

# DIPLOMARBEIT

## ARCHITECTURE\_ENGINE\_1.0

Annäherung an Entwurfsanwendungen für Architekten auf  
der Basis programmierbarer Game Engines

ausgeführt zum Zwecke der Erlangung des akademischen  
Grades eines Diplom-Ingenieurs.

unter der Leitung von:

Univ. Prof. Arch. Dipl. Ing. Manfred-Wolff Plottegg

Univ. Ass. Dipl. Ing. Dipl.- Soz. Harald Trapp

Wiss.MA. Dipl. Ing. Christoph Falkner

E253/1 - Institut für Architektur und Entwerfen

Abt. Gebäudelehre

eingereicht an der Technischen Universität Wien

Fakultät für Architektur und Raumplanung

von Jochen Hoog

Mat. Nr.: 9626807

Kaiserstraße 77/10, 1070 Wien

Wien, im September 2004

# Vorwort

*„Forschung und Rechtfertigung haben zwar zahlreiche Einzelziele, aber kein überwölbendes Ziel namens Wahrheit.“ (Rorty 1994: 30)*

„Mir ist mal etwas ganz Eigenartiges passiert: Ich hatte auf einmal vergessen, was eher kommt - sieben oder acht.

Ich ging zu den Nachbarn und fragte, was sie meinten. Aber wie groß war meine Verwunderung, als sich plötzlich herausstellte, daß auch sie die Reihenfolge der Zahlen vergessen hatten. 1, 2, 3, 4, 5 und 6 wußten sie noch, aber wie weiter, hatten sie vergessen.

Wir gingen zusammen zum Kaufhaus »Gastronom« in der Snamenskaja, Ecke Bassejnaja, und fragten die Kassiererin. Die Kassiererin lächelte traurig, nahm ein kleines Hämmerlein aus dem Mund, zog die Luft durch die Nase ein und sagte: »Meines Erachtens kommt sieben in dem Fall nach acht, wenn acht nach sieben kommt.«

Erfreut bedankten wir uns bei der Kassiererin und liefen hinaus. Doch plötzlich, als wir uns die Auskunft der Kassiererin genauer überlegten, verstummten wir wieder, denn sie kam uns völlig sinnlos vor. Was tun? Wir gingen in den Sommergarten und fingen an, die Bäume zu zählen. Doch als wir bei sechs angelangt waren, blieben wir stehen und gerieten in Streit. Nach Ansicht der einen folgte sieben, nach Ansicht der anderen acht.

Wir würden noch lange gestritten haben, aber zum Glück fiel ein Kind von der Bank und brach sich beide Kiefer. Das brachte uns von unserem Streit ab. Da trennten wir uns und gingen nach Hause.

12. November 1935“

(Charms 2003)

# Einleitung

*„The kind of work I do is one in which I'm not the master of my material.“*

(Samuel Beckett)

*Grundlage dieser Diplomarbeit ist die Auseinandersetzung mit einem für Architekten noch neuen Themengebiete: des Computerspiels.*

*Der Umsatz der weltweiten Computerspielbranche kletterte 2003 nach Angaben des Marktforschungsunternehmens DFC Intelligence auf den Rekordwert von 19,2 Mrd. Euro und erreichte damit fast das Niveau der Filmindustrie. (Knauer 2004)*

*Die Technologie, die es ermöglicht virtuelle Umgebungen in Echtzeit auf den Bildschirm zu bringen, ist zuverlässig, schnell, billig und leicht zu bekommen. Dies macht sie neben Künstlern, Theoretikern und Horden von programmierenden Jugendlichen auch für Architekten interessant. Tatsache ist, dass diese Branche noch jung und unerforscht ist. Fragen über das Spiel als kulturelles, soziales oder ökonomisches Phänomen sind weitgehend unbeantwortet.*

*Mein Interesse in dieser Arbeit gilt im allgemeinen der Architektur in Computerspielen, wie der Wirkung der virtuellen Räume auf den Spieler, sowie im besonderen die Entwicklung einer Architekturentwurfsanwendung für Architekten.*

# Zieldefinition

Ziel dieser Arbeit ist es generelle Möglichkeiten einer Anwendung auf Basis einer programmierbaren Game Engine aufzuzeigen. Welche Möglichkeiten bietet diese Technologie für den Entwurf und für die Darstellung von Architektur? Wie funktioniert sie, welche Komponenten benutzt sie, wie denken Programmierer und warum spielen wir überhaupt?

Welche entwurfsrelevanten Eigenschaften bietet diese Technologie?

Wie kann diese Technologie als Tool oder Game zum Entwerfen von Architektur überhaupt genutzt werden?

Wie könnten architektonische Ergebnisse aussehen und in wie weit wird Architektur anders entworfen.

Wo liegen grundlegende Unterschiede zu Simulationsanwendungen wie „The Regionmarker“ oder „Function Mixer“ von MVRDV?

Warum kann ein Entwurf gespielt werden?

Was ist das Ziel?

Kann man auch verlieren?

Dabei wird diese Technik als neues Medium benutzt und nicht als begehbare dreidimensionale CAD Umgebung.

# Methodik

Ziel der Arbeit ist es meine gewonnenen Erkenntnisse aus dem Bereich Programmierung und Computerspiel als Architekt zu verarbeiten und als Entwurfsmethodik umzusetzen.

Einarbeitung in Programmiersprachen, insbesondere C++ sowie C# .

Einarbeitung in die Game Engine von 3D Game Studio.

Quellenanalysen zu den Themengebieten: Computerspiele, Virtualität, Raum, Modding, Spieltheorien, Code generierte Architekturen, Informations- und Kommunikationstechnologien.

Der erste Teil dieser Arbeit wird kurz und selektiv auf theoretische Kontexte des Themengebietes dieser Diplomarbeit eingehen und einen Abriss der Geschichte von Computerspielen geben.

Im Hauptteil wird die Technologie der Computerspiele eingehender beschrieben, um die Funktionsweise der entwickelten Anwendung ARCHITECTURE\_ENGINE\_1.0 besser nachvollziehen zu können. Danach werden generelle Möglichkeiten der Anwendung beschrieben und ein möglicher Anwendungs-

verlauf gezeigt und im letzten Teil die Ergebnisse im Sinne der definierten Zielsetzung erläutert.

Im Anhang dieser Diplomarbeit befindet sich eine CD mit einer spielbaren Version der ARCHITECTURE\_ENGINE\_1.0, sowie die Software von 3D Game Studio Version 6.0 als Probeversion. Ebenso befinden sich dort alle von mir entwickelten Programmscripts.

# Inhaltsangabe

//Vorwort //

//Einleitung //

//Zieldefinition //

//Methodik //

//1.//Theoretische Kontexte // 15

1.1 Spieldefinitionen 15

1.1.1 Klassische und moderne Spieltheorien 16

1.2 Why do I play and cannot stop? 17

1.2.1 Macht 17

1.2.2 dispersed control pleasure 17

1.2.3 Bewusstseinsprozesse in der virtuellen Welt 17

1.3 Spiel als selbstständige Kategorie 18

1.4 Lerneffekte von Computerspielen 19

1.5 Computerspiele und Regeln 19

//2.//Computerspiele// 21

2.1 Definition Computerspiel 21

2.2 Geschichte der Computerspiele 21

2.3. Genre von Computerspielen 24

2.3.1 Action 24

2.3.2 Abenteuer 25

2.3.3 Rollenspiele 25

2.3.4 Strategiespiele 26

2.3.5 Simulationen 26

2.4 Narrativität ,Interaktivität und Offenheit 27

2.5 Architektur in Computerspielen 29

2.6	Dramaturgischer Aufbau eines Spiels	30
2.7	God Mode, Ego-Perspektive, Third-Person	30
2.8	Modding	32
2.9	Cheating	32
2.10	Grammatik des Films	33
2.11	Machinima	35

