigenschaften vorgegeben werden, und zwar in einer Form, die die bestehende VRML2 Syntax nicht verletzt. Aufbauend auf den VRML2-Standard will Universal Avatars also eine - wie sie es nennen - Universal Avatar Markup Language spezifizieren, die eine formale Beschreibung einer Benutzerrepräsentation erlaubt. (vgl. [MA96])

Folgende Inhalte zur Definition von Avataren werden vorgeschlagen:

- ein Benutzerprofil als URL auf ein HTML-Dokument, welches die reale oder eine erdachte Identität des Benutzers (wie eine Homepage) vorstellt,
- ein Avatar-Modell, bestehend aus einem 3D-Modell in VRML sowie Angaben darüber, wann und wie das Modell zu animieren ist. Bei komplexen Objekten sollen auch vereinfachte Versionen für Betrachter mit langsameren Rechnern zur Verfügung gestellt werden,
- ein Mechanismus zur sicheren Identifizierung der realen Person, die sich hinter dem Avatar verbirgt, zum Beispiel ein PIN oder ein Zertifikat. Dies ist beispielsweise für Online-Shopping in virtuellen Welten unbedingt vonnöten,
- Informationen über die Erreichbarkeit per Chat oder Internet-Telephonie, die bekanntgeben, auf welche Weise man auch mit Partnern jenseits der Grenzen einer virtuellen Welt zu kommunizieren bereit ist,

- anbieterspezifische Daten, etwa in Spielen erworbene Punkte oder Gegenstände respektive bei Shopping-Welten der Inhalt des Warenkorbes.

(vgl. [MA96])

Solche Konzepte werden in Zukunft auch bei der Implementierung von automatisierten Charakteren zu beachten sein, um die Grenze zwischen Avataren und Agenten weitestgehend zu verwischen.

# 7 Programmgesteuerte VR-Charaktere

## 7.1 Überlegungen für eine eigene Realisierung

### 7.1.1 Motivation

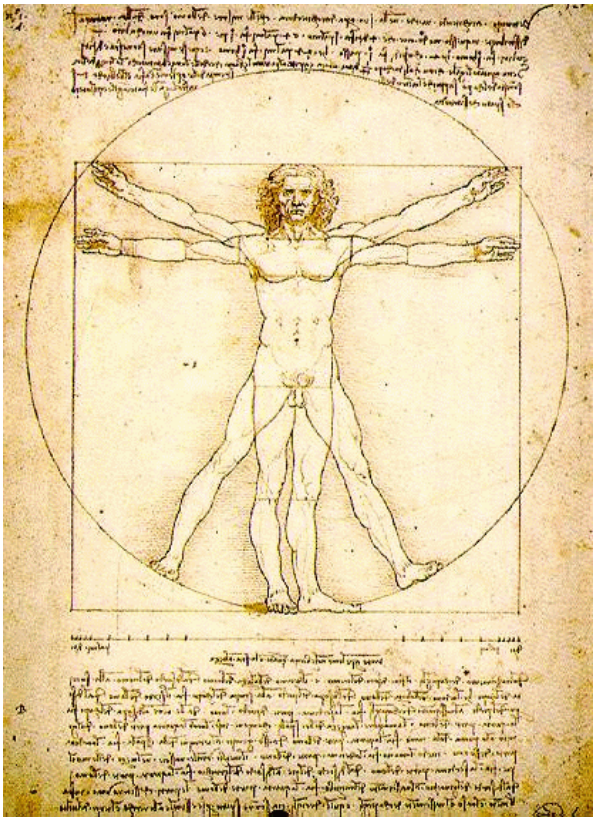


Abbildung 11: Proportionstudie von Leonardo da Vinci

(WebMuseum, Paris. http://www.fhi-berlin.mpg.de/wm/paint/auth/vinci/sketch/ am 20.5.1997)

Virtuelle Welten ohne Bewohner wirken oft wie Geisterstädte, sie vermitteln ein Gefühl der Einsamkeit, und es mangelt ihnen an Interaktivität. Diesen Mangel versuchen Anbieter von VR-Systemen, die einen Betrieb für mehrere Benutzer vorsehen, abzustellen. In solchen Systemen werden die jeweils anderen zugeschalteten Benutzer durch Avatare repräsentiert. Allerdings wollen diese menschlichen Akteure meist die virtuelle Welt konsumieren und nicht etwa eine ihnen zugewiesene Rolle spielen oder gar Dienste anbieten. Wie aber wirkt ein virtuelles Shopping-Center ohne Verkäufer, eine Bibliothek ohne Bibliothekar oder ein Museum ohne Museumsführer? Wie sollen VR-Spiele funktionieren, wenn unbeliebte Rollen nicht besetzt werden oder Charaktere einfach nicht eingeloggt sind?

In den nächsten Abschnitten wird nun als Hauptgegenstand meiner Untersuchungen ein Verfahren entwickelt, wie bestimmte Rollen durch automatisierte - also programmgesteuerte - Charaktere, die wie Avatare figürlich dargestellt werden, übernommen werden können.

Meine Umsetzung wird ausgewählte Erkenntnisse und Modelle aus der Verhaltenssteuerung beim Menschen und der Animation von 3D-Objekten mit Agentenkonzepten neu kombinieren und diese für automatisierte Charaktere in virtuellen Welten einsetzen. Dabei soll nicht die Nachbildung menschlicher Intelligenz im Vordergrund stehen, sondern vielmehr die Modellierung von vielfältigem und vor allem glaubhaftem Verhalten. Auch soll zunächst kein Wert auf perfekten Realismus bei der Darstellung gelegt werden, was einen Echtzeitbetrieb bei durchschnittlicher Hardwareausstattung verhindern würde. Nichtsdestotrotz soll das Konzept Schnittstellen und Möglichkeiten zur Erweiterung enthalten und so eine Plattform für Weiterentwicklungen und die Einbindung spezialisierter Module darstellen. Außerdem will ich nicht - wie andere, die sich theoretisch mit Agenten beschäftigen - ein experimentelles VR-System speziell für meine Anforderungen selber entwickeln, sondern bestehende Standards berücksichtigen. Dies erhöht die Möglichkeiten für den praktischen Einsatz meiner Software, setzt mich auf der anderen Seite jedoch den Mängeln der verwendeten Standards und Produkte aus. Meine Implementation wird auf VRML2 (Beschreibungssprache für virtuelle Welten) aufsetzen und Steuerungsmodule in Java für automatisierte Charaktere enthalten, die sowohl eine Java-Programmierschnittstelle als auch eine spezielle Skriptsprache zur Verhaltensdefinition zur Verfügung stellen.


Abbildung 12: Automatisierter Charakter

### 7.1.2 Anforderungen

Das Verfahren zur Erstellung automatisierter Charaktere sollte verschiedenen Anforderungen genügen.

Diese sind im einzelnen:

- Leichte Einbindbarkeit in bestehende Welten soll gegeben sein;
- Komplexe Bewegungen mit parallel ablaufenden Teilbewegungen sollen definierbar sein;
- Aus unterschiedlichen Situationen soll potentiell unterschiedliches Verhalten resultieren;
- Textdialoge mit möglichst vielfältigem Dialogverhalten sollen unterstützt werden;
- Mechanismen zur Parametrisierung und Konfiguration, mit denen Charaktere ohne Programmieraufwand in möglichst vielfältiger Form verwendet werden können, sollen bereitgestellt werden;
- Die Wiederverwendbarkeit der Software-Module soll durch objektorientierte Programmierung sichergestellt werden;
- Spezialisierte Module - zum Beispiel zur Animation von Haut, Kleidung und Haaren - sollen einbindbar sein.

### 7.1.3 Auswahl der Software-Konfiguration

In den letzten Jahren hat sich VRML als der Standard zur Beschreibung von interaktiven 3D-Objekten und virtuellen Welten entwickelt. In VRML definierte virtuelle Welten können im WWW (World Wide Web) verteilt vorliegen und mit sogenannten VRML-Browsern erforscht werden. Diese Browser sind Programme, die die VRML-Welten darstellen und dem Benutzer erlauben, durch diese Welten zu navigieren. Ähnlich wie beim „Surfen“ durch das WWW mit Hilfe von HTML-Browsern, können auch in VRML-Welten Rechner- und Ländergrenzen über Hyperlinks mühelos überwunden werden.

Die erste Version der VRML1-Spezifikation wurde von Silicon Graphics auf der Grundlage des Open Inventor Dateiformates entwickelt, bei der VRML2-Spezifikation waren zusätzlich insbesondere die Firmen Sony Research und Mitra beteiligt. Bei der Standardisierung wurden auch Vorschläge aus der Email-Diskussionsgruppe www-vrml@wired.com berücksichtigt.

In der VRML2-Spezifikation sind verschiedene Verfahren zur Steuerung bewegter Objekte enthalten. Der einfachste Mechanismus sind die sogenannten Interpolatoren, mit denen Rotationen und Translationen von Objekten und Teilobjekten durchgeführt werden können, jedoch bleiben die Bewegungsabläufe immer dieselben. Wesentlich mehr Möglichkeiten bietet die Steuerung über JavaScript, eine Untermenge von Java. Die in dieser Skriptsprache notierten Programmteile werden direkt in die VRML-Dateien geschrieben und vom Browser interpretiert. Die dritte und umfangreichste Alternative bietet die Schnittstelle zu externen Java-Programmen. Solche Programme können alle Vorzüge der Programmiersprache Java nutzen und neben der bloßen Steuerung der 3D-Objekte zum Beispiel auch Dateioperationen, fensterbasierte Benutzungsschnittstellen und Netzwerk-Zugriffe durchführen. Über die Java-VRML-Schnittstelle dürfen verschiedene Informationen ausgetauscht werden: Parameter können mit Hilfe von *Fields* (Startparameter) von der virtuellen Welt an Javaprogramme übergeben werden, oder es können Kanäle für den Ereignisaustausch - *EventIns* und *EventOuts* - eingerichtet werden.

| VRML2-Welt → Java-Programm | Java-Programm → VRML2-Welt |
|---|---|
| Startparameter (*Fields*) | |
| Ereignisse (*EventIns*), Reaktion auf z.B. Mausklicks auf ein Objekt, Zeittakte, Veränderung der räumlichen Nähe, der Sichtbarkeit oder einzelner Vektoren | Steuerung (*EventOuts*) beliebiger Vektoren oder Schalter zu jeder Zeit des Programmablaufes |


Abbildung 13: Der VRML2-Browser von Sony „Community Place“

Um nicht den Beschränkungen der Skriptsprache JavaScript zu unterliegen, um Konfigurationsdaten abspeichern und um eine fensterorientierte Benutzerführung realisieren zu können, habe ich mich entschlossen, zur Realisierung der automatisierten VR-Charaktere die Java-Schnittstelle zu verwenden.

Wegen der Aktualität von VRML2 gibt es derzeit leider keinen Browser, der die Spezifikation vollständig erfüllt, so daß ich während der Entwicklung meiner Software ständig auf der Suche nach den aktuellsten Produkten und Versionen war. Zum Testen der aktuellen Programme und VRML2-Welten habe ich den Browser *Community Place* für Windows95/NT der Firma Sony verwendet, da dieser mit der Implementation der Java-Schnittstelle am weitesten fortgeschritten ist.

### 7.1.4 Ebenen der Verhaltenssteuerung

Das Verfahren zur Verhaltensdefinition von automatisierten Charakteren soll verschiedene Steuerungsmöglichkeiten erlauben. Wie schon in den vorangegangenen Kapiteln über Modelle menschlicher Verhaltenssteuerung, über die Animation von 3D-Objekten und über autonome Agenten erläutert wurde, können hierbei mehrere Abstraktionsebenen und Denkmodelle unterschieden werden. Ziel meiner Implementation ist insbesondere, Schnittstellen auf einer möglichst großen Zahl dieser Ebenen zu schaffen, um so flexible und vielschichtige Handlungen definierbar zu machen und gleichzeitig eine Plattform für weitergehende Modelle zur Verfügung zu stellen. Ebenen der Verhaltenssteuerung könnten meines Erachtens sein:

1. Spezifikation einzelner Vektoren,
2. Interpolation von Schlüsselvektoren zu flüssigen Bewegungen,
3. Synchronisation von sequentiellen und parallelen Einzelbewegungen,

4. Reaktion auf Umwelteinflüsse,

5. Handlungsauswahl unter Einbeziehung von Zielen und Sensorinformationen,

6. Festlegung der Handlungsauswahl auf der Grundlage von Charaktereigenschaften,

7. Adaption von Handlungen im Sinne von zielbezogener Optimierung.

Diese Ebenen zeichnen sich durch einen ansteigenden Abstraktionsgrad aus und sollen gleichzeitig eine Prioritätenliste für meine Realisierung als Javaklassen sein. Während in Ebene 1, die von der VRML-Java-Schnittstelle implementiert wird, ausschließlich das Setzen konkreter Werte unterstützt ist, werden von mir in den folgenden Ebenen - jeweils aufbauend auf den vorigen - abstraktere Ansteuerungen und Konzepte bereitgestellt. Mechanismen der Ebene 6 könnten zum Beispiel erlauben, Charaktere durch eine Anzahl von markanten Parametern, etwa Kenngrößen für Freundlichkeit, Agilität, Aggression oder ähnliche, zu spezifizieren. Eine solche Verhaltenssteuerung durch Vorgabe einer Persönlichkeit mit Hilfe von Motivationsprofilen haben Barbara Hayes-Roth, Philippe Morignot und andere vom Knowledge Systems Labroratory der Stanford University vorgeschlagen (vgl. [HAY94], [HAY95a], [HAY95b], [HAY95c], [HAY95d], [HAY96a], [HAY96b], [HAY96c], [HAY96d]).

Einige grundlegende Funktionen, die in dem beschriebenen Schichtenmodell benannt wurden, habe ich als Javaklassen implementiert, die als Programmierschnittstelle für Weiterentwicklungen dienen können. Die einzelnen Module und ihr Zusammenspiel werden im folgenden kurz behandelt.

## 7.2 Überblick über die Softwaremodule

Um meine Software mit angemessenem Aufwand fertigstellen zu können, war es unbedingt notwendig, eine Auswahl aus den vielen Dingen, die man realisieren könnte, zu treffen. So sind in den Kapiteln über Animation und Agenten einige Lösungen für Teilprobleme vorgestellt worden, beispielsweise wie realistisch animiert werden kann und wie bei Agenten reaktives und adaptives Verhalten erzeugt wird. Im Rahmen meiner Implementation ist es nicht das Anliegen, solche speziellen Lösungen zu perfektionieren. Vielmehr möchte ich eine allgemeine Basis schaffen, die einerseits die bequeme Definition von reaktivem Verhalten und andererseits die Integration der verschiedensten spezialisierten Konzepte auf einfache Weise gestattet. Aufsetzend auf dem populärsten Standard zur Beschreibung virtueller Welten, nämlich VRML2, sollen meine Java-Klassen die notwendigen Primitiven und Werkzeuge zur Verhaltenssteuerung automatisierter Charaktere in virtuellen Welten

zur Verfügung stellen und durch eine dynamische Schnittstellenverwaltung die Erweiterbarkeit und Anpaßbarkeit sicherstellen.

Die wichtigste Funktionalität meiner Implementation besteht in der Realisierung einer speziellen Skriptsprache zur Definition von reaktivem Verhalten (☞ 7.3 Eine Skriptsprache für Handlungspläne) mit direktem Zugriff auf konfigurierbare Schnittstellen zur virtuellen Welt. Außerdem stelle ich durch eine objektorientierte Programmierschnittstelle eine Basisfunktionalität zur Erweiterung der Skriptsprache oder für andere Weiterentwicklungen zur Verfügung. Insgesamt besteht mein Java-Programm aus 20 Klassen, zusammengefaßt in drei Modulen (VRMLInterface, VRRobot, SchemeRobot), die in diesem Abschnitt im Groben erläutert werden, da sie die Programmierschnittstelle repräsentieren.

Die dreidimensionale Figur, die ich zum Testen meiner Javaklassen entworfen habe, besteht aus einfachen geometrischen Formen. Sie ist als sogenannter PROTO in VRML2 realisiert, was soviel bedeutet, daß es sich um einen Prototypen handelt, der beliebig oft instanziiert und durch die Übergabe bestimmter Parameter geeignet variiert werden kann. Mit Hilfe dieses Mechanismus wird die gleiche Figur - jeweils mit einer unterschiedlichen Verhaltenskonfiguration - einfach an verschiedenen Orten der virtuellen Welt einsetzbar.

In VRML2 existiert ein spezieller Knotentyp namens Script zur Einbindung von Javaprogrammen in virtuelle Welten. Diesem Knoten, der irgendeinem VRML-Objekt untergeordnet sein kann, wird als Eigenschaft eine URL auf eine Javaklasse übergeben. Diese Klasse muß als Elternklasse Script haben. Sie ist Teil der - in der VRML2-Spezifikation vorgegebenen - VRML-Java-Schnittstelle, die insbesondere den lesenden oder schreibenden Zugriff auf Schnittstellendaten erlaubt. Die Schnittstelle besteht aus beliebigen VRML2-Datentypen - zum Beispiel aus Vektoren, Strings oder boolschen Werten, die als Fields, EventIns oder EventOuts ausgetauscht werden. Die Bereitstellung der Javaklasse Script ist Aufgabe der Hersteller von VRML2-Browsern. Ein Script-Knoten ist als Steuerungseinheit Teil meines VRML-Prototypen, der die dreidimensionale Repräsentation meines automatisierten Charakters enthält. Auf der Javaseite habe ich die Klassen VRMLInterface, VRRobot, und SchemeRobot als Unterklassen von Script hergestellt und so nach und nach mehr Funktionalität hinzugefügt.

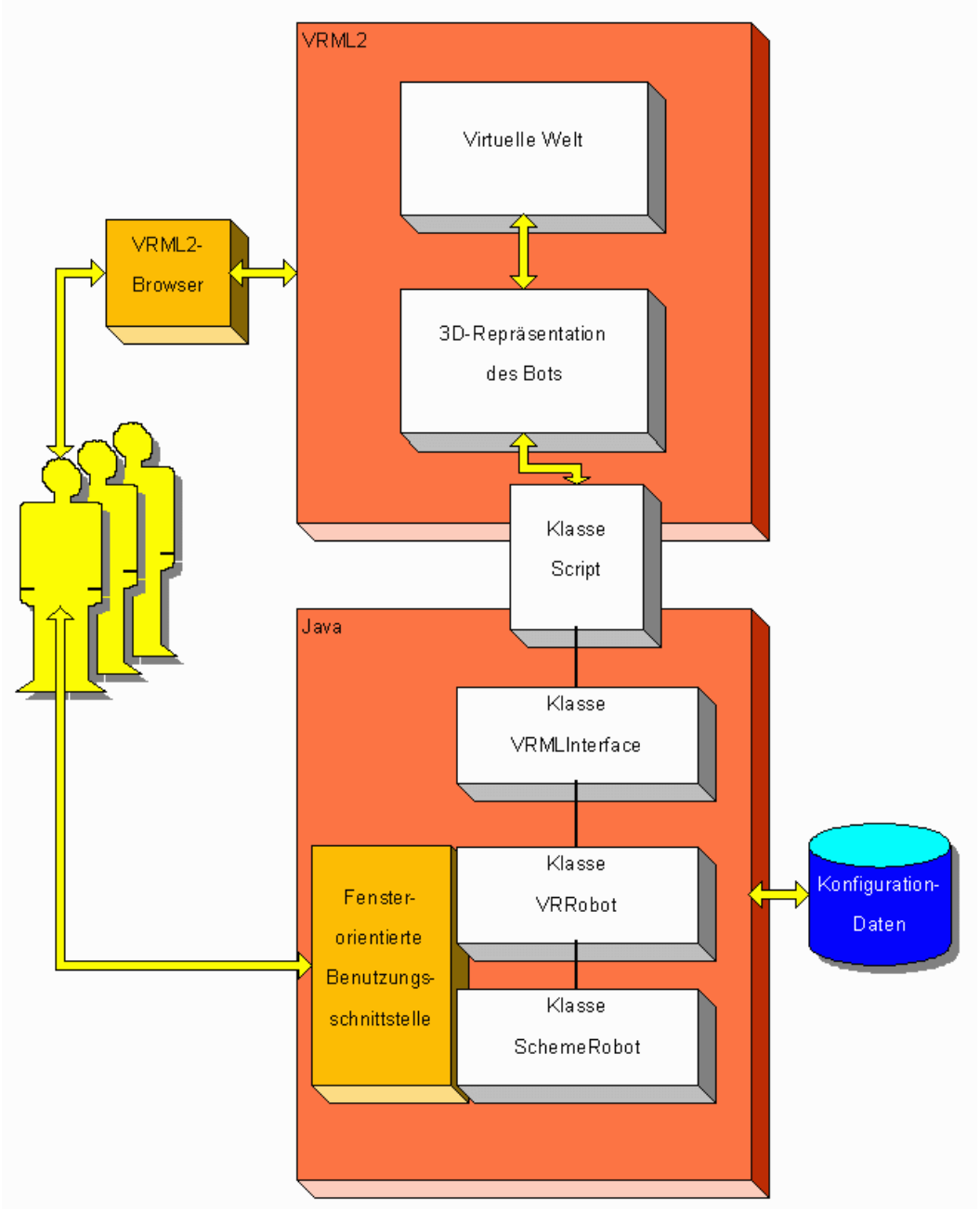



Abbildung 14: Überblick über Softwaremodule

Während Script nur die nackte Schnittstelle zur virtuellen Welt realisiert, ist die Klasse VRMLInterface für eine weitergehende Verwaltung der Schnittstelle zuständig. Die zu verwaltenden *EventIns* und *EventOuts* werden dabei als Startparameter übergeben und sind danach mit Hilfe von Zugriffsfunktionen über ihren Namen ansprechbar. Die Schnittstelle zur virtuellen Welt ist somit nicht im Programm festgelegt, sondern läßt sich bei Bedarf verändern oder erweitern. Die von VRMLInterface bereitgestellte, dynamische Schnittstellenverwaltung umfaßt auch die Implementation von Funktionen zur Animation linearer Translations- und Rotationsbewegungen und deren Synchronisation anhand eines gemeinsamen Zeittaktes. Sowohl sequentielle als auch parallele Einzelbewegun-

gen werden unterstützt. Außerdem besteht die Möglichkeit, Bewegungen zyklisch - also in ständiger Wiederholung - bis zur expliziten Beendigung ablaufen zu lassen.

Die Klasse VRRobot baut auf der Funktionalität von VRMLInterface auf und ergänzt einen Interpreter für die von mir entworfene Skriptsprache zur Definition von Handlungsplänen für automatisierte Charaktere in virtuellen Welten. Diese Skriptsprache (☞ 7.3 Eine Skriptsprache für Handlungspläne) bietet nicht nur den Zugriff auf die Funktionalität von VRMLInterface, sondern erlaubt auch die Konfiguration von Dialogverhalten, die Strukturierung mit Hilfe von Prozeduren und Callbacks, Variablen und Kontrollstrukturen. Eine Fensterbedienung erlaubt das Durchführen von Textdialogen sowie das Editieren, Laden und Speichern der Verhaltensdefinitionen.

Die gesamte Funktionalität der Skriptsprache habe ich zusätzlich als Java-Programmier-schnittstelle zur Verfügung gestellt, um Weiterentwicklungen zu erleichtern. Als exemplarische Demonstration der Programmierschnittstelle habe ich in der Klasse SchemeRobot einen schemaorientierten Ansatz implementiert, der mit einer angepassten Fensteroberfläche eine Strukturierung in Verhaltensschemata realisiert und die Skriptsprache um spezifische Prozeduren erweitert (☞ 7.5.3 Beispielhafte Anwendung der Programmierschnittstelle: Der SchemeRobot).