//3.//Technologie der Computerspiele ////////////////////////////////////// 37

3.1	DirectX / OpenGL	37
3.2	3D Game Engine's	38
3.3	Spielesoftware 3D Game Studio	38
3.4	Aufbau eines 3D Spiels	39
3.5	Umgebungserkennung	40
3.5.1	Umgebungserkennung per trace Anweisung	40
3.5.2	Umgebungserkennung per scan Anweisung	41
3.5.3	Kollision und Bewegung per ent_move Anweisung	41
3.6	Entities	42
3.6.1	Definition Entity	42
3.6.2	Modell Entities	42
3.6.3	Map Entities	43
3.6.4	Sprite Entities	44
3.6.5	Terrain Entities	44
3.7	Programmiersprachen	45
3.7.1	Strukturierte Programmierung	45
3.7.2	Prozedurale Programmierung	45
3.7.3	Modulare Programmierung	45
3.7.4	Objektorientierte Programmierung (OOP)	45
3.8	Begriffe der Programmiersprachen	46
3.8.1	Klassen	46
3.8.2	Abstraktion	46
3.8.3	Kapselung von Daten	46

3.8.4	Vererbung	46
3.8.5	Polymorphie	47
3.9	C# / C-Script	47
3.9.1	Funktionen (abstrakte Klassen)	47
3.9.2	Objekte (Klassen)	47
3.9.3	Scripting / Pseudocode	48
//4.// Anwendungen //////////////////////////////////////		49
4.1	Allgemein	49
4.2	Anwendungsmöglichkeiten oder „Was soll das ganze“	49
4.3	Struktur einer Anwendung	51
4.3.1	Entwicklungsebene	51
4.3.2	Anwendungsebene	51
4.4	Mögliche Anwendungen	53
4.4.1	Transfer eines Raumprogramms	53
4.4.2	Transfer eines Nutzungsschemas	54
4.4.3	Arbeiten mit Künstlicher Intelligenz (KI)	54
//5.// ARCHITECTURE_ENGINE_1.0////////////////////////////////////		57
5.1	Gebrauchsanweisung	57
5.2	Funktionsweise der ARCHITECTURE_ENGINE_1.0	58
5.3	Anwendungsverlauf	59
5.3.1	Spielstart	59
5.3.2	Stiege / Rampe	60
5.3.3	Vererbung	61
5.3.4	Aktivieren	62
5.4	Wahrnehmungsräume	64
5.4.1	1st Person View	64
5.4.2	God View	65
5.4.3	Grundriss View	65



# //1\_ Theoretische Kontexte////////////////////////////////////

## 1.1 Spieldefinitionen

*“Denn, um es endlich auf einmal herauszusagen, der Mensch spielt nur, wo er in voller Bedeutung des Wortes Mensch ist, und er ist nur da ganz Mensch, wo er spielt.”* (Schiller 1795: 131)

Die scheinbar einfache Frage “Was ist ein Spiel?” scheitert schon an grundsätzlichen Merkmalen: Ist *spielen* ein Zustand, eine Tätigkeit, ist es ein physischer, psychischer oder geistiger Prozess?

Der Spielbegriff zeichnet sich nicht gerade durch seine Eindeutigkeit aus, sondern deckt ein großes Feld unterschiedlichster Phänomene ab. Hinzukommen noch sprachlich kulturelle Unterschiede, die den Spielbegriff noch mehr unterteilen. Aus den vielen unterschiedlichen Definitionen werden nun ein paar herausgenommen um zu zeigen wie unerforscht dieser menschliche Trieb bisher ist, denn *„der Begriff, Spiel ist ein Begriff mit verschwommenen Rändern.“* (vgl. Wittgenstein 1951: §71)

Das deutsche Wort „Spiel“ kommt vermutlich aus dem Althochdeutschen Wort “spilen”, was bedeutet: das Fließende, Bewegliche, das Schwebende, sich Bewegende.

Alltagssprachlich wird in unserem Kulturkreis Spiel etwa als Vergnügen, Unterhaltung und damit im Gegensatz zur Arbeit, zu Zwang und Pflichten definiert.

Andererseits sprechen wir auch davon, dass jemand uns etwas “vorspielt”, also nur so tut als ob. Spiel in diesem Sinne ist also das Gegenteil von Wahrheit oder Echtheit. (vgl. Klages-Conti 2001)

Im Englischen unterscheidet man *play* und *game*, wobei *play* eine eher subjektive Einstellung zum Spielmaterial und *game* institutionale Spielaktivitäten bezeichnet, die durch Regeln bestimmt sind, wie sportliche Wettkämpfe.

Johan Huizinga stellt in seinem 1938 erschienenem Buch „Homo Ludens“ die Behauptung auf, dass die menschliche Kultur aus dem Spiel entstanden sei. Alles was der Mensch vollbringt, entsteht aus Spiel. Er zeigt, *„dass Kultur in Form von Spiel entsteht, dass Kultur anfänglich gespielt wird.“*

(Huizinga 1938: 57)

Der Nobelpreisträger für Biochemie Manfred Eigen und Ruthild Winkler gehen in ihrer Definition über die Interpretation von Huizinga hinaus. Sie betrachten das Spiel als Naturphänomen, das in seiner Zweiteilung von Zufall und Notwendigkeit allem Geschehen zugrunde liegt. (vgl. Eigen und Winkler 1990: 11)

Der Zufall als Akt der Mutation, durch den Neues entstehen kann, ohne zunächst notwendigerweise zweckmäßig zu sein. *„Würfel und Spielregel - das sind die Symbole für Zufall und Naturgesetz.“* (Eigen und Winkler 1990: 19) Auch wenn der Ausgangspunkt beider Naturwissenschaftler die Molekulartheorie der Evolution ist, stellen sie den Anspruch, den Rahmen von naturwissenschaftlichen über philosophische und soziologische hin zu ästhetischen Aspekten von Spiel zu spannen.

### 1.1.1 Klassische und moderne Spieltheorien

Die Psychologie und Physiologie bemühen sich das Spielen von Tieren, Kindern und erwachsenen Menschen zu beobachten und zu erklären. Hierbei wird angenommen, dass das Spiel eine notwendige zumindest nützliche Bedeutung hat. Laut Huizinga hat man dem Spiel schon folgende biologische Funktionen zugesprochen:

1. Das sich Entlasten können von einem Überschuss an Lebenskraft.
2. Das lebende Wesen beim Spielen gehorcht einem angeborenen Nachahmungstrieb.
3. Das Spiel befriedigt ein Bedürfnis nach Entspannung oder übt für ernsthafte Tätigkeit.
4. Das Spiel ist eine unschuldige Abregung schädlicher Triebe, oder die Befriedigung für in Wirklichkeit unerfüllbarer Wünsche.

(vgl. Huizinga 1938: 10)

Im folgenden werden nach der Unterteilung von Peter Thiessen in seinem *Arbeitsbuch Spiel*, einige Spieltheorien aufgezählt.

*„Kraftüberschusstheorie:*

*Herbert Spencer* (Anm. d. Verf.: 1820 - 1903, er war Soziologe und Philosoph und mit einer eigenen Evolutionstheorie einer der Gegenspieler von Darwin) deutete das Spiel als Kräfteüberschuss des Spielenden. Die Körperkräfte und geistigen Fähigkeiten, die eine Grundlage für das organische Gleichgewicht darstellen, verlangen nach Betätigung.

*Einübungstheorie:*

*Karl Gross und William Stern* (Anm. d. Verf.: beide Psychologen) sahen das Spiel als eine Vorübung auf das Leben als Erwachsener. Das Kind nimmt die Dinge seiner Umwelt auf und verarbeitet sie im Spiel.

(...)

*Reinigungstheorie:*

*Harvey Carr* deutete das Spiel als eine Möglichkeit des Menschen, seine aggressiven, gesellschaftsfeindlichen Tendenzen auf unschädlich Weise abzureagieren und sich somit davon zu "reinigen".

(...)

*Theorie der Ich –Ausdehnung:*

*Edouard Claparede und später Erik Erikson* sahen in spielerischen Aktivitäten eine Erweiterung des Ichs als Zentrum der heranreifenden Persönlichkeit.“

(Thiessen 2004)

Die Spieltheorie (engl. game theory) im wissenschaftliche Sinn, ist ein Teilgebiet der Mathematik, Operations Research und der Wirtschaftswissenschaften. Sie beschäftigt sich mit der Analyse von Handlungsstrategien in Systemen mit vorgegebenen Regeln ("Spielen"). Dazu untersucht sie vorhergesagtes und tatsächliches Verhalten von Akteuren in Spielen und leitet optimale Strategien her. Durch das Aufstellen von Spielern die nach bestimmten Regeln handeln und untereinander agieren, entstehen Ergebnisse, die nicht abzusehen sind. Neben der abstrakten und theoretischen Behandlung von Strategiespielen hat die Spieltheorie auch Anwendungen in den Wirtschaftswissenschaften. Dabei wird z.B. die Marktwirtschaft als ein 'Spiel' angesehen, in dem die Handelspartner als 'Spieler' einen höchstmöglichen Gewinn zu erwirtschaften suchen. Auch in den Sozialwissenschaften wird die Spieltheorie als „Rational Choice-Modell“ für strategische Konflikte eingesetzt.