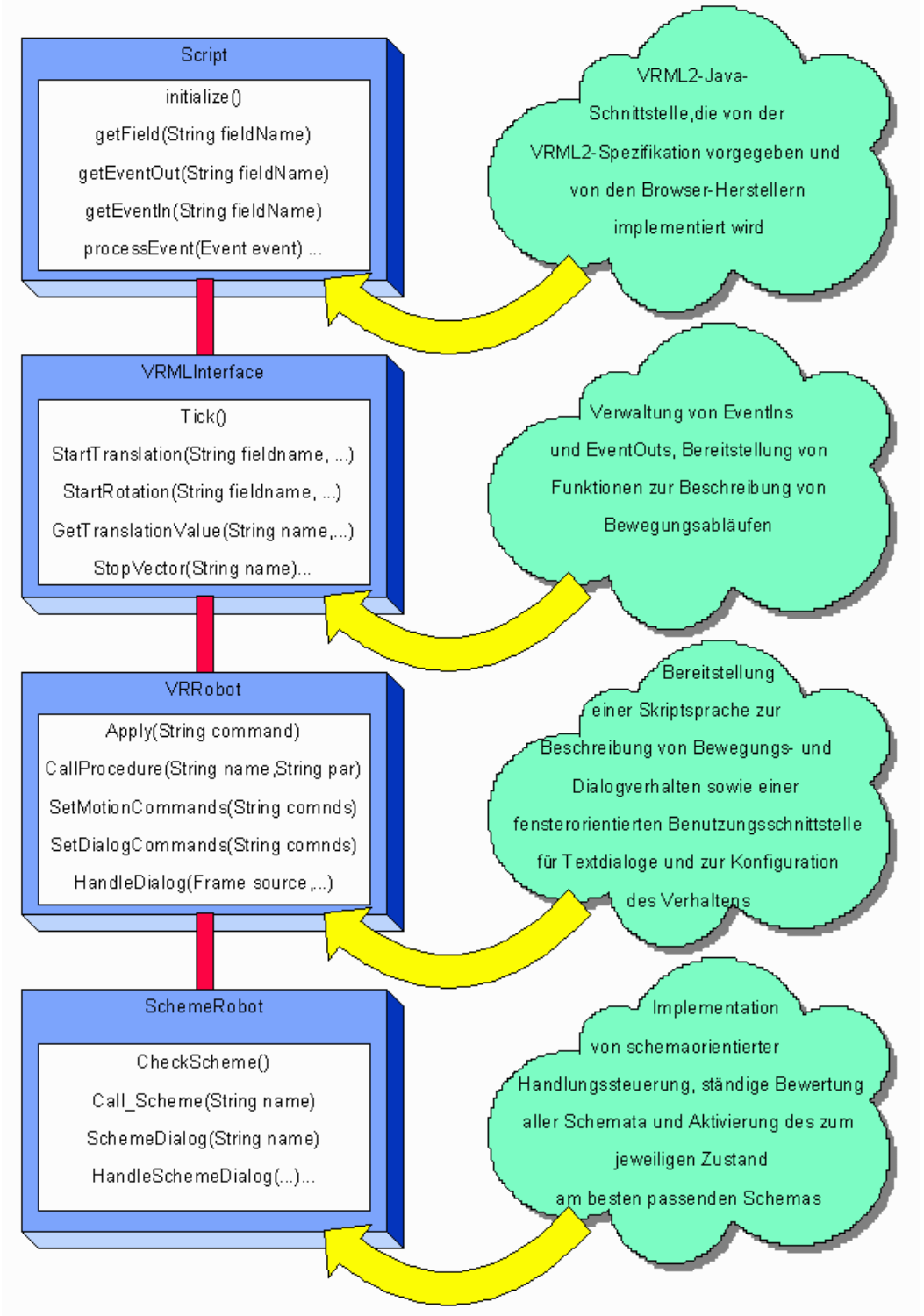



Abbildung 15: Überblick über Java-Komponenten

## 7.3 Eine Skriptsprache für Handlungspläne

Eine wichtige Funktionalität des von mir entwickelten Java-Programmes ist die Bereitstellung einer speziellen Skriptsprache zur Handlungsdefinition. Dies bedeutet im einzelnen, daß jedem der Agenten eine oder mehrere Textdateien zugeordnet sein können. Der in diesen Dateien enthaltene Text wird vom Java-Programm zur Laufzeit als Handlungsplan interpretiert und in konkrete Aktionen übersetzt, wobei der Text den Regeln der Skriptsprache gehorchen muß. Die genaue Definition dieser Sprache ist im Anhang dieser Arbeit zu finden, einige Eigenschaften und Beispiele werden jedoch im folgenden herausgestellt.

### 7.3.1 Vordefinierte Prozeduren

Eine Reihe von Primitiven stehen dem Benutzer der Skriptsprache in Form von parametrisierbaren Prozeduren zur Verfügung, die sich auf die Ansteuerung von Schnittstellen sowie die Ausgabe von Text in dem dafür vorgesehenen Dialogfenster beziehen. Die Parameter solcher Prozeduren werden generell als durch eine mit Kommata getrennte Liste übergeben, wobei jeder Parameter aus einem Parameternamen/Parameterwert-Paar besteht. Zum direkten Setzen von *EventOuts* der Typen Rotations-/Translationsvektoren, Strings und Flags ist die Prozedur *EventOut* vorgesehen, mit deren Hilfe zum Beispiel Textnachrichten versendet, Schalter verändert oder Körperteile ruckartig bewegt werden können.

Beispiel (Senden eines Strings an ein Vorlesemodul über die Schnittstelle „ReadEngineIn“):

```
EventOut vector ReadEngineIn, value 'Test der Vorlesefunktion'
```

Für lineare Animationen können *Translate* und *Rotate* benutzt werden. Dabei werden jeweils der Zielzustand und die Dauer der Bewegung als Parameter übergeben. Die Dauer wird in Einheiten eines globalen Zeittaktes, der in der virtuellen Welt mittels eines sogenannten TimeSensors vorgegeben wird, definiert. Sowohl *Translate* als auch *Rotate* können den Kontrollfluß optional entweder sofort zurückgeben oder bis zum Ende der Bewegung behalten, bei Rotationen kann außerdem eine endlose Wiederholung angewiesen werden, etwa um die ständigen Arm- und Beinbewegungen im Hintergrund ablaufen zu lassen. Solche parallel gesteuerten Vektoren können mit Hilfe der Prozedur *Stop* selektiv zum Stillstand gebracht werden. Für eine zeitliche Synchronisation sequentieller und paralleler Aktionen ist die Prozedur *Wait* gedacht, die für eine bestimmte Anzahl von Zeittakten den Kontrollfluß für sich beansprucht.

Beispiel (Achselzucken):

```
Rotate vector arm1rot , axisx 0, axisy 0, axisz 1 , direction 50 50 0, slides 5, wait false
Rotate vector arm2rot , axisx 0, axisy 0, axisz 1 , direction -50 -50 0, slides 5, wait false
Wait slides 15
```

Sollen Textnachrichten an den Benutzer der virtuellen Welt gerichtet werden, kann die Prozedur *Say* zum Einsatz kommen. Der als Parameter übergebene Text wird in der oberen Hälfte eines geteilten Fensters ausgegeben, während frühere Ausgaben nach oben gescrollt werden. Gleichzeitig steht der untere Teil für Texteingaben des Benutzers zur Verfügung (☞ 7.4 Dialogverhalten).

Beispiel (Textausgabe):

```
Say Folge mir, ich führe dich hier herum!
```

### 7.3.2 Vordefinierte Funktionen und Operatoren

Um komplexe Animationen durchzuführen, sind Berechnungen notwendig. Diese Anforderung erfüllt die Skriptsprache mit Hilfe vordefinierter Operatoren und Funktionen auf rationalen oder boolschen Werten, die unter Beachtung der Syntax und der Typen beliebig verschachtelt werden können. Die Reihenfolge der rekursiven Abarbeitung kann durch Einsatz von Klammern geeignet beeinflußt werden. Neben den Grundrechenarten (+, –, *, /) und logischen Operatoren (&, |, !, =, >, <, >=, <=) sind insbesondere die trigonometrische Funktionen (*sin, cos, tan, asin, acos, atan*) sowie die Wurzelfunktion (*sqrt*) implementiert.

Für die Suche in Strings nach Teil-Strings existiert der Operator *Contains*. Mit seiner Hilfe kann zum Beispiel in den bisherigen Benutzereingaben (Funktion *history*) nach Schlüsselwörtern gesucht werden.

Normalerweise wird ein Ausdruck erst berechnet, wenn er wirklich gebraucht wird, also wenn eine der Primitiven seinen Wert benötigt. Manchmal macht es aber auch Sinn, diese Berechnung explizit herbeizuführen, beispielsweise um doppelte Auswertungen zu verhindern oder die Prozedur *Say* anzuweisen, nicht einen bestimmten Ausdruck, sondern seinen Wert auszugeben. Eine solche erzwungene Auswertung kann mit der Funktion *Evaluate* erzielt werden.

Beispiele (Ausgeben der Ergebnisse von Ausdrücken):

```
Say Evaluate (180+(atan (20.3/12.6))) <= 90
Say Evaluate (history(5) contains Essen) | (history(5) contains Hunger)
```

### 7.3.3 Variablen

Um flüchtige Zustände zu speichern und um Berechnungen in Teilschritte zerlegen zu können, braucht man globale und lokale Variablen. Die Wertzuweisungen werden ohne vorherige Deklaration der Variablen mit *GlobalVar* respektive *LocalVal* durchgeführt, wobei es belanglos ist, ob diese zuvor schon einmal verwendet wurden. Lokale Variablen stehen im Gegensatz zu den immer gültigen, globalen Variablen nur in der gerade aktiven selbstdefinierten Prozedur (☞ 7.3.5 Selbstdefinierte Prozeduren) zur Verfügung. Da den Variablen keine Typen zugeordnet sind, geschieht die Wertzuweisung zunächst nur auf der Basis von Zeichenketten, falls die Auswertung nicht explizit mittels *Evaluate* herbeigeführt wird. Der einer Variablen zugewiesene Wert kann durch ihren Namen mit dem %-Zeichen als Präfix referenziert werden.

Beispiel (Wertzuweisung und Referenzierung der globalen Variable „food"):

```
GlobalVar food Evaluate %food-1
```

Da die Schnittstellendaten - also die definierten EventIns und EventOuts - ständig als globale Variablen auslesbar sind, ist ein bequemer Zugriff auf den Zustand der virtuellen Welt möglich.

### 7.3.4 Kontrollstrukturen

Kontrollstrukturen sind wichtige Mechanismen, um den Ablauf eines Programmes zu manipulieren. Essentiell sind dabei die bedingte Befehlsausführung und das Vor- und Zurückspringen innerhalb eines Programmes.

Bedingte Aktionen können in der Skriptsprache mit Hilfe einer „*If....then...else...*"-Anweisung definiert werden. Die Durchführung von Handlungen kann somit abhängig von Bedingungen gemacht werden.

Beispiel (if-Anweisung):

```
if %food < 5 then Say ich habe Hunger else Say Noch bin ich satt
```

Für Sprünge innerhalb des Programmes wird der Befehl „*Goto*" bereitgestellt, mit dem definierte Programmstellen, die mit „*Tag*" gekennzeichnet werden, angesprungen werden können. Die dazwischenliegenden Zeilen werden nicht ausgeführt. Mit Hilfe dieses Mechanismus können, falls die Einsprungstelle hinter dem Sprungbefehl steht, Aktionen unterdrückt werden; wenn sich die Ein-

sprungstelle aber vor dem Sprungbefehl befindet, kommt es zu einer wiederholten Ausführung der eingeschlossenen Zeilen. Auf diese Weise können Schleifen realisiert werden.

Beispiel (Schleife):

```
Tag WartenaufEssen
Say ich habe Hunger
Wait slides 10
if %food < 5 then Goto WartenaufEssen
```

### 7.3.5 Selbstdefinierte Prozeduren

Sollen komplexe Programme beziehungsweise Handlungspläne erzeugt werden, ist die Hierarchiebildung ein unerläßliches Hilfsmittel, um die vielfältigen Einzelaktionen zu strukturieren und damit handhabbarer zu machen. Um dies zu ermöglichen, sieht die Skriptsprache das Erstellen selbstdefinierter Prozeduren vor. Solche Prozeduren können sowohl vordefinierte als auch selbstdefinierte Prozeduraufrufe zu neuen, komplexeren Einheiten zusammenstellen. Bei der Definition eigener Prozeduren kann eine Parameterliste angegeben werden, wobei die einzelnen Parameter niemals zwingend, sondern immer optional sind und deswegen jeweils ein Default-Wert angegeben werden muß.

Beispiel (die selbstdefinierte Prozedur namens „Nod“ realisiert Kopfnicken):

```
procedure Nod wait true
begin
  Rotate vector headrot, axisx 1, axisy 0, axisz 0 , direction 20 -20 20 -20 0, slides 5
  if %wait then Wait slides 25
end
```

Die Prozedur *VRRobotMain* ist die Hauptprozedur, die nach dem Laden einer Programmdatei gestartet wird.

Eine spezielle Art selbstdefinierter Prozeduren sind die Callbacks, die die Reaktion auf Ereignisse aus der Umwelt beschreiben sollen. Callbacks werden durch gleichnamige *EventIns* der Java-VRML2-Schnittstelle ausgelöst. Falls für einen eintretenden *EventIn* ein Callback, das heißt eine gleichnamige Prozedur, existiert, wird diese ausgeführt. Die vorher gerade aktive Prozedur wird derweil auf einem Stack abgelegt und nach Abarbeitung des Callback wieder aufgenommen.

Prozeduraufrufe können vor ihrer Ausführung wie Strings behandelt werden, also auch Variablen zugewiesen oder als Parameter an andere Prozeduren übergeben werden.

Normalerweise werden die Prozedurdefinitionen sofort beim Laden der Skriptdateien eingelesen und werden danach in einer Liste verwaltet. In manchen Fällen ist es jedoch notwendig, zur Laufzeit neue Prozeduren zusammenzustellen oder alte Prozeduren zu verändern. Indem die Skriptsprache hierfür ein Verfahren implementiert, wird der Grundstein für adaptives Verhalten gelegt.

Beispiel (Definieren und Aufruf der Prozedur „newproc“):

```
LocalVar commands [Say Das Herstellen neuer Prozeduren...\nSay ... zur Laufzeit]
LocalVar commands [%commands\nSay ...FUNKTIONIERT!]
LocalVar defproc [procedure newproc\nbegin\n %commands\nend]
%defproc
newproc
```

### 7.3.6 Preprozessor

Beim Einlesen der Konfigurationsdatei findet zunächst eine Vorverarbeitung des Skriptprogrammes durch einen Preprozessor statt. So werden zum Beispiel Kommentare entfernt, die wie in den Programmiersprachen C oder Java mit /*...*/ oder //... notiert werden. Desweiteren erlaubt der Preprozessor eine Modularisierung in mehrere Dateien, die mit einer *Import*-Anweisung eingebunden werden können, so daß Prozeduren zu Modulen zusammenfaßbar sind.

### 7.3.7 Erweiterbare Schnittstelle zur virtuellen Welt

Um Erweiterbarkeit auch ohne Änderungen in den Javaklassen zu gewährleisten, werden die Schnittstellen zwischen der Skriptsprache, Java und VRML dynamisch verwaltet. Das bedeutet, daß die Javaklassen selbst kein Wissen über den Aufbau der anzusteuernden Charaktere oder die virtuelle Welt besitzen, statt dessen verwalten sie nur Ein- und Ausgangsdaten, deren Namen sie in der Skriptsprache als Referenz auf die Schnittstellendaten durchreichen. Die Spezifikation der Schnittstelle besteht aus Listen von *EventIns* und *EventOuts* und wird beim Laden der Javaklassen übergeben. Diese *EventIns* und *EventOuts* können Rotations- oder Translationsvektoren, boolsche Werte oder Zeichenketten sein, die in der VRML2-Welt beliebig mit Objekten verknüpfbar sind. Solche Objekte sind zum Beispiel die einzelnen Körperteile oder Sensoren des anzusteuernden Charakters; es kommen aber auch andere Objekte sowie weitere Javaklassen wie selbstdefinierte Sensoren oder spezielle Module etwa zur Sprachein- und -ausgabe o.ä. in Frage. Durch diesen Mechanismus zum Austausch von Botschaften mit anderen Javaklassen wird die Aufteilung der Gesamt-Funktionalität in parallel ablaufende, spezialisierte Experten nach dem Vorbild von Agentenarchitekturen möglich.

Auf die in der Schnittstellenbeschreibung spezifizierten *EventOuts* kann in den Prozeduren *EventOut*, *Translate* und *Rotate* über deren Namen Bezug genommen werden, *EventIns* wirken sich durch den Aufruf gleichnamiger Callbacks aus. Außerdem stehen sowohl *EventIns* als auch *EventOuts* für den Lesezugriff als globale Variablen zur Verfügung.

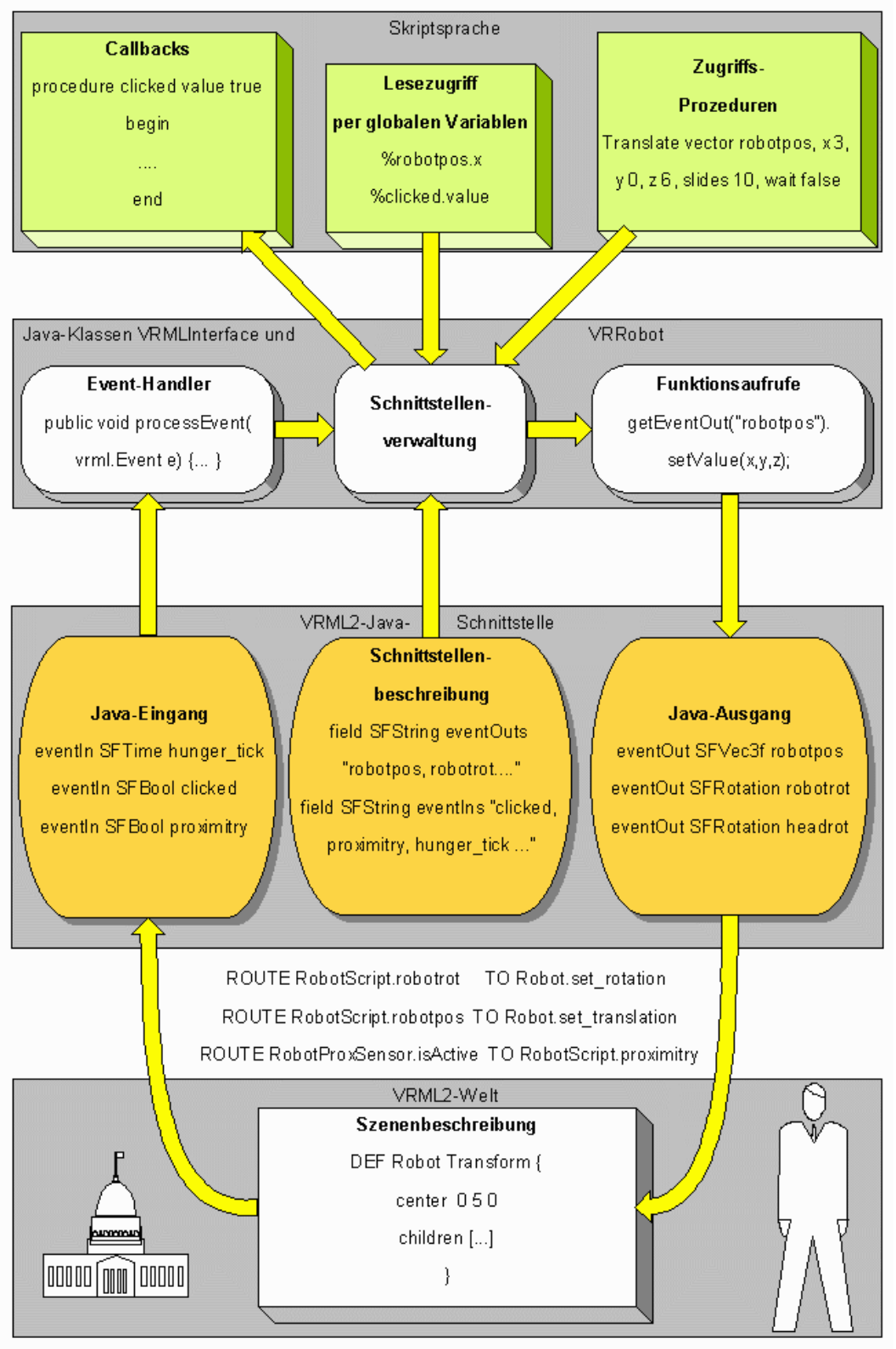



Abbildung 16: Erweiterbare Schnittstelle der Skriptsprache zur virtuellen Welt

## 7.4 Dialogverhalten

Bislang ist im VRML2-Standard kein Verfahren zur textlichen Interaktion vorgesehen Das liegt daran, daß der Mehrbenutzerbetrieb insgesamt noch nicht enthalten ist. Zwar bieten einige Browser eigene Lösungen an, die jedoch untereinander nicht kompatibel sind. Um die Möglichkeit zur sprachlichen Verständigung mit meinen automatisierten Charakteren nicht von Firmenstandards abhängig zu machen, habe ich für diesen Zweck einen eigenen Mechanismus vorgesehen, der mit jedem Java-fähigen Browser funktioniert. Natürlich ist es für die Zukunft wünschenswert, sowohl mit Avataren als auch mit Agenten auf dieselbe Art und Weise zu kommunizieren. Meine Lösung, die also als eine temporäre betrachtet werden sollte, sieht ein geteiltes Fenster vor, in dessen unteren Teil der Benutzer Textnachrichten an den Agenten eingeben kann. Im oberen Teil werden die Aussagen beider Gesprächspartner scrollbar ausgegeben, so daß sich der bisherige Verlauf der Unterhaltung gut zurückverfolgen läßt.

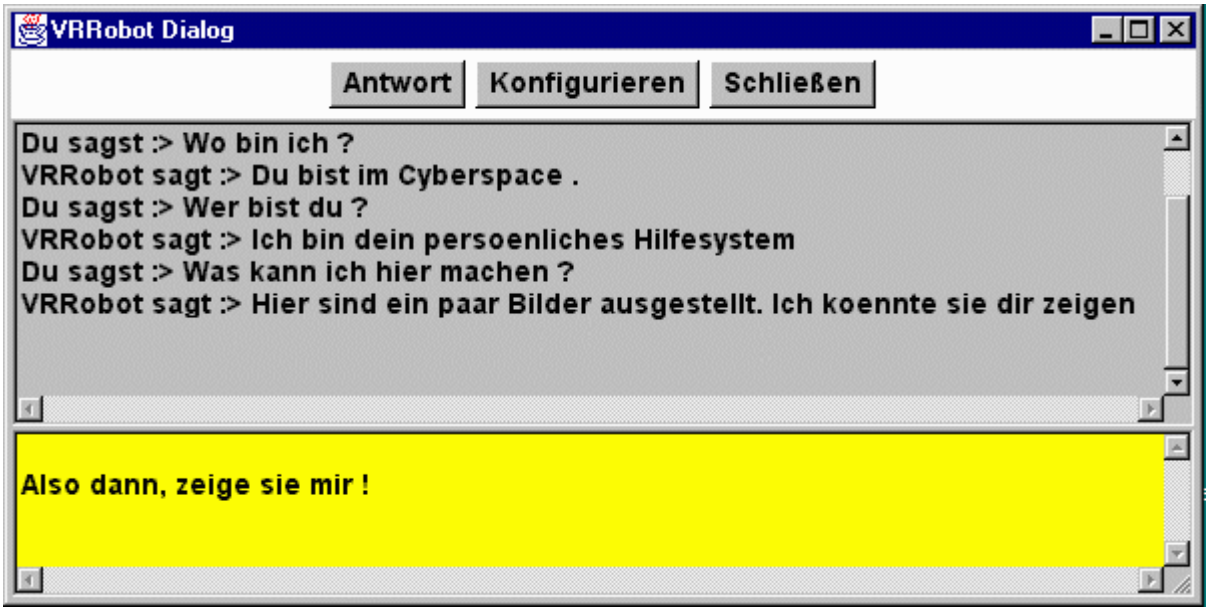



Abbildung 17: Textdialog

Die Fähigkeit zur textlichen Interaktion beschränkt sich auf einen einfachen Mechanismus, der eher mit Weizenbaums ELIZA als mit einem sprachverstehenden KI-System zu vergleichen ist. Die Grundlage des Dialogverhaltens bildet eine Liste von Tripeln, die aus der Konfigurationsdatei gelesen wird. Mit Hilfe der Funktion des Preprozessors zum Importieren von Dateien kann auch hier eine Modularisierung vorgenommen werden. Die einzelnen Tripel können aus einem Suchmuster, einer Antwort und einem Befehl der Skriptsprache bestehen.