Andere Anwendungsgebiete finden sich in der Evolutionsbiologie, Anthropologie, Militärstrategie und Regelungstheorie. (vgl. Wikipedia)

## 1.2 Why do I play and cannot stop?

Eine Antwort auf diese Frage beansprucht mehrere Ansätze.

### 1.2.1 Macht

Zum einen ist es die scheinbare Macht des Spielenden über seine virtuelle Figur (Avatar), die er per Interface, also meist Maus, Tastatur oder Joystick, besitzt. Diese Figur macht was er will, sie gehorcht in ihren Handlungen dem Spieler.

Er ist also mit dem Computersystem interaktiv. Dadurch glaubt er Macht über die Technik des Computers zu haben (vielleicht zum ersten Mal). Aber auch die Macht über die Spielwelt erstreckt sich nur im Rahmen der, vom Programmierer dem Anwender zugedachten, Rahmen.

### 1.2.2 dispersed control pleasure

*“(...) The third and most complex form of pleasure results from the amount of data that have to be managed by the user. I call it, “dispersed control pleasure” as it depends upon an illusion of the user controlling numerous aspects of the system, which are in fact impossible to control .”* (Sitarski (o.A.))

### 1.2.3 Bewusstseinsprozesse in der virtuellen Welt

Computerspiele werden wie Räume betreten und verlassen. Vor jedem Spielstart werden neben der Ladezeit für die Anwendung, Bilder und Musik gezeigt die den Spieler atmosphärisch auf die virtuelle Welt vorbereiten. Ist er einmal in dieser virtuellen Welt muss diese als interessant angesehen werden und ihn von den realen gleichzeitigen Ereignissen ablenken. Kinogleiche Kamerafahrten, Soundeffekte, rüttelnde Joysticks, eigens komponierte Musik, zielen auf die menschlichen Sinne. (Ein normaler Ego-Shooter hat z.B. an die 800 Soundeffekte und 50 verschiedene Musikstücke) Bleibt der Spieler in dieser virtuellen Welt, kommt es zu einer kognitiven und emotionalen Inanspruchnahme, die sich durch Stress oder Flow ausdrücken kann. (vgl. Fritz) (Anm. d. Verf.: Mit Flow (englisch fließen, schweben) wird das Gefühl des völligen Aufgehens in einer lustbetonten Tätigkeit bezeichnet.) Diesen Zustand der Aufmerksamkeit erlebt der Spieler *„als einen Zustand höchster Wachheit und Konzentration. Bewusst wird ihm dies jedoch erst hinterher. Solange die Aufmerksamkeit nach außen auf das Spiel gerichtet ist, erlebt er sich im Strom eines Handelns, das lediglich die Außenperspektive des Bewusstseins kennt.“* (Fritz (o.A.))

Ganz anders als die meisten Anwendungen, lebt oder stirbt ein Computerspiel durch die Effektivität seines User-Interface-Designs. Ein Spiel, das nicht intuitiv zu erfassen und einfach zu benutzen ist, wird ganz einfach nicht populär.

*„Jedes neue Medium muss von neuem bedenken, wie es den Spieler fesseln kann, und die besten Spiele sind deshalb erfolgreich, weil sie neue Strukturen der Interaktion entdecken und neue Genres erfinden. Was für Avantgarde-Film oder –Literatur gilt – der Bruch mit gewohnten Formen der Darstellung, die Entwicklung neuer Modi der Ansprache – ist in der Welt der Computerspiele eine Standardprozedur.“* (Friedman 1999)

### 1.3 Spiel als selbstständige Kategorie

Laut Steffen Walz sind Computerspiele ein bedeutendes, massenhaftes Phänomen der radikalen Moderne, da „sie „Spiel“ als anthropologische Konstante und „Computer“ (im Sinne von Informations- und Kommunikationstechnologien: IKT) synthetisieren.“ (Walz 2003)

Ohne die Entwicklung immer leistungsanspruchsvollerer Computerspiele würde sich die Leistung der Computertechnologie nicht so rasant verbessern und somit auch die IKT. Und umgekehrt gilt ebenso: um so schneller sich die Hardware verbessert um so leistungsanspruchsvoller werden die Computerspiele.

*„Mir scheint, dass mit dieser Synthese zwischen Computer und Spiel nicht nur die Medieninhalte - z.B. Kinofilme, die sich wie Games anfühlen - sondern auch die Kommunikations- und Handelswege, Wissensquellen, Vernetzungsmöglichkeiten, einzelmenschlichen Bindungen, Erfahrungen und Konventionen unserer Gesellschaften nachhaltig verändert und geprägt werden - das Phänomen der Computerspiele also im Begriff ist, sich in ein wissenschaftlichen Paradigma zu verwandeln.“* (Walz 2003)

*„Wenn nämlich IKT - so NewMedia-Theoretiker Lev Manovich - unsere vielfältigen Alltagskulturen sowie unsere Berufs- und Bildungswesen computerisieren, so sollten wir bedenken, dass computerbasierte Games in Form von Hard- und Softwareumgebungen nicht nur unmittelbar als Unterhaltungs-, Edutainment- und Infotainmentangebote wirken können, wenn wir in diesen per se sanktionsfreien Eigenwelten Zeit verbringen, um z.B. zu Lernen oder uns zu entspannen, weil wir so besser mit unseren Alltagsfertigkeiten fertig werden. Nein, das Phänomen der Computerspiele schlägt sich auch mittelbar als computerisierte „Verspielisierung“, als Ludofication nieder - z.B. indem Unternehmen sich die intrinsisch motivierende Wirkung von computerspielartigen Herausforderungen für ihre Arbeitsstrukturen und -prozesse zunutze machen; wenn Skill-spezifische Computerspiele bei Job-Assessments Anwendungen finden, d.h. es zu Transferleistungen zwischen Spielwelt und Nicht-Spielwelt kommt; wenn Spiele als Test- bzw. Simulationsumgebungen dienen; wenn Technologien, die Spiele ermöglichen, als Innovationsmotor in andere Kontexte eingepasst werden; wenn Spielwirklichkeiten die Wirklichkeiten unserer physikalischen Welt erfordernisse um- und erspielen, abgrenzen und entgrenzen.“*

(Walz 2003, verweist auf Manovich 2002 und Kuhn 1967)

Computerspiele sind eine neue Art der ästhetischen und der gesellschaftlichen Auseinandersetzung, weil sie eine Alternative zum narrativen Diskurs anbieten, der bisher das herrschende Paradigma war, wenn es darum ging Wissen und Erfahrung weiterzugeben.

## 1.4 Lerneffekte von Computerspielen

Der Medienwissenschaftler Claus Pias weist darauf hin, dass die Computerspiele, speziell die Simulationsspiele direkt aus der Operations Research, Militärstrategie, und der mathematischen Spieltheorie kämen. Strategiespiele sind laut Pias in erster Linie Verwaltungsakte, Bürokratien, deren Regeln man einhalten muss. Abenteuerspiele bei denen die Spieler fremde Welten erforschen und dabei eine Reihe von Aufgaben lösen müssen, um am Ende ein Ziel (etwa die Befreiung einer Person) zu erreichen, sieht Pias als *„eine Art Interface, das es erlaubt, mit der Datenbank, die das Adventurespiel technisch ist, umzugehen, sie zu durchlaufen, Verbindungen herzustellen usw.“* (Pias 2000)

Die Handlungsfreiheit der Spieler ist also in beiden Fällen sehr relativ. Sie können sich nur innerhalb der festen Spielregeln bewegen. Es geht darum, Ordnung herzustellen, die einzig richtige Lösung zu finden. Und das wirke, so Pias, sicher eher stabilisierend. Wie ein Happy End in einem Film.