Beispiele:

| Suchmuster | Antwort | Befehl |
|---|---|---|
| "Kannst Du %1" | "Natuerlich. Und Du? Kannst Du %1 ?" | Nod |
| "Erinnerst Du Dich an %1" | "Wie könnte ich %1 vergessen ?" | Shrug |
| "%1 vielleicht %2" | "Du scheinst unsicher zu sein" | PointUser |
| "Komm %1 her %2" | "Ok, ich komme !" | GotoUser |
| "%1" | "Fuehlst du dich von Maschinen bedroht?" | Oh |

Die Suchmuster enthalten variable Anteile (Wildcards), die mit einem %-Zeichen und einer Nummer von 1 - 9 gekennzeichnet werden und denen beliebige Textteile zugewiesen werden können. Diese Textteile können im Antworttext und im Befehl referenziert werden. Bei Benutzereingaben werden alle geladenen Suchmuster auf ihre Gültigkeit geprüft (Pattern-Matching), wobei eine Liste passender Tripel erstellt wird, dann davon eines ausgewählt und zur Anwendung gebracht wird. Die Auswahl aus der Liste passender Tripel geschieht zufällig, um unvorhersehbares, vielfältiges und interessantes Dialogverhalten herzustellen. Allerdings wird zuvor eine Gewichtung der Tripel vorgenommen, die im wesentlichen von der Anzahl der übereinstimmenden, konstanten Worte im Suchmuster abhängt. So werden spezielle Regeln immer gegenüber allgemeinen Regeln bevorzugt.

Konkret habe ich folgende Gewichtungsfunktion g implementiert[6]:

Sei

trip_anz = Anzahl der passenden Tripel

konst_anz = Anzahl der übereinstimmenden, konstanten Wörter des Suchmusters

var_anz = Anzahl der Wildcards des Suchmusters

Dann ist

g(trip_anz,konst_anz,var_anz) = trip_anz*konst_anz+var_anz

Durch die Multiplikation der Anzahl der übereinstimmenden, konstanten Wörter des Suchmusters mit der Anzahl der passenden Tripel wird verhindert, daß eine große Zahl allgemeiner Regeln eine zu große Übermacht gegenüber wenigen speziellen Regeln erhält.

Normalerweise wird die Abbildung einer Texteingabe des Benutzers auf eine Antwort automatisch aufgerufen, wenn im Dialogfenster die RETURN-Taste oder der ANTWORT-Knopf gedrückt wird. Zusätzlich kann dieser Vorgang auch explizit innerhalb der Skriptsprache mit Hilfe der vordefinierten Prozedur *Answer* ausgelöst werden, etwa wenn eine Textnachricht über einen *EventIn* eingeht.

[6] aus Gründen der Performance habe ich auf eine Normierung verzichtet.

## 7.5 Anwendung der Skriptsprache und der Programmierschnittstelle

### 7.5.1 Einbeziehung von Umwelteinflüssen

Oft sollen Umwelteinflüsse in die Handlungsauswahl mit einbezogen werden, um reaktives Verhalten zu erzeugen. Solche Informationen über den Zustand der virtuellen Welt erreichen den automatisierten Charakter über seine *EventIns*, die immer dann einen Callback erzeugen, wenn sich der Zustand des mit ihnen verbundenen Datums verändert hat. Grundsätzlich können *EventIns* mit beliebigen Daten gleichen Typs in der virtuellen Welt verknüpft sein, zum Beispiel mit Vektoren, Flags oder Strings anderer Objekte sowie eigener Teilobjekte. Es existieren jedoch spezielle Sensorobjekte in VRML2, die ebenfalls Ereignisse erzeugen, insbesondere basierend auf Benutzeraktionen. Im einzelnen sind Sensoren für räumliche Nähe, für Anklicken, Kollision mit dem Benutzer oder Sichtbarkeit vorhanden. Andere Sensoren - sogenannte *TimeSensors* - sorgen für regelmäßige Zeittakte.

Einigen Anwendungen werden diese Standardsensoren jedoch nicht ausreichen. Sollen etwa Agenten entwickelt werden, die sich autonom in unbekannten Welten zurechtfinden, werden komplexere Sensoren benötigt, beispielsweise um Kollisionen mit anderen Objekten vorherzusehen. Solche spezialisierten Sensoren können durch separate Java-Klassen implementiert werden, die ihrerseits *EventOuts* erzeugen.

### 7.5.2 Möglichkeiten der Verhaltenssteuerung mit Hilfe der Skriptsprache

Die Möglichkeiten und Grenzen der Skriptsprache sollen an dieser Stelle auf der Grundlage der zuvor differenzierten Ebenen der Verhaltenssteuerung (☞7.1.4 Ebenen der Verhaltenssteuerung) untersucht werden.

Ein genereller Vorteil von Skriptsprachen ist die Tatsache, daß kein separater Kompilierungsvorgang anfällt: das Editieren, Laden und Speichern des Skriptprogrammes kann zur Laufzeit geschehen. Genereller Nachteil ist dagegen die allgemein schlechte Performance aller interpreterbasierten Sprachen.

Während die Java-VRML2-Schnittstelle selbst lediglich das Setzen von Schnittstellendaten wie zum Beispiel Vektoren gestattet, kann mit Hilfe der in der Skriptsprache vordefinierten Prozeduren zusätzlich auch eine lineare Interpolation von vorgegebenen Schlüsselvektoren vorgenommen werden. Dadurch wird die Animation von Objekten in virtuellen Welten wesentlich erleichtert. Außer-

dem wird die Synchronisation von parallel und sequentiell ablaufenden Einzelbewegungen unterstützt. Anfallende Berechnungen können mit Hilfe vordefinierter Funktionen und Operatoren durchgeführt werden, wobei auch inkrementelle Berechnungen mit Hilfe von Schleifen möglich und Zwischenergebnisse durch Variablen speicherbar sind. Die Beschreibung von komplexem Verhalten wird durch selbstdefinierbare Prozeduren und durch in Module aufteilbare Skriptprogramme erheblich entlastet, da so eine hierarchische Strukturierung der Handlungen machbar ist.

Die Handlungsauswahl kann bequem von Sensorinformationen abhängig gemacht werden, da zur Reaktion auf beliebige Ereignisse und Einflüsse jeweils Callbacks, das heißt Prozeduren in der Skriptsprache, aufgerufen werden und die aktuellen Zustände dieser Umwelteinflüsse immer als globale Variablen verfügbar sind.

In einem eingeschränkten Maße können auch explizite Ziele in die Handlungsauswahl einbezogen werden, falls sich diese durch Kenngrößen, Zielvektoren oder ähnliches als Variablen (z.B. Nahrung_soll, Nahrung_ist) beschreiben lassen. Allerdings existiert kein eingebauter Mechanismus zum Abgleich von Situationen und Zielen, der automatisch eine zielgerichtete Aktion ausführen würde.

Ebenfalls nicht direkt implementiert ist die Verwaltung von Charaktereigenschaften, mit der eine Variation des Verhaltens auf der Grundlage von Charakterparametern möglich wäre. Dies müßte etwas umständlich mit bedingten Befehlsausführungen an den entscheidenden Stellen des Skriptprogrammes in Abhängigkeit von Variablen, die den Charakter anhand von Kenngrößen beschreiben, gelöst werden.

Bei manchen Anwendungen ist adaptives Verhalten gefragt, das heißt, daß die Handlungen einer zielbezogener Optimierung unterliegen. Dies setzt eine Rückkopplung über das Erreichen von Zielen voraus. Da die Skriptsprache jedoch keine Verwaltung von Zielen vorgibt, kann auf dieser Ebene auch kein eingebauter Lernmechanismus angeboten werden. Wer jedoch innerhalb der Skriptsprache Ziele selbst verwaltet, kann durchaus auch adaptives Verhalten realisieren, indem er beispielsweise die Handlungsauswahl von einer Gewichtungsmatrix abhängig macht und die einzelnen Gewichte basierend auf dem Feedback über den Erfolg von Aktionen optimiert. Zusätzlich besteht die Möglichkeit, komplett neue Handlungsabläufe in Form von Prozeduren zur Laufzeit zu erzeugen oder alte Prozeduren durch neue zu ersetzen.

### 7.5.3 Beispielhafte Anwendung der Programmierschnittstelle: Der SchemeRobot

Ein wesentlicher Mangel der Skriptsprache, die von der Klasse VRRobot zur Verfügung gestellt wird, ist das Fehlen eines integrierten Verfahrens, das Verhalten abhängig von zur Laufzeit variierenden Bedingungen im Ganzen oder in Teilen durch neues, situationsgerechteres auszuwechseln.

Solche Bedingungen, die zu einer Veränderung des Verhaltens führen sollen, ergeben sich zum Beispiel aus Umwelteinflüssen, internen Zuständen oder der Interaktion mit dem Benutzer.

Dieser Mangel kann, wie im folgenenen ausgeführt, wegen der Verfügbarkeit der Java-Programmierschnittstelle einfach behoben werden, genauso wie auch andere Verbesserungen auf diesem Wege denkbar sind.

Um die Gestaltung von situationsabhängigem Verhalten zu erleichtern und um die Verwendung der Programmierschnittstelle zur einfachen Erweiterung der Funktionalität des *VRRobot* zu demonstrieren, habe ich den sogenannten *SchemeRobot* als Kindklasse von *VRRobot* entwickelt. Somit erbt dieser die Basisfunktionalität zur Interpretation der Skriptprogramme und zur Steuerung der automatisierten Charaktere, ergänzt aber ein Schemakonzept sowie eine spezielle Programmierumgebung zum Verwalten von Schemata. Diese Schemata fassen jeweils ein Skriptprogramm, eine Konfiguration des Dialogverhaltens und eine Bewertungsfunktion zusammen. Ein Hintergrundprozeß überprüft ständig anhand der Bewertungsfunktionen, welches der Schemata zur aktuellen Situation paßt und aktiviert dessen Skriptprogramm, wobei nicht nur dessen Hauptprozedur *VRRobotMain* ausgeführt wird, sondern auch die Liste der bekannten Prozeduren und Callbacks sowie die Dialogdefinition ersetzt wird. Die Bewertungsfunktionen sind normale Prozeduren der Skriptsprache, mit der Besonderheit, daß sie jeweils die globale Variable *rating* mit einem Zahlwert belegen müssen. Ansonsten dürfen beliebige Berechnungen, Kontrollstrukturen und Bezüge auf globale Variablen enthalten sein, um *rating* zu ermitteln.

Zusätzlich zu dieser Eigenbewertung können sich die Schemata gegenseitig in ihrer Bewertung positiv oder negativ beeinflussen, um die Aktivierung der jeweils anderen zu hemmen oder zu fördern. Für diesen Zweck erweitert *SchemeRobot* den Befehlssatz der Skriptsprache um die Prozedur *Inhibit*, die als Parameter einen Wert, der größer oder kleiner Null sein kann, und einen Schemanamen erwartet. Der Wert wird dann zu der Gesamthemmung des angesprochenen Schemas addiert; die Gesamtbewertung eines Schemas ergibt sich aus der Summe seiner Eigenbewertung und aus seiner Hemmung. Ein Schema kann den Befehl *Inhibit* auch auf sich selbst anwenden, um plötzlichen, ungewollten Unterbrechungen vorzubeugen oder aber um sich selbst zu deaktivieren. Auf diese Weise sind semantische Netze mit Verstärkungen und Hemmungen zwischen den Schemata aufbaubar.

Eine zusätzliche Strukturierung wird durch eine hierarchische Ordnung der Schemata erreicht, die mit einer speziellen Maske erzeugt wird. Dieser Dialog erlaubt die separate Vererbung von Bewertungsfunktion, Handlungsplan und Dialogverhalten von Schemata an Subschemata.