Ob es sich nun um ein Strategiespiel handelt, bei dem es um die Errichtung einer virtuellen Stadt geht, oder um einen sogenannten Ego-Shooter, dessen Ziel die Beseitigung von schiesswütigen Gegnern ist: *„Beseitigung und die Errichtung sind zwei Aspekte derselben Frage nach Effektivität oder ökonomischer Rationalität“.* (Pias 2000)

Amerikanische Psychologen vermuten, dass der ständige Anstieg des Intelligenzquotienten unter anderem an dem Umgang mit den neuen Medien wie Computerspiele liegt. Feinmotorik, Reaktionsvermögen, Konzentration, unlogische Problemlösungen, in den Spielen fordern nämlich genau die Fähigkeiten die bei dem Intelligenztest abverlangt werden. (vgl. Pias 2000)

## 1.5 Computerspiele und Regeln

*„Jedes Spiel hat seine eigene Regeln. Sie bestimmen was in der Spielwelt gelten soll und was nicht. Diese Regeln sind bindend und dulden keinen Zweifel. Man kann sie nur außerhalb des Spiels ändern. (...) Paul Valery hat es einmal beiläufig gesagt, und es ist ein Gedanke von ungemeiner Tragweite. Gegenüber den Regeln eines Spiels ist kein Skeptizismus möglich. (...) Sobald die Regeln übertreten werden, stürzt die Spielwelt zusammen.“* (Huizinga 1938: 20)

Ein wesentlicher Unterschied zwischen Spielen und Computerspielen besteht darin, dass bei klassischen Spielen die Regeln erst gelernt und verstanden werden müssen, hingegen kommt in Computerspielen erst das Spiel. Man beginnt direkt im Spiel und erst durch sukzessives Handeln und Erleben lernt man die Regeln kennen. Dies ändert also auch grundlegend die Herangehensweise der Spieler. Die Unkenntnis, die in eigentlichen Spielen ein Nachteil ist, wird hier zu einem Vorteil. Der Spieler wird zur Aktivität gezwungen, muss Lösungswege finden bzw. erfinden. Hier kennzeichnet sich schon ab, dass eine Computerspiel ein Raum mit Regeln ist. Der Spieler befindet sich in einem drei dimensionalen Ereignisraum, dessen Regeln er aber erst erfahren muss.

## //2\_ COMPUTERSPIELE////////////////////////////////////

### 2.1 Definition Computerspiel:

„Ein Computerspiel ist ein Computerprogramm, das es den Benutzern ermöglicht, ein durch im Programm implementierte Regeln formalisiertes Spiel zu spielen. Die Benutzer interagieren mit dem Computerspiel über Eingabegeräte, während das Computerspiel im Regelfall einen Bildschirm (daher Videospiele) für den primären Feedback in Form von Bildern oder Texten nutzt. Als Software ist ein Computerspiel eine (interaktive) elektronische Publikation.“

(netlexikon )

### 2.2 Geschichte der Computerspiele

Der Vollständigkeit wegen wird ein kurzer auszugsartiger Überblick über die Geschichte der Computerspiele gegeben.

Das erste richtige Computerspiel war nicht wie oft beschrieben *Spacewar* von Steve Russel und Dan Edwards, sondern das Spiel „*Tennis for two*“ von William Higinbotham aus dem Jahr 1958. Higinbotham war Leiter der Information Division am „Brookhaven National Laboratory“, welches mit neuen Geräten arbeitete, die für einzelne Institute zu teuer waren. Higinbotham arbeitete mit Analogcomputern und entwickelte „*Tennis for two*“ zum Tag der offenen Tür, quasi als Demoanwendung zum Präsentieren seines Arbeitsgeräts. (siehe Abb. 1) Hier konnten zwei Benutzer auf einem 12 cm großen Oszilloskop gegeneinander antreten. (vgl Lischka 2002: 19)

1962 kam „*spacewar*“, entwickelt, wie oben erwähnt, von zwei MIT Mitarbeitern.

(siehe Abb. 2) Anfang der Siebziger Jahre kamen die ersten Münz-Videospiele heraus, wie z.B. 1972 „*Pong*“ von der Firma Atari. (siehe Abb. 3)

Danach folgten die ersten Heimkonsolen und erste textbasierte adventure games wie „*Adventure*“ von William Crowther (1974).



Abb. 1: *Tennis for two* 1958

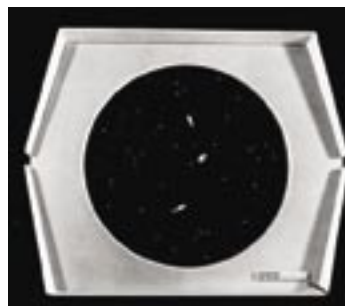


Abb. 2: *Spacewar* 1962



Abb. 3: *Pong* 1972

1979 erschien das erste dreidimensionale Computerspiel „*Star Raider*“ ( siehe Abb. 5) von Dog Neubauer (Atari) welches dem Spieler bereits zwei Perspektiven anbietet. Dies zeigt schon ganz früh, dass die dreidimensionale Perspektive des Spielers nicht ausreicht, um die komplexe Umwelt wahrnehmen zu können.

1980 erscheint „*Pac Man*“, herausgegeben von Nintendo. 1983 kommt der „*Flight Simulator*“ von Microsoft. 1989 bringt Nintendo dem Game Boy mit dem Spiel „*Tetris*“ und 1990 das Spiel „*Super Mario Bros(thers)*“. Ebenfalls 1989 erscheint Will Wrights „*SimCity*“.

„*Doom*“ von John Romero und John Carmack (ID Software) erscheint 1993 und ist der erste voll texturierte 1st-person-shooter. (siehe Abb. 4)

Als Nachfolgespiel erscheint 1996 „*Quake*“. Darauffolgend erscheinen unter anderen vermehrt Fortsetzungen von amerikanischen Filmproduktionen wie „*Enter the Matrix*“ (2003) oder „*Spider Man 2*“ (2004).

Sony brachte 2004 das „*EyeToy*“ für die Playstation2 heraus, bei dem per Kamera der Spieler mit seinen Bewegungen in das Spiel integriert wird. (siehe Abb. 7)

Atari bringt Ende 2004 eine Konsole heraus, die den Spieler per Handschuh am Spiel teilnehmen lässt.



Abb. 4:  
*Doom* 1993

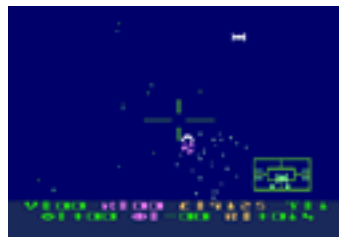


Abb. 5:  
*Star Raider* 1979

(siehe Abb. 6)

Laut Ernest Adams in einem Interview zur Zukunft der Computerspiele, liegt diese in der Schnittstelle zwischen Mensch und Computer, den sogenannten Interfaces. So gibt es schon Forschungsrichtungen die sich mit *augmented reality* beschäftigen, also eine Erweiterung unserer Realität, durch Brillen, die eine virtuelle Umgebung mit der realen überlagern.

Das Institut für Wearable Computer Lab an der School of Computer and Information Science, University of South Australia, entwickelte „*ARQuake*“ ein Spiel, aufbauend auf „*Quake*“, bei dem die Spieler per Datenbrille virtuelle Monster auf ihrem Campus bekämpfen können. (siehe Abb. 8-10)



Abb. 6: Atari Konsole 2004

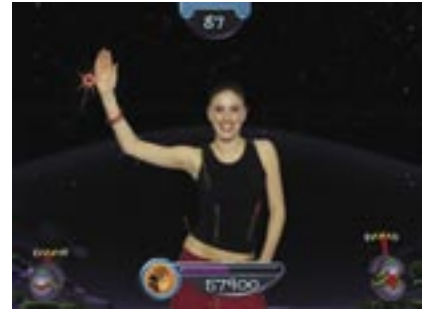


Abb. 7: Sony Eye Toy 2004

„ARQuake is an Augmented Reality (AR) version of the popular Quake game. Augmented Reality is the overlaying of computer generated information onto the real world. Unlike a Virtual Reality (VR) world, where the viewer sees a completely synthetic (virtual) environment, users immersed in an AR environment can still see the real world. A classic example of an AR device is the “Heads Up Display” found in Fighter Aircraft. In this case, information about the aeroplanes position and speed or about the enemy aircraft is displayed over the real image of the world outside.” (Bruce 2002)



Abb. 8: ARQuake 2002 Headset



Abb. 9: ARQuake 2002, Blick



Abb. 10: ARQuake 2002

## 2.3 Genre von Computerspielen

Folgende Genres sollen kurz beschrieben werden: Actionspiele, Abenteuerspiele, Rollenspiele, Strategiespiele und Simulationen.