Abbildung 18: Maske zum Anzeigen der Schemahierarchie

Eine weitere Ergänzung der Skriptsprache stellt die Prozedur *CallScheme* dar, mit der ein expliziter Aufruf eines Schemas und die Übergabe von Parametern ermöglicht wird. Da in der Form aktivierte Schemata entgegen ihrer eigentlich zu niedrigen Bewertung ausgeführt werden, sind sie nicht vor Abarbeitung ihrer Hauptprozedur *VRRobotMain* zu stoppen, weswegen dieser Mechanismus nicht für zu langandauernde Schemata eingesetzt werden sollte.

### 7.5.4 Anwendungsbeispiel mit Hilfe von Skripts und Schemata

Um die Anwendbarkeit der von mir entwickelten Mechanismen zu demonstrieren, habe ich ein konkretes, kleines Beispiel entworfen. Dieses Beispiel ist für die Vorführung meiner Software während eines Vortrages über diese Arbeit optimiert, was sich insbesondere dadurch äußert, daß unvorhersehbares Dialogverhalten möglichst vermieden wird. Diese Eigenschaft der Vielfältigkeit, die in realen Fällen meist erwünscht ist, wäre in einer Vortragssituation eher störend. Auch die Komplexität des Handlungsplanes orientiert sich an dem Vorhaben, die Zusammenhänge während einer Vortragslänge beschreiben zu wollen.

Für die Demonstration meiner automatisierten Charakter habe ich ein virtuelles Museum mit einem Museumsführer entworfen (Abbildung 19: Virtuelles Museum mit Museumsführer). Es besteht aus drei durch Brücken verbundenen Plattformen, auf denen jeweils drei Bilder von einem Künstler präsentiert werden. Diese Werke von Paul Klee, Wassily Kandinsky und Henri Matisse sind auf den Außenflächen von freistehenden Quadern ausgestellt.


Abbildung 19: Virtuelles Museum mit Museumsführer

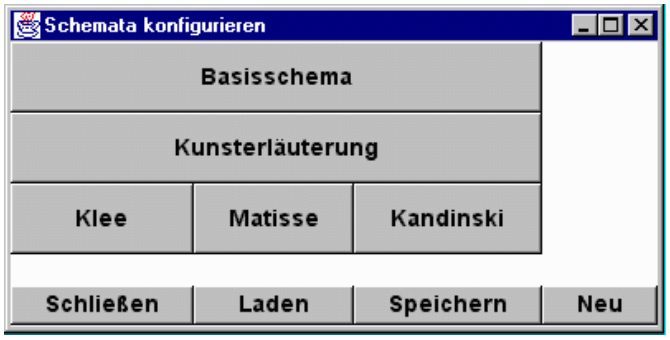



Abbildung 20: Schemahierarchie beim Museumsführer

Der Museumsführer wird durch einen SchemeRobot gesteuert, der je ein Schema für die Künstler bereithält (☞ Abbildung 20: Schemahierarchie beim Museumsführer). Der Handlungsplan ist in diesem einfachen Beispiel für die drei Schemata gleich und wird von einem gemeinsamen Elternschema („Kunsterläuterung“) geerbt. Lediglich das Dialogverhalten, das spezifische Antworten für den jeweiligen Künstler beinhaltet, unterscheidet die drei Schemata. Der gemeinsame Handlungsplan schreibt dem Museumsführer vor, dem Benutzer zu folgen, wenn dieser die Plattform wechselt. Außerdem wendet er sich ständig zur Position des Benutzers. Ein Teil des entsprechenden Skriptprogrammes ist im folgenden beispielhaft aufgeführt[7]:

```
procedure VRRobotMain
begin
GlobalVar eatlock false
StopWalk
FollowArea
end

procedure FollowArea
begin
  Tag Loop
   LocalVar userx %userpos.x
   LocalVar userz %userpos.z
   if ((%userx > -65)&(%userx < 65 )&(%userz > -65)&(%userz < 65 )) then GotoKleeArea
   if ((%userx > -65)&(%userx < 65 )&(%userz >  65)&(%userz < 195)) then GotoKandinskyArea
   if ((%userx >  65)&(%userx < 195)&(%userz > -65)&(%userz < 65 )) then GotoMatisseArea
   TurnToUser
   Wait slides 10
  Goto Loop
end

procedure GotoKleeArea
begin
if ((%robotpos.x>-65)&(%robotpos.x<65)&(%robotpos.z>-65)&(%robotpos.z<65)) then Goto ende
Say Warte, ich komme auch in den Bereich von Paul Klee!
Inhibit 100
GlobalVar eatlock true
StartWalk
if (%robotpos.x>65)&(%robotpos.x<195)&(%robotpos.z>-65)&(%robotpos.z<65) then Goto FromMatisse
if (%robotpos.x>-65)&(%robotpos.x<65)&(%robotpos.z>65)&(%robotpos.z<195) then Goto FromKandin
Goto talkpos
Tag FromMatisse
Go x 100, z 0, startwalk false, stopwalk false
Go x 30, z 0, startwalk false, stopwalk false
goto talkpos
Tag FromKandin
Go x 0, z 100, startwalk false, stopwalk false
Go x 0, z 30, startwalk false, stopwalk false
Tag talkpos
Go x -25, z 25, startwalk false, stopwalk true
Inhibit -100
GlobalVar eatlock false
Tag ende
end     ...
```

[7] Das komplette Skript ist im Anhang enhalten.


Abbildung 21: Schema für den Klee-Bereich

Die Bewertungsfunktionen der Schemata sorgen dafür, daß abhängig davon, auf welcher Plattform sich der Museumsführer gerade befindet, das entsprechende Künstler-Schema aktiviert ist (☞ Abbildung 21: Schema für den Klee-Bereich).

Sobald der Museumsführer dem Benutzer auf eine andere Plattform gefolgt ist, bewirken die von der Position des Museumsführers abhängigen Bewertungsfunktionen einen Schemawechsel. Der Benutzer kann nun vom Museumsführer Informationen zum jeweiligen Künstler beziehungsweise zu dessen Bildern erfragen. Durch die Situationsabhängigkeit wird erreicht, daß etwa die Frage „Wann hat er gelebt?“ immer mit dem richtigen Künstler assoziiert wird.

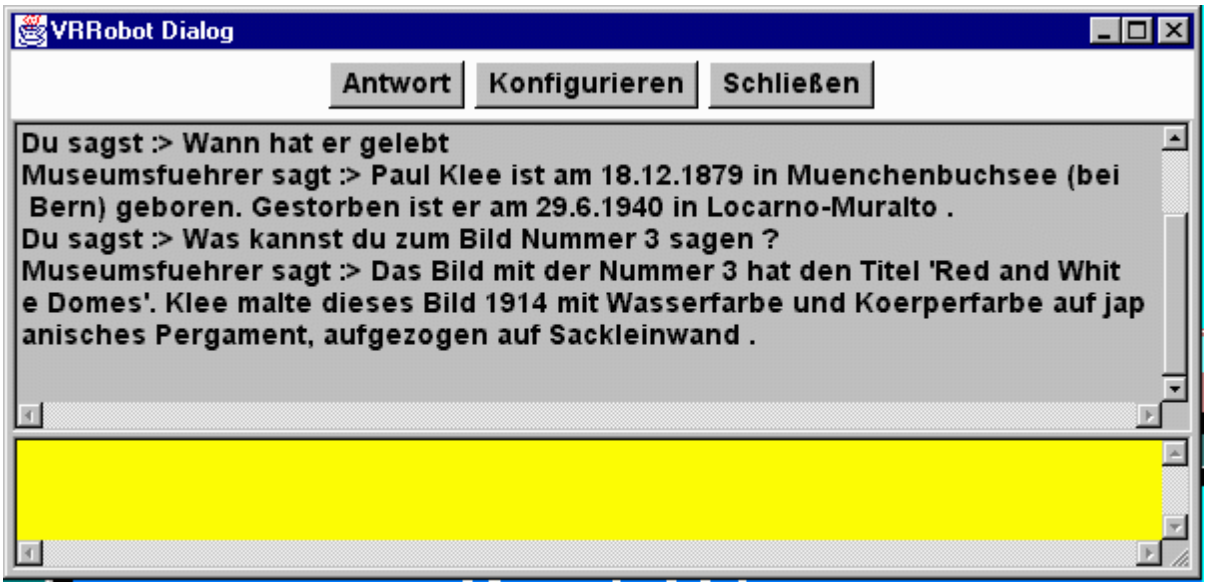



Abbildung 22: Dialog mit dem Museumsführer im Bereich 'Paul Klee'

Ein besonderer Zeitgeber (*EventIn*) bewirkt in diesem Beispiel ein regelmäßiges „Hungergefühl", das den Museumsführer veranlaßt, eine Futterstelle auf der Klee-Plattform aufzusuchen und danach zur Ausgangsposition zurückzukehren. Dies demonstriert den Nutzen des Befehls *Inhibit*, dessen Einsatz an dieser Stelle zur Folge hat, daß kein Schemawechsel zugunsten des Klee-Schemas stattfindet, obwohl die Bewertungsfunktionen genau das fordern würden. Mit Hilfe des Befehls *Inhibit* wird hier einer Unterbrechung der Aktion vorgebeugt, indem eine Hemmung für andere Schemata aufgebaut wird (☞ 7.5.3 Beispielhafte Anwendung der Programmierschnittstelle: Der SchemeRobot).

# 8 Ausblick

Im Rahmen dieser Arbeit sind Javaklassen entstanden, die die Beschreibung von reaktivem Verhalten für automatisierte Charaktere in virtuellen Welten mit Hilfe einer Skriptsprache erlauben. Diese bietet Unterstützung bei der Definition von Akteuren in virtuellen Welten, indem sie Animationsdesign erleichtert, den Aufbau von Prozedurbibliotheken für Handlungsabläufe gestattet und einfaches Gesprächsverhalten konfigurierbar macht. Der Schwerpunkt der Arbeit war es dabei nicht, für Teilprobleme optimale Lösungen zu suchen, sondern eine Plattform für Weiterentwicklungen zu schaffen. So können spezialisierte Module in Form von abgeleiteten Klassen eingebracht werden oder nach dem Vorbild von Agentenarchitekturen als parallel ablaufende Prozesse realisiert werden. Für diesen Zweck bietet die Skriptsprache Mechanismen für den Austausch von Botschaften mit anderen Agenten an.

Auf diese Weise lassen sich spezialisierte Sensoren - beispielsweise zur Kollisionsvermeidung beim Navigieren in unbekannten Welten - oder Interaktionsformen wie etwa die Spracheingabe per Mikrophon ergänzen. Auf der anderen Seite ist eine Verfeinerung der Bewegungsabläufe mit Hilfe von Verfahren wie der Inversen Kinematik, der Gesichtsanimation oder ähnlichem durch ein Umleiten der Ausgangsvektoren zu entsprechenden Experten denkbar.

Die von mir implementierte, objektorientierte Programmierschnittstelle, die durch Ableiten von der Klasse *VRRobot* genutzt werden kann, dient der grundsätzlichen Erweiterung der Verhaltenssteuerung um weitergehende Mechanismen und Konzepte, wenn Aufgabenstellungen nicht oder nur unbequem mit der Skriptsprache oder parallel ablaufenden Agenten lösbar sind. So habe ich anhand des *SchemeRobot* gezeigt, wie einerseits die Skriptsprache erweitert und andererseits ein fortgeschrittener Mechanismus zur Handlungsauswahl eingebracht werden kann. Ich stelle mir vor, daß auf demselben Weg beispielsweise auch KI-Methoden zum Sprachverstehen, der Wissensrepräsentation oder dem Lernen aus Erfahrung integrierbar sind.

Eine andere Möglichkeit der Ergänzung meiner Implementation ist die Einbeziehung künftiger Standards für Avatare und Agenten in virtuellen Welten. Dies betrifft insbesondere die Interaktionsformen zwischen menschlichen, aber auch automatisierten Akteuren, etwa auf der Basis von Textdialogen oder sogar Gesten. Die hierfür notwendigen Änderungen sollten durch die Herstellung entsprechender Unterklassen mit partiell modifizierter Funktionalität realisierbar sein.