### 2.3.1 Actionspiele

Als Urform des arcade game gilt gemeinhin das Spiel „*Spacewar*“, ein Spiel, das Steve Russell vom Massachusetts Institute of Technology (MIT) 1961 für den Großcomputer PDP-1 entwickelte. Die diesem Bereich zugehörigen Spiele zeichnen durch eine iterative Handlung mit steigendem Schwierigkeitsgrad, eine eigene Zeitlichkeit, die der Spieler nicht beeinflussen kann, und vorwiegend gewalttätige Konfliktlösungsstrategien aus. Zwar verfügen die meisten Actionspiele über eine Rahmenhandlung, doch diese hat im Spiel meist kaum eine Bedeutung. Wichtiger ist die atmosphärische Dichte des Spiels. Herauszuheben sind insbesondere die sogenannten jump 'n' runs, die sich durch eine ähnliche Grundstruktur wie die „reinen“ Actionspiele auszeichnen, aber auf „echte“ Gewalt weitgehend verzichten. Ein Beispiel für dieses Subgenre ist die „*Super Mario*“ World-Serie, in der die Gegner dadurch überwunden werden, dass die Spielfigur auf sie springt und sie plattdrückt.

Weiter zu erwähnen sind die *first-person shooters* (fps) wie „*Doom*“, „*Quake*“ oder „*Counter Strike*“. Diese werden heute meist in Gruppen über das Internet gespielt. Ihr Aufbau ist dem Medium des Actionfilms entlehnt, kurze Rahmengeschichte, viel Action. Dazu bedienen sich diese Spiele ästhetischer Verfahrensweisen aus dem Hollywood-Film: In „*Max Payne*“ gibt es eine sogenannte Bullettime Frequenz die stark an die Zeitlupenaufnahme aus „*Matrix*“ erinnert. (vgl.: Abb. 11 und 12) Die fps werden auch oft als *corridor games* bezeichnet.



Abb. 11:  
Computerspiel *Max Payne* 2003



Abb. 12:  
Kinofilm *Matrix* 1999

### 2.3.2 Abenteuerspiele (adventure games)

In dem gemeinsam verfassten Aufsatz „*Nintendo® and New World Travel Writing*“ (1995) vergleichen Fuller und Jenkins Computerspiele mit den Reiseberichten Sir Walter Raleighs und Captain John Smiths, in denen sich das Motiv der entführten Prinzessin wiederholt, und stellen fest:

*“[T]he movement in space that the rescue plot seems to motivate is itself the point, the topic, and the goal and [...] this shift in emphasis from narrativity to geography produces features that make Nintendo® and New World narratives in some ways strikingly similar to each other and different from many other kinds of texts”.* (Fuller und Jenkins 1995)

Für das Abenteuerspiel lässt sich eine Beziehung zwischen seiner narrativen Struktur und seiner typischen Form der Zeitlichkeit herstellen. Im Abenteuerspiel steht die Zeit meist still, solange der Spieler sie nicht durch seine Interaktion vorantreibt. Dabei gliedert oft tatsächlich das Vorwärtskommen im Raum die Handlung des Spiels.

### 2.3.3 Rollenspiele

Heute auch meist MMORPGs (Massive Multiplayer Online Role Play Game) genannt, da sie fast nur noch im Internet gespielt werden. Gegenüber Abenteuerspielen ist die Handlung von Rollenspielen lose organisiert. Statt eines linearen Handlungsstrangs, dem die Geschichte folgt, gibt es oft mehrere Wege, zum Ziel zu gelangen. In „*Ultima Online*“ ist der Gedanke eines Spielziels ohnehin weitgehend in den Hintergrund gerückt. Zwar können die Spieler ihre Charaktere auf verschiedene Missionen (sogenannte Questen) schicken, doch die Welt von „*UO*“ fungiert im Wesentlichen wie eine Parallelwelt zur wirklichen Welt, so dass es im Ermessen des Spielers liegt, welche Ziele er oder sie sich setzt. (siehe Abb.13)



Abb. 13:  
*Ultima Online*

### 2.3.4 Strategiespiele

Rollenspiele rangieren hinsichtlich der Offenheit der Handlung über den Action- und Abenteuerspielen. Auf einer noch höheren Stufe stehen diesbezüglich Strategiespiele. Prototypische Vertreter dieses Genres sind Spiele, bei denen der Spieler für die Entwicklung eines Volkes verantwortlich ist, wie „Civilization“ oder „Age of Empires“. Wie bei „Civilization“ handelt es sich bei „Die Siedler“ um die Adaption eines Brettspiels. Dies macht auf den prinzipiellen Unterschied zwischen Strategiespielen und Simulationen aufmerksam. Bei Strategiespielen übernimmt der Computer lediglich die Rolle der anderen Spieler, die vor dem Hintergrund einer virtuellen Welt agieren. Simulationen haben hingegen ein „Eigenleben“, das es dem Spieler ermöglicht, auf ein Eingreifen zu verzichten und einfach nur die Entwicklung zu beobachten.

Strategiespiele verfügen wie Action- und Rollenspiele über eine eigene Zeitlichkeit. Vertreter dieses Genres zeichnen sich oft dadurch aus, dass sie wie Brettspiele auf der abstrakten Einheit einer „Runde“ basieren. Innerhalb einer Runde kann der Spieler meist nur eine begrenzte Zahl von Handlungen ausführen.

### 2.3.5 Simulationen

Simulationen zeichnen sich, wie bereits erwähnt, durch ein „Eigenleben“ aus. Als das erste Simulationsspiel überhaupt kann wohl John Conways „Game of Life“ (1970) gelten. Das Spiel besteht aus einem Raster mit „zellulären Automaten“, bei dem der Spieler die Ausgangskonfiguration bestimmen kann. Die zellulären Automaten sind durch einfache Regeln definiert. Aus diesen Regeln entstehen sehr rasch komplexe Strukturen.

„Game of Life“ weist bereits alle Charakteristika auf, die sich auch späteren Simulationsspielen beobachten lassen. Beispielsweise kann der Spieler jederzeit in das Geschehen eingreifen und neue Zellen „erschaffen“ bzw. „töten“. Dies ist aber für die Entwicklung des Spiels nicht notwendig.

Prototypische Vertreter dieses Genres sind die *god games*, bei dem der Spieler in die Rolle eines »Gottes« oder gottgleichen Herrschers über eine virtuelle Welt schlüpft. (siehe Abb. 14 und 15)



Abb. 14:  
Sim City 1989



Abb. 15:  
Sim City 2003

Als das erste kommerziell erfolgreiche Simulationsspiel kann wohl Will Wrights „*SimCity*“ gelten, in dem der Spieler die Rolle des Herrschers über eine Stadt übernimmt, und die Aufgabe hat, dieser Stadt zu Wohlstand und Ansehen zu verhelfen. Der Spieler übernimmt dabei nicht nur die Rolle des Stadtplaners, sondern auch die des Bürgermeisters, des Polizeichefs und des Vorsitzenden der Verwaltung. Im Vergleich zu anderen Computerspielgenres fällt auf, dass Simulationen bei weitem den höchsten Grad an Offenheit aufweisen. Zwar sind die Interaktionsmöglichkeiten des Spielers oft durch einen das Spiel verzögernden Mangel an Geld begrenzt, doch prinzipiell kann der Spieler meist tun, was ihm gerade in den Sinn kommt. Damit einher geht ein relativ niedriges Narrativitätsniveau, da sich der Spieler seine Ziele weitgehend selbst setzen kann. Wright bezeichnet „*SimCity*“ daher auch nicht als Computerspiel, sondern als *software toy*:

*„Wright offers a ball as an illuminating comparison: It offers many interesting behaviors, which you may explore. You can bounce it, twirl it, throw it, dribble it. And, if you wish, you may use it in a game: soccer, or basketball, or whatever. But the game is not intrinsic in the toy; it is a set of player-defined objectives overlaid on the toy.*