Einen Überblick über den erreichten Stand der Entwicklung und über die angedeuteten, denkbaren Weiterentwicklungen soll das abschließende Schaubild geben:

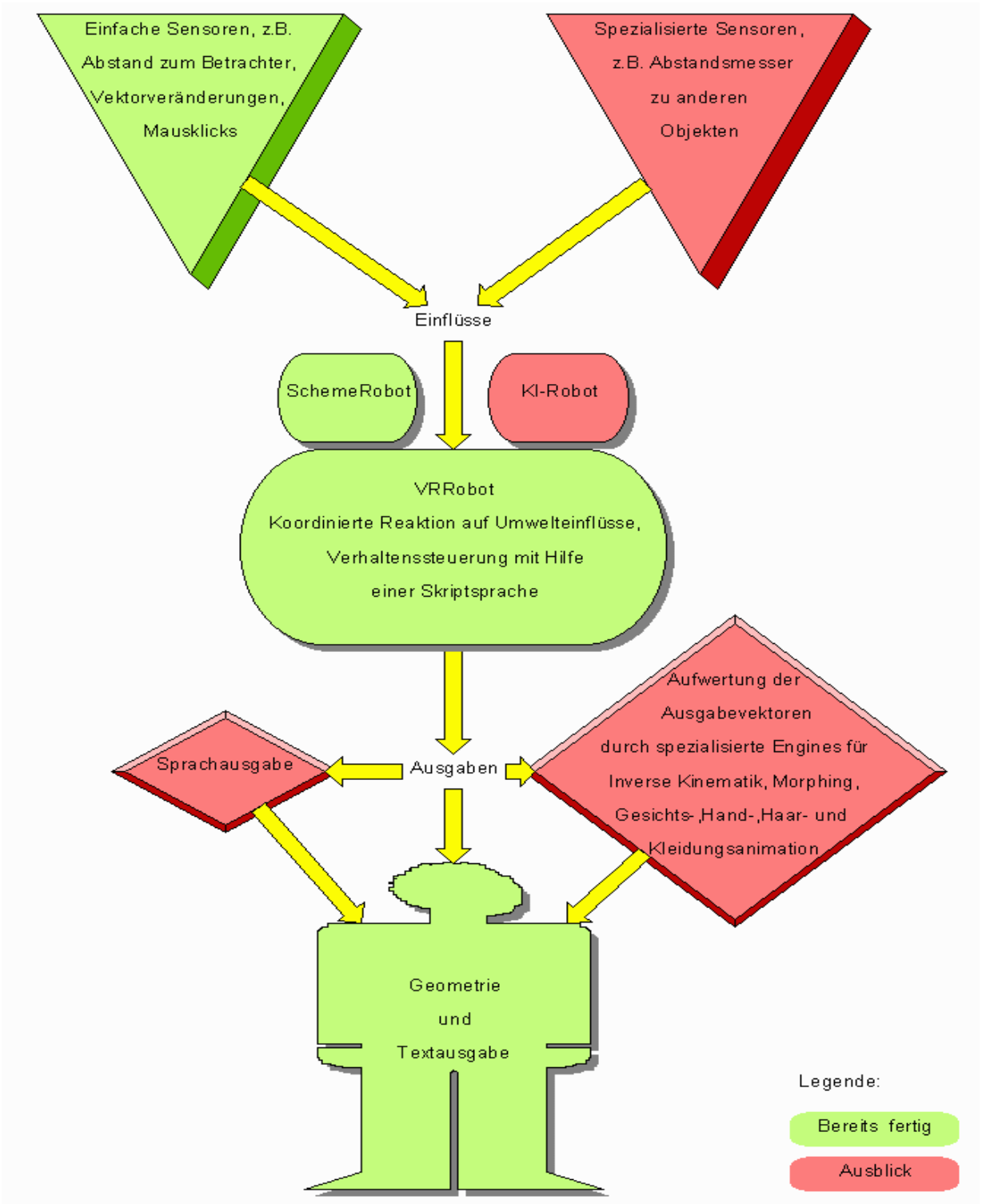



Abbildung 23: Mögliche Weiterentwicklungen

# LITERATURVERZEICHNIS


[AER96] Aereal Inc. The Personalized Internet marketing & Publishing Company. In: *WWW* - Online Präsentation, 12.1996 (http://www.aereal.com/)

[AIK96] AIKIU. In: *WWW* - Online Präsentation, 12.1996 (http://www.aikiu.de/)

[ALG96] The Artificial Life Group At MIT. In: *WWW* - Online Präsentation, 12.1996 (http://www.ai.mit.edu/people/emaley/alife/)

[ANI96] Animators Mailing List. In: *WWW* - Online Präsentation, 12.1996 (http://www.xmission.com/~grue/animate/index.html)

[BAD86] BADLER, NORMAN I./TSOTSOS, JOHN K. (HG.): *Motion: Repräsentation and Perception.* New York: Elesevier Science Publishers B.V., 1986

[BAD95] BADLER, NORMAN I. U.A.: The Center for Human Modeling and Simulation (1995). In: *WWW* - Report als Postscript-Datei. Philadelphia: Department of Computer & Information Science, University of Pennsylvania, 12.1996 (http://www.cis.upenn.edu/~graphics/)

[BEN91] BENEDIKT, M. (HG.): *Cyberspace.* First Steps. London: MIT Press, 1991

[BLU95] BLUMBERG, BRUCE M. / GALYEAN, TINSLEY A.: Multi-Level Direction of Autonomous Creatures for Real Time Virtual Environments (1995). In: *WWW* - Report als Postscript-Datei. Cambridge: MIT Media Lab, 02.1997 (http://lcs.www.media.mit.edu/groups/agents/papers.html)

[CEN96] Center for Human Modeling and Simulation. In: *WWW* - Online Präsentation, 12.1996 (http://www.cis.upenn.edu/~hms/)

[CHA96] The Character Shop. In: *WWW* - Online Präsentation, 12.1996 (http://www.character-shop.com/)

[CHR96] Chris´s Virtual Reality Stuff. In: *WWW* - Online Präsentation, 12.1996 (http://www.cms.dmu.ac.uk/~cph/vrstuff.html)

[COR96] CORNELL, GARY / HORSTMANN, CAY S.: *Core Java*. Mountain View: Sun Microsystems, Inc., 1996

[CYB96] Cyberartist VRML2. In: *WWW* - Online Präsentation, 12.1996 (http://www.cyberartist.com/vrml2/)

[DES96] DesignSpace. In: *WWW* - Online Präsentation, 12.1996 (http://gummo.stanford.edu/html/DesignSpace/home.html)

[DÖB96] DÖBEN-HENISCH, GERD: Hören, riechen, hungrig sein. Philosophie mit einem neuen Typ künstlicher Intelligenz (12.03.1996). In: *WWW* - Online Präsentation: Frankfurter Rundschau, 07.04.1997 (http://www.inm.de/inm_info/fr_120396kip.html)

[EBE94] EBERLEH, EDMUND / OBERQUELLE, HORST / OPPERMANN, REINHARD: *Einführung in die Software-Ergonomie*. Gestaltung graphisch-interaktiver Systeme: Prinzipien, Werkzeuge, Lösungen. 2. Auflage. Berlin, New York: Walter de Gruyter, 1994

[EVA67] EVANS, S.H.: A brief statement of schema theory. In: *Psychonomic Science*, 8.1967 (S. 87-88)

[FAL97] FALK, LORNE: Strange, Wonderfull Worlds of Sommerer & Mignonneau. In: *WWW* - Online Präsentation, 12.04.97 (http://www.mic.atr.co.jp/~christa/SGIpage.html)

[FAS94] FAßLER, MANFRED / HALBACH, WULF R. (HG.): *Cyberspace*. Gemeinschaften, Virtuelle Kolonien, Öffentlichkeit. München: Wilhelm Fink Verlag, 1994

[FAU95] FAULSTICH, WERNER (HG.): *Grundwissen Medien*. 2. Auflage. München: Wilhelm Fink Verlag, 1995

[FIS95] FISCHER, JOACHIM / GENSIOR, SABINE (HG.): *Netzspannungen*. Trends in der sozialen und technischen Vernetzung von Arbeit. Berlin: Edition Sigma, 1995

[FOL93] FOLEY, JAMES / VAN DAM, ANDRIES U.A.: *Computer Graphics*. Principles and Practice. 2. Aufl.: Addison-Wesley, 1993

[FRA96] FRANKLIN, STAN / GRAESSER, ART: Is it an Agent, or just a Program?. A Taxonomy for Autonomous Agents (1996). In: *WWW* - Online Präsentation. Institut for Intelligent

Systems. University of Memphis: Springer Verlag, 23.04.1997 (http://www.msci.memphis.edu/~franklin/AgentProg.html)

[GAM96] Gamelan, The Official Directory for Java. In: *WWW* - Online Präsentation, 12.1996 (http://www.gamelan.com/index.shtml)

[GIB96] GIBSON, WILLIAM: *Die Neuromancer Trilogie*. Neuromancer / Biochips / Mona Lisa Overdrive. 1. Aufl.. Frankfurt: Zweitausendeins, 1996

[HAM90] HAMMOND, GEOFFREY R. (HG.): *Celebral Control of Speech and Limb Movements*. New York: Elsevier science Publishers B.V., 1990

[HAY94] HAYES-ROTH, BARBARA / MORIGNOT, PHILIPPE: Why Does an Agent Act? (1994). In: *WWW* - Report als Postscript-Datei. Stanford: Knowledge System Laboratory (KSL 94-76), 12.1996 (http://www-ksl.stanford.edu/)

[HAY95A] HAYES-ROTH, BARBARA / VAN GENT, ROBERT: Story-Making with Improvisational Puppets and Actors (1995). In: *WWW* - Report als Postscript-Datei. Stanford: Computer Science Department, 12.1996 (http://www-ksl.stanford.edu/)

[HAY95B] HAYES-ROTH, BARBARA / MORIGNOT, PHILIPPE: Goal Generation and Revision für Planning Agents in Unpredictable Environments (1995). In: *WWW* - Report als Postscript-Datei. Stanford: Knowledge System Laboratory (KSL 95-75), 12.1996 (http://www-ksl.stanford.edu/)

[HAY95C] HAYES-ROTH, BARBARA / MORIGNOT, PHILIPPE: Adaptable Motivational Profiles for Autonomous Agents (1995). In: *WWW* - Report als Postscript-Datei. Stanford: Knowledge System Laboratory (KSL 95-01), 12.1996 (http://www-ksl.stanford.edu/)

[HAY95D] HAYES-ROTH, BARBARA / BROWNSTON, LEE / SINCOFF, ERIK: Direct Improvisation by Computer Characters (1995). In: *WWW* - Report als Postscript-Datei. Stanford: Knowledge System Laboratory (KSL 95-04), 12.1996 (http://www-ksl.stanford.edu/)

[HAY96A] HAYES-ROTH, BARBARA / VAN GENT, ROBERT: Improvisational Puppets, Actors, and Avatars (1996). In: *WWW* - Report als Postscript-Datei. Stanford: Computer Science Department, 12.1996 (http://www-ksl.stanford.edu/)

[HAY96B] HAYES-ROTH, BARBARA / VAN GENT, ROBERT / HUBER, DANIEL: Acting in Character (1996). In: *WWW* - Report als Postscript-Datei. Stanford: Computer Science Department, 12.1996 (http://www-ksl.stanford.edu/)

[HAY96C] HAYES-ROTH, BARBARA / MORIGNOT, PHILIPPE: Motivated Agents (1996). In: *WWW* - Report als Postscript-Datei. Stanford: Knowledge System Laboratory (KSL 96-22), 12.1996 (http://www-ksl.stanford.edu/)

[HAY96D] HAYES-ROTH, BARBARA / DOYLE, PATRICK: An Intelligent Guide for virtual Environments (1996). In: *WWW* - Report als Postscript-Datei. Stanford: Knowledge System Laboratory (KSL 96-20), 12.1996 (http://www-ksl.stanford.edu/)

[HOF93] HOFFMANN, JOACHIM: *Vorhersage und Erkenntnis*. Göttingen: Hogrefe, 1993

[HOM90] HOMMEL, BERNHARD: *Quellen der Interferenz beim Simon-Effekt*. Eine Untersuchung zur Verwendung räumlicher Information bei der Auswahl und Planung einer einfachen Handlung - Dissertation. Bielefeld: Univ., 1990