*Just so Sim City. Like many computer games, it creates a world which the player may manipulate, but unlike a real game, it provides no objective. Oh, you may choose one: to see if you can build a city without slums, perhaps. But Sim City itself has no victory conditions, no goals; it is a software toy.”*

(Costikyan 1994)

## 2.4 Narrativität, Interaktivität und Offenheit

*“Interactivity is almost the opposite of narrative; narrative flows under the direction of the author, while interactivity depends on the player for motive power”* (Adams 1999)

Oft wird versucht Computerspiele als Interaktives Medium zu bezeichnen. Dies kann man jedoch nur zum Teil. Im Vergleich zu Büchern oder Filmen als narrative Medien, gibt es bei Computerspielen verschiedenen Bereiche die alle zwischen Narrativität, Interaktivität und Offenen System anzusiedeln sind. (siehe Abb. 16)

*“There is a direct, immediate conflict between the demands of a story and the demands of a game. Divergence from a story’s path is likely to make for a less satisfying story; restricting a player’s freedom of action is likely to make for a less satisfying game.”* (Costikyan 2000)

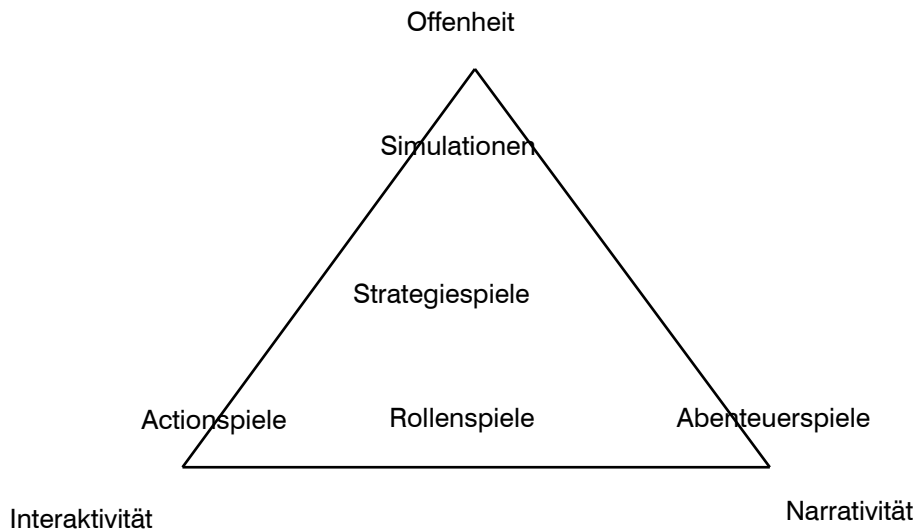


Abb 16:  
Einteilung der Spiele nach Interaktivität, Narrativität und Offenheit

Das Kriterium der Offenheit unterscheidet sich von dem der Interaktivität dadurch, dass es sich auf das Spektrum der zur Verfügung stehenden Möglichkeiten bezieht, während das Niveau der Interaktivität die Häufigkeit der Interaktionen angibt.

Spiele, die sich durch eine hohe Interaktionsfrequenz auszeichnen, lassen dem Spieler meist nur die Wahl zwischen zwei oder drei Alternativen, die sich nicht wesentlich auf die fiktionale Welt auswirken. Ein Beispiel dafür sind arcade games wie „*Space Invaders*“, in dem sich die Interaktion des Spielers auf die die Bewegung nach links oder rechts sowie schießen oder nicht schießen beschränken.

Spiele mit einem hohen Grad an Offenheit stellen den Spieler hingegen vor die Wahl zwischen einer Fülle an verschiedenen Möglichkeiten, die das Potenzial haben, die Welt des Spiels nachhaltig zu verändern. Ein Beispiel für diese Form der Interaktivität sind komplexe Strategiespiele wie „*Civilization*“, bei denen der Spieler in die Rolle eines gottgleichen Herrschers über die jeweilige Welt schlüpft.

## 2.5 Architektur in Computerspielen

Die Architektur in Computerspielen zu analysieren würde erst einmal bedeuten sie allumfassend darzustellen. Diese Arbeit wird aber eher versuchen grundlegende Muster und Regeln zu finden.

So ist die Architektur in erster Linie einmal abhängig vom thematischen Inhalt des Spiels. Gerade ihr textueller Bezug wird meistens realistischen Vorbildern nachgebildet, oder besser gesagt deren Klischees. Wie weiter oben schon erwähnt sind Spiele Räume mit Regeln. Der Spieler erlernt die Spielregeln im virtuellen Raum des Spiels. So organisieren die Spiele ihre Ereignisse in einer räumlich begehbaren Struktur. Die Ereignisse werden räumlich organisiert. Dies hat sich auch durch den Sprung in dreidimensionale Spielwelten nicht groß geändert. Obwohl der Spieler hier theoretisch in alle Richtungen gehen kann, ist sein Weg durch seine nächste Aufgabe vorherbestimmt. Auch die Architektur mit ihrer Umgebung und Licht oder Farbgestaltung ist so organisiert, dass der Spieler gelenkt wird. Hier bleibt also die narrative Erzählstruktur vieler Spiele deutlich sichtbar.

Die Räume überlappen sich in Spielen nie. Sie sind da, um Ordnung zu schaffen, es ist nur selten nötig lange an einem Ort zu verweilen, die nächste Aufgabe wartet an einem neuen Ort. Christiane Funken und Martina Löw gehen führen den Begriff des Containerraums ein: „(...) *die Räume in Computerspielen sind – entgegen dem Internetmythos – als Container für das Spielgeschehen konzipiert.*“ (...) *„Die Räume sind demnach in den Computerspielen nicht Gegenstand der Auseinandersetzung, sondern strukturierende Kisten.“* (Funken und Löw 2001: 76)

Die Räume sind also formales Strukturprinzip eines mathematischen Regelsystems. Der Ereignisraum steuert die potentielle Verhaltensweisen des Spielers, durch seine Regeln determiniert er erwünschtes und unerwünschtes Verhalten, sowohl räumlich als auch handlungsorientiert. (siehe Abb. 18 und 19)

Selbst offene Landschaften, die scheinbar frei begehbar sind, haben meist nur einen gehbaren Weg. (siehe Abb. 17) Schlägt der Spieler einen eigenen Weg ein, kommt er bald an ein Hindernis. Der Spielpfad muss begangen werden. Es gibt ein paar wenige Spiele die von diesem Pfadsystem abweichen und der Spieler die Möglichkeit hat sich frei z.B. durch eine Stadt zu gehen, doch auch hier bleibt der Haupthandlungsstrang aussen vor und man spielt auf Nebenschauplätzen irgendwelche kleinen Aufgaben durch. Also sind selbst die Landschaftsumgebungen nichts anders als große Container.

Die Containerräume haben aber auch noch einen anderen häufig nicht genannten Grund: die Technik. Die Technologie der Computerspiele aufbauend auf der Game Engine (siehe Kapitel „Technologie der



Abb. 17:  
Pfade in einer offenen Landschaft

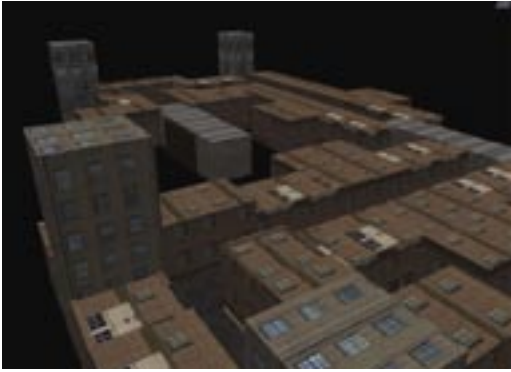


Abb. 18:  
Containerräume in *Medal of Honor 2002*



Abb. 19:  
Containerräume in *Splinter Cell 2004*

Computerspiele“ dieser Arbeit) erlauben als Grundeinheit bisher fast nur die der Box. Dies hat mit der Geschwindigkeit zu tun, da die Berechnung einer Box von den aktuellen Game Engine's am schnellsten passiert.

## 2.6 Dramaturgischer Aufbau eines Spiels

Viele Computerspiele haben den gleichen dramaturgischen Aufbau. Zu Beginn, verläuft die Handlung langsam, der Spieler braucht Zeit sich zurechtzufinden, sich zu orientieren. Nach kurzer Zeit verdichten sich die Aktivitätsmomente, die Aufgaben kommen in kürzerer räumlicher und zeitlicher Abfolge und vor allem werden die Handlungsoptionen des Spielers vielfältiger. Oft gibt es eine den Spieler begleitende Variable, die für Lebensenergie des Spielers oder für den Kontostand steht und nicht gegen Null laufen darf. Um dies zu erreichen, muss der Spieler bestimmte Aufgaben erfüllen. Umso weiter er im Spiel kommt umso schwerer werden zudem die zu erfüllenden Aufgaben.

## 2.7 God-View, First-Person-View, Third-Person-View

Bestimmte grafische Perspektiven im Spiel erlauben es den Spielenden die Kontrolle des Spielers zu erhöhen, während andere es erleichtern, ihm die Kontrolle zu entziehen. Spiele mit isometrischer Perspektive übertragen dem Spieler beispielsweise ein sehr hohes Maß an Kontrolle und damit ein hohes Maß an raumzeitlicher Immersion. Bezeichnenderweise ist dies die Perspektive der god games, Aufbauspielen also, bei denen der Spieler die Rolle des Schöpfers (*Black & White*) oder zumindest des Bürgermeisters übernimmt (*SimCity*). Aber auch Strategiespiele, bei denen der Spieler in die Rolle eines Herrschers oder Heerführers schlüpft, bedienen sich dieser Perspektive. Es entsteht ein Wechselspiel zwischen Kontrolle und Kontrollverlust. (siehe Abb. 20)



Abb. 20:  
God-View in *Black and White*

Ein etwas geringeres Maß an Kontrolle erlauben Spiele, die sich der third-person-view bedienen, wie viele Abenteuerspiele (*Tomb Raider* oder *Max Payne*) und Rollenspiele (*EverQuest*). Hier herrscht eine ausgewogene Balance zwischen raumzeitlicher und emotionaler Immersion, die im Wechsel von interaktiven und nicht-interaktiven Sequenzen des Action-Adventures ihren Ausdruck findet. (siehe Abb. 21)



Abb. 21:  
3rd-Person-View in *Tomb Raider*

Weit verbreitet ist auch der *first-person-view*, der sich einer Perspektive bedient, welche den Kontrollverlust befördert, obwohl dieser Blick (view) ja eigentlich dem des Menschen am nächsten kommt. Dieser menschliche Blick langt jedoch oft nicht aus, so dass zusätzlich Übersichtskarten im Spiel gezeigt werden, die dem Spieler eine bessere Kontrolle geben. (siehe Abb. 22)



Abb. 22:  
1st-Person-View in *The Twin Snakes*

## 2.8 Modding

Einer der wichtigsten Aspekte des Computerspiels ist der des Moddings. Modding bezeichnet die Möglichkeit zu einem Spiel neue Maps, also virtuelle Spiellandschaften, auch Levels genannt, selbst zu bauen. Es gibt Hunderte solcher von Spielern gebauten Maps im Internet. Eines der zur Zeit bekanntesten Ego-Shooter Spiele „Half-Life“ ist eine solche Modifikation (=Mod). Die Hersteller bieten eine Software (Editor) an mit der man diese Mods erstellen kann, oder sie geben Teile ihres Quellcodes frei (Open Source), damit Spieler diese Editoren selbst programmieren können. Diese der Spielergemeinschaft gegebenen Freiheit in Kombination mit dem Verteilungsnetz Internet, ist einer der Erfolgsgründe der Computerspiele.

*“I argue that modding is a form of player-creative game play that cannot be adequately explained by theories of cultural production, distribution and consumption in which consumption is always already in a position of weakness against production and distribution, which are always already in positions of power. A theory of computer gaming culture must account for player-creative game play as encompassing a range of contradictory creative practices, some of which are simultaneously empowering and disempowering, resistant and submissive. As a contribution towards such a theory, I examine game modding and its communities to demonstrate the emergence of conflicting models of production, distribution and consumption. For instance, by providing mod tools to their customers, game developers increase sales, help to generate a committed and long-term community following, and potentially train newly employable game developers. At the same time, however, they also provide the means for mod creators to challenge the very models of production and distribution—including the rules of intellectual property—that maintain relations between consumers and producers in which cultural works are commodified for capital profit. In other words, the mod community/industry is a space of conflicting modes of cultural production that, if theorized within a framework that accounts for its contradictions, can also help to explain other forms of digitally-mediated participatory culture.” (Mactavish)*

## 2.9 Cheating

Fast alle Computerspiele haben die Möglichkeit sich selbst zu manipulieren. Das heißt der Spieler kann durch spezielle Tastenkombinationen bestimmte Aktionen auslösen, wie z.B. sich eine Rüstung anlegen, die ihn unverwundbar macht. Diese *Cheats* werden meist im Internet verbreitet, sind aber von Anfang an von den Entwicklern mit eingedacht worden.

*“Furthermore, cheat codes carry cultural significance beyond the games they stem from. In an article on slashdot.com entitled “Up, Up, Down, Down” Jon Katz quotes a cheat combination for several games on the Nintendo Entertainment System (NES). He goes on to say that if you “[r]ecite this combination to millions of younger Americans, [...] it’s like a secret handshake.” Therefore, cheat codes can be regarded as a sort of symbolic currency within gaming cultures - the communities that will inevitably form around specific games, game genres, or game systems. These cultures, however, are closely connected to their cultural environment, and thus games come to play a role in culture in general. Similar to concepts such as “copy and paste”, that have acquired significance outside computer culture, cheats might also become part of our everyday semiotic repertoire. “ (Newman)*

## 2.10 Grammatik des Films

Das mediale Umfeld des Computerspiels ist die Nähe zum Film, aber auch zu Videoclips und Comics. So erinnert gerade die atmosphärische Gestaltung vieler Computerspiele an die Ästhetik von Spielfilmen, oder bei japanischen Computerspielen an Comics (Manga). Die realistische Oberfläche, Kameraführungen, Schnitte und Lichtführung kommen bei westlichen Computerspielen aus dem Medium Film. Wichtig ist beiden Medien das persönliche Eintauchen des Betrachters (Immersion) in die Welt des Mediums.

Auch umgekehrt werden Strukturen von Computerspielen in Filmen übernommen, man denke an „Tron“ (Steven Lisberger 1982), „Lola Rennt“ (Tom Tyker 1998) oder „Cube“ (Vincenzo Natali 1997). Beide Industriezweige vermischen sich und bringen verwandte Produkte auf den Markt. So wie der gerade erschienene Kinofilm „Spider Man 2“ und das parallel erschienene Computerspiel. Jedoch gelten bei eingefleischten Spielern diese Hybridprodukte meist als schlecht, da der Spielspaß oft unter der zwanghaften Nähe zum Film leidet. Dies liegt wahrscheinlich an der Tatsache, dass das lineare narrative Element des Films, mit Anfang und Ende und knappen 120 Minuten dazwischen schwer auf ein interaktives offenes Medium, das über Wochen gespielt wird, übertragen werden kann. So sind beide Medien grammatikalisch verwandt, jedoch strukturell verschieden.

Mark J.P. Wolf betrachtet in seinem Buch „*The Medium of the Video Game*“ (2001) zwei Bereiche der Überschneidung beider Medien genauer. Er benennt sie als *first-person narrative* und *spatial orientation*. Mit der Entwicklung der Computergrafik wurde es möglich, dreidimensionale Räume in Echtzeit zu berechnen, bestimmte Lichtstimmungen zu erzeugen, verschiedene Perspektiven einzusetzen und Schnitte zur Erzeugung narrativer Zusammenhänge zu verwenden (*continuity editing*).