[KEL82] KELSO, SCOTT / CLARK, JANE E. (HG.): *The Development of Movement Control and Co-ordination*. John Wiley & Sons Ltd., 1982

[KLI87] KLIX, F. / VAN DER MEER, ELKE / PREUß, M. / WOLF, M.: Über Prozeß- und Strukturkomponenten der Wissensrepräsentation beim Menschen. In: *Zeitschrift für Psychologie 195*, 1987 (S. 39 - 61)

[KWA96] KWAN W.: Human Arm Animator (1996). In: *WWW* - Online Präsentation, 11.1996 (http://www.cs.curtin.edu.au/~chinkw/html/ik)

[LEE93] LEE, DAVIS S. / LEIFER, LARRY J.: A Graphical Programming Language for Robots Operating in Lightly Stuctured Environmnents (1993). In: *WWW* - Report als Postscript-Datei. Stanford: Center for Design Research, Department of Mechanical Engineering, Stanford University, 12 (http://www-ksl.stanford.edu/)

[LIV96] LivingWorlds. In: *WWW* - Online Präsentation, 12.1996 (http://www.livingworlds.com/draft_1/index.html)

[LOE93] LOEFFLER, CARL E.: Distributed Virtual Reality. Applications for education, entertainment and industry (1993). In: *WWW* - Online Präsentation, 13.02.1997 (http://www.nta.no/telektronikk/4.93.dir/Loeffler_C_E.html)

[MA96] MA, MOSES / GREENING, DAN, U.A.: Universal Avatars (1996). In: *WWW* - Online Präsentation, 08.1996 (http://www.chaco.com/avatar/avatar.html)

[MAE95A] MAES, PATTIE: Artificial Life Meets Entertainment. Lifelike Autonomous Agents (1995). In: *WWW* - Online Präsentation. Cambridge: MIT Media-Laboratory, 02.1997 (http://pattie.www.media.mit.edu/people/pattie/CACM-95/alife-cacm95.html)

[MAE95B] MAES, PATTIE: Modeling Adaptive Autonomous Agents (1995). In: *WWW* - Report als Postscript-Datei. Cambridge: MIT Media-Laboratory, 02.1997 (http://pattie.www.media.mit.edu/people/pattie/cv.html#publications)

[MEY91] MEYER, JEAN-ARCADY / WILSON, STEWART W. (HG.): *From animals to animats*. Proceedings of the first international conferende on simulation of adaptive behavior (1991). Cambridge: MIT Press

[NAT96] Nats Virtual Reality Index. In: *WWW* - Online Präsentation, 12.1996 (http://scorch.doc.ic.ac.uk/~np2/virtual_reality/indes.html)

[NET96] NETZVISION: *Villa Utopia*. Ein Projekt unter der Betreuung von Coy, Wolfgang / Pirr, Uwe / Heimbucher, Achim - Projektbericht auf CD-ROM. Bremen: Universität Bremen, 1996

[NEU86] NEUMANN, JOHN VON: *Die Rechenmaschine und das Gehirn*. 5. Aufl.. München: Oldenbourg, 1986

[NEU89] NEUMANN, ODMAR: Kognitive Vermittlung und direkte Parameterspezifikation. Zum Problem mentaler Repräsentation in der Wahrnehmung. In: *Sprache und Kognition 8*, 1989 (S. 32 - 49)

[NEU90] NEUMANN, O. / PRINZ W. (HG.): *Relationschips Between Perception and Action*. Current Approaches. Berlin, heidelberg: Springer Verlag, 1990

[NEU92] NEUMANN, ODMAR: Theorien der Aufmerksamkeit. Von Methapern zu Mechanismen. In: *Psychologische Rundschau 43*, 1992 (S. 83 - 101)

[OPA94] OPASCHOWSKI, HORST W.: *Schöne, neue Freizeitwelt ?*. Wege zur Neuorientierung - Projektstudie. Hamburg: BAT Freizeit-Forschungsinstitut, 1994

[OZ96] OZ.Inc.. In: *WWW* - Online Präsentation, 12.1996 (http://www.oz-inc.com/main_bot.html)

[PRI70] PRINZ, WOLFGANG: *Untersuchungen zur Funktionsanalyse visueller Erkennungsprozesse bei mehrdimensional variierendem Figurenmaterial*- Dissertation. Bochum: Univ., 1970

[PRI84] PRINZ, W. / SANDERS, A. F. (HG.): *Cognition and Motor Process*. Berlin, Heidelberg: Springer Verlag, 1984

[PRI94] PRINZ, A.: *Bildverstehen*. Wien, New York: Springer, 1994

[PRO96] Professional Group SIGGRAPH Mexico City. In: *WWW* - Online Präsentation, 12.1996 (http://www.siggraph.org.mx/sighomei.html)

[ROB96] ROBERTSON, BARBARA: Best Behaviors (1996). In: *WWW* - Online Präsentation, 08.1996 (http://www.cgw.com/cgw/Archives/Magic/08/08story1.html)

[ROB96] Robotik und Prozeßinformatik - EDDA. In: *WWW* - Online Präsentation, 12.1996 (http://www.cs.tu-bs.de/rob/projects/edda/)

[ROS79] ROSENBAUM, DAVID A.: *Processes of human movement initiation*. Ann Arbor, Mich: Univ. Microfilms Internat., 1979

[RÖT93] RÖTZER, FLORIAN / WEIBEL, PETER (HG.): *Cyberspace: Zum medialen Gesamtkunstwerk*. München: Klaus Boer Verlag, 1993

[ROU96] ROUSSEAU, DANIEL / HAYES-ROTH, BARBARA: *Personality in Synthetic Agents* (1996). Stanford: Knowledge System Laboratory (KSL 96-21)

[SCH82] SCHMIDT, RICHARD A.: *Motor Control and Learning*. A Behavior Emphasis. Campain, Ill.: Human Kinetics Publ., 1982

[SCH90] SCHMIDT, RICHARD A.: *Eine Schematheorie über das Lernen diskreter motorischer Fertigkeiten* Kassel, 1990

[SHAN76] SHANNON, CLAUDE E. / WEAVER, WARREN: *Mathematische Grundlagen der Informationstheorie*. München: Oldenbourg Verlag, 1976

[SIL96] Silicon Graphics. In: *WWW* - Online Präsentation, 12.1996 (http://vrml.sgi.com/)

[SIM96] Simulation-based Training and Technical Documentation for the Web. Inflorescence, Inc.. In: *WWW* - Online Präsentation, 12.1996 (http://besoft.com/index.html)

[SOM97A] SOMMERER, CHRISTA / MIGNONNEAU, LAURENT: Hompage of Christa Sommerer and Laurent Mignonneau. In: *WWW* - Online Präsentation, 12.04.97 (http://www.mic.atr.co.jp/~christa)

[SOM97B] CHRISTA SOMMERER: Kunst als lebendiges System (14.02.1997). In: *WWW* - Online Präsentation, 7.4.1997 (http://www01.ix.de/tp/vag/6066/1.htm)

[SPE95] SPERLICH, TOM: Digitale Kreaturen. Charakteranimation auf dem Weg zum "virtuellen Menschen". In: *c't Magazin für Computer Technik*. Heft 3: Heise Verlag, 1995 (S. 92-105)

[STA96] Stanford Knowledge Systems Laboratory home page. In: *WWW* - Online Präsentation, 12.1996 (http://www-ksl.stanford.edu/)

[STE96] STEINBUCH, A.: VRML-Focus. In: *WWW* - Online Präsentation, 12.1996 (http://www.ask.uni-karlruhe.de/~doc/vrml/fokus/fokus.html)

[TAG96] Tag/Traum. In: *WWW* - Online Präsentation, 12.1996 (http://www2.tagtraum.de/tagtraum/)

[THA93] MAGNENAT-THALMANN, NADJA / THALMANN, DANIEL: The World of Virtual Actors. In: *Virtual Worlds and Multimedia*. John Wiley and Sons Ltd., 1993

[THA95] MAGNENAT-THALMANN, NADJA / THALMANN, DANIEL U.A.: The HUMANOID Environment for Interactive Animation of Multiple Deformable Human Characters. In: *EUROGRAPHICS '95*. Volume 14, Number 3: Eurographics Association, 1995

[TRA96] TRANSOM TECHNOLOGIES, INC.: Transom Jack Home Page. Virtual people. Virtual places. Real solutions.. In: *WWW* - Online Präsentation, 12.1996 (http://www.transom.com)

[VIE96] Viewpoint Datalabs International, Inc.. In: *WWW* - Online Präsentation, 12.1996 (http://www.viewpoint.com/)

[VIL96] The World Wide Web Virtual Library. Computer Graphics and Visualization. In: *WWW* - Online Präsentation, 12.1996 (http://www.dataspace.com/WWW/vlib/comp-graphics.html)

[VIR96A] Virtual Society on the Web. Sony Corporation. In: *WWW* - Online Präsentation, 12.1996 (http://vs.spiw.com/vs/)

[VIR96B] Virtual Reality Designers. StellStudio. In: *WWW* - Online Präsentation, 12.1996 (http://www.marketcentral.com/vrml/vrml.html)

[VIR96C] Virtual People, Virtuel Places, Real Solutions. Transcom Technologies, Inc.. In: *WWW* - Online Präsentation, 12.1996 (http://www.transom.com)

[VRM96A] VRML Review. Imaginative Entertainment. In: *WWW* - Online Präsentation, 12.1996 (http://www.imaginative.com/Vresources/vrml/)

[VRM96B] VRML Models. In: *WWW* - Online Präsentation, 12.1996 (http://www.ocnus.com/models/models.html)

[VRM96C] VRML Architecture Group. In: *WWW* - Online Präsentation, 12.1996 (http://vag.vrml.org)

[VRM96D] The Virtual Reality Modeling Language Specification (12.1996). In: *WWW* - Online Präsentation, 12.1996 (http://vrml.sgi.com/moving-world/)

[WAR94] WARNECKE, H.-J. / BULLINGER, H.-J. (HG.): *Virtual Reality '94. Anwendungen und Trends*. IPA /IAO Forum. Institutszentrum der Fraunhofer-gesellschaft Stuttgart-Vaihingen: Springer Verlag, 1994

[WAT92] WATT, ALAN / WATT, MARK: *Advanced Animation and Rendering Techniques, Theory and Praxis*. New York: Addison-Wesley, 1992

[WEI78] WEIZENBAUM, JOSEPH: *Die Macht der Computer und die Ohnmacht der Vernunft* 1. Aufl.. Frankfurt am Main: Suhrkamp, 1978

[WES96] WESTPHAL, VOLKER: Grundlagen Genetischer Algorithmen (1996). In: *WWW* - Vorlesungsskript. Jena: FSU Jena, Institut für Informatik, WS 1995/96, 2.1997 (http://www.minet.uni-jena.de/www/fakultaet/schukat/GA/index.html)

[WOO95] WOOLDRIDGE, MICHAEL J. / JENNINGS, NICHOLAS R. (HG.): *Intelligent Agents*. Berlin: Springer, 1995

[YAH96] Yahoo! - Computer and Internet: Graphics: Computer Animation. In: *WWW* - Online Präsentation, 12.1996 (http://www.yahoo.com/Computers_and_Internet/Graphics/Computer_Animation/)

[ZEP96] ZEPEDA, ARNULFO: Synthetec Actors. In: *WWW. Newsletter SIG Mex* - Online Präsentation, 12.1996 (http://www.siggrapg.org.mx/sm-bol2i.html)

[ZIE88] ZIEßLER, M. / HÄUEL, KATHRIN / HOFFMANN, J.: Die Programmierung struktureller Eigenschaften von Bewegungen. In: *Zeitschrift für Psychologie 196*, 1988 (S. 371 - 388)

# ABBILDUNGSVERZEICHNIS

# INDEX

## *F*



## *G*



## *H*



## *I*



## *J*



## *K*



## *L*



## *M*

## *N*



## *O*



## *P*



## *R*



## *S*

## *T*



## *U*



## *V*



## *W*



## *Z*

Hiermit versichere ich, daß ich die Arbeit zur Erlangung des akademischen Grades

**Diplom Informatiker**

selbständig verfaßt und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Diese Versicherung gilt ebenfalls für Software, Bilder und ähnliches.

Bremen, den 24. Juli 1997

Hauke Ernst