Gleichzeitig werden die Charaktere von Videospielen immer wichtiger, die visuellen Effekte immer spektakulärer und die „Action“ des Spiels wird in einen narrativen Kontext gestellt: „*By the 1990s, video games had title screens, end credits, cutting between different sequences, multiple points of view, multiple locations, and increasingly detailed storylines.*“

(Wolf 2001: 32)

Neben formalen Gesichtspunkten sind es also vor allem räumliche Strukturen, die das Videospiel aus dem Film übernimmt. Dazu merkt Wolf an, dass die Art und Weise wie *on-screen space* und *off-screen space* in den Medien Video, Film und Computerspiel genutzt wird, recht ähnlich ist. Allerdings hat der Raum durch die Elemente der Navigierbarkeit und Interaktivität im Computerspiel eine wichtigere Bedeutung als im Film: der Raum wird hier nicht nur wahrgenommen, sondern erfahren. Nicht nur ein quantitativer, sondern ein qualitativer Unterschied besteht darin, dass der *off-screen space* im Computerspiel kein Gegenstück in der materiellen Welt hat. Denn während Filme – zumindest traditionell – *on location* oder im Studio gedreht werden, entstehen Videospiele am Computer: „*In a video game, not only the representation of space, but even its implication, depend on being programmed and actively created.*“ (Wolf 2001: 52)

Dies gibt dem Medium des Computerspiels weit gehende Freiheiten hinsichtlich der Formung und Strukturierung des *off-screen space*. Ein etwas triviales Beispiel dafür ist die Tatsache, dass „Pac-Man“, wenn er den Bildschirm auf der einen Seite „verlässt“, auf der andern Seite wieder „hereinkommt“. Dieses Phänomen ist geübten Spielern völlig vertraut, während die gleiche Szene im Film – je nach Genre – für Erstaunen oder Irritation sorgen würde.

Auch bei der Entwicklung der räumlichen Darstellung kommt der Technik eine maßgebliche Rolle zu: „*[G]raphical advances along with greater storage capacity, allowed the diegetic worlds of video games to enlarge beyond a few screens, requiring methods of representation which could link different spaces*

together, and link on-screen and off-screen spaces.“ (ebenda) Während dieser Entwicklung scheint der »visuelle Realismus« der audiovisuellen Medien immer als ein Maßstab gedient zu haben. Wolf macht darauf aufmerksam, dass heutzutage kaum noch abstrakte Spiele produziert werden, während dies noch in den 80er Jahren gang und gäbe war.



Abb. 23:  
Screenshot aus *Tron* (Steven Lisberger 1982)

Während die Darstellung des on-screen space bei 3D-Spielen von stark begrenzten Darstellungen mit voreingestellten Perspektiven bis hin zu völlig frei zugänglichen VR-Umgebungen reicht, kommt vor allem in first-person shooters dem off-screen space eine wichtige Bedeutung zu: „*The first-person perspective increased the importance of off-screen space because it positioned the player within the space, subjectively, as opposed to the third-person objective view in earlier games. This shift also allowed a more mature usage of off-screen space and one closer to that of cinematic off-screen space.*“ (Wolf 2001: 66)

Wolf kommt schließlich zu dem Ergebnis, dass Computerspiele mit dreidimensionaler Grafik zwar weiterhin eine Tendenz zu immer realistischerer Grafik zeigen. Doch in Zukunft könnten sich die Spiel-designer darauf besinnen, dass die Stärke des Computers darin liegt, räumliche Konfigurationen zu berechnen und anzuzeigen, die in anderen Medien nicht dargestellt werden können.

## 2.11 Machinima

Machinima ist ein Kunstwort aus Machine, Cinema und Animation und steht für eine neue Art von Filmen die mit Hilfe von Game Engines hergestellt werden. Der Zugang zu den Game-Engines erfolgt entweder durch Knacken des Codes oder durch legale Nutzung - indem Hersteller Teile der Engines für Machinimas zur Verfügung stellen. Machinima Regisseure verwenden verschiedenen Technologien und fügen diese dann zusammen. Meistens werden Techniken aus PC-Spielen und Digitalvideo verwendet. Die Filme selbst entstehen nicht durch Bild-für-Bild Animation sondern werden als "virtuelle Realität" generiert. In diesen "virtuellen" Räumen werden dann die Filmszenen inszeniert - ähnlich dem Video-Dreh in der Realität.

Wesentliche Grundlage für die Machinima Technik ist sog. Echtzeit 3D Rendering - also die Darstellung von Computer-Modellen in Bilder in Echtzeit - ohne zeitaufwendiges Zwischenberechnen. Die Bilder werden "on the fly" dargestellt: what you film is what you get. Unter Mithilfe mehrerer Gamer kann man dann gleichzeitig mehrere Spielfiguren steuern und diese Sequenzen aufzeichnen. In der Nachbearbeitung kommen dann noch Dialoge sowie Musik und Sound hinzu.

Die Charaktere, das Set, die Animation etc. des Machinima werden mit herkömmlichen Tools erstellt. Danach verwenden Machinima Regisseure eine Game-Engine und fügen ihre Charaktere ein, inszenieren die Story und "filmen" dies ab. Die Generierung von Spieleszenen ist etwas komplizierter. Grundlage hierfür sind z. B. auch wieder existierende Szenen - z. B. Explosion eines Raumschiffes. Dieser Code wird verwendet, um den eigenen Film zu inszenieren, egal ob es sich um das Anzünden einer Zigarette, die Unterhaltung mit einem Partner oder das Öffnen einer Tür handelt.

Heute werden unterschiedliche Game Engines verwendet, wie z. B. die Quake-Serie, die Heavy Metal: FAKK2 engine und die Serious Sam engine. Inzwischen gibt es eine rege Szene und sogar internationale Wettbewerbe. (siehe [www.machinima.com](http://www.machinima.com))

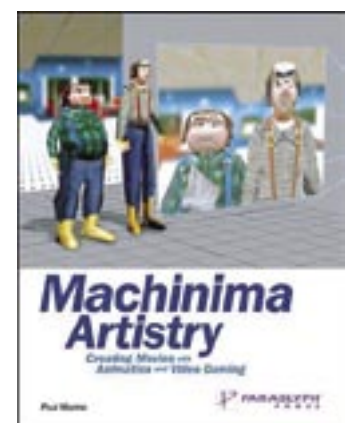


Abb. 24:  
Buch über die Technik von Machinima

## //3\_ Technologie der Computerspiele//////////

Computerspiele sind Bildschirmspiele, die eine mit dreidimensionalen Objekten gestaltete virtuelle Realität darstellen und dem Spieler die Möglichkeit geben in diesen virtuellen Räumen per Interface (Tastatur, Maus, Joystick) zu agieren.

Mit Hilfe von Programmiersprachen, heute meist C++, entstehen Basisbibliotheken von Anwendungen, sogenannte Engine's. Diese greifen auf die Hardwarekomponenten des Computers, wie CPU, Grafikspeicher, Hauptspeicher, Tastatur oder Maus zu und ermöglichen es so auf dem Bildschirm eine Momentaufnahme der grafischen Ausgabe des Computerspiels wiederzugeben. Diesen Zugriff übernimmt auf windowsbasierten Systemen fast immer DirectX oder – weniger verbreitet, weil nicht von Microsoft, aber besseres System (unabhängiger, stabiler) – OpenGL. Beides sind geräteunabhängige Schnittstellen um Multimediaanwendungen effizient nutzen zu können.

### 3.1 DirectX

DirectX ist in folgenden Komponenten aufgeteilt: (aktuelle Version: DirectX 9.0b)

DirectDraw:

Erlaubt direkten Zugriff auf die Grafikkarte.

Direct3D:

Sammlung von Funktionen mit denen Geometrietransformationen (Bewegung, Rotierung oder Skalierung) sowie Beleuchtung und Texturierung ausgeführt werden können. Bis Version 7.0 wurden alle Geometrieoperationen und Lichtberechnungen (T&L=Transform and Lighting) von der CPU (central processor unit) ausgeführt, ab dieser Version vom Prozessor der Grafikkarte. Ab Version 6.0 arbeitet Direct3D im Immediate Mode auf der Basis von Dreiecken (Vertices) und Vektoren, statt mit komplexen Objekten und Hierarchien. Einziger Grund hierfür ist die Geschwindigkeitserhöhung der Anwendungen. Im Unterschied zu OpenGL, bei dem komplexere Objekte wie Polygone) möglich sind, jedoch hat sich die Spieleindustrie im Moment auf DirectX „geeignet“.

Direct3DX Utility Library:

Ist eine Sammlung von mathematischen Funktionen, auf die man in der Programmierung zurückgreifen kann.

Direct Input:

Regelt die Eingabegeräte wie Tastatur, Maus oder Joystick. Ermöglicht Force-Feedback-Effekte. (wie ruckeln des Lenkrades bei Autospielen, usw)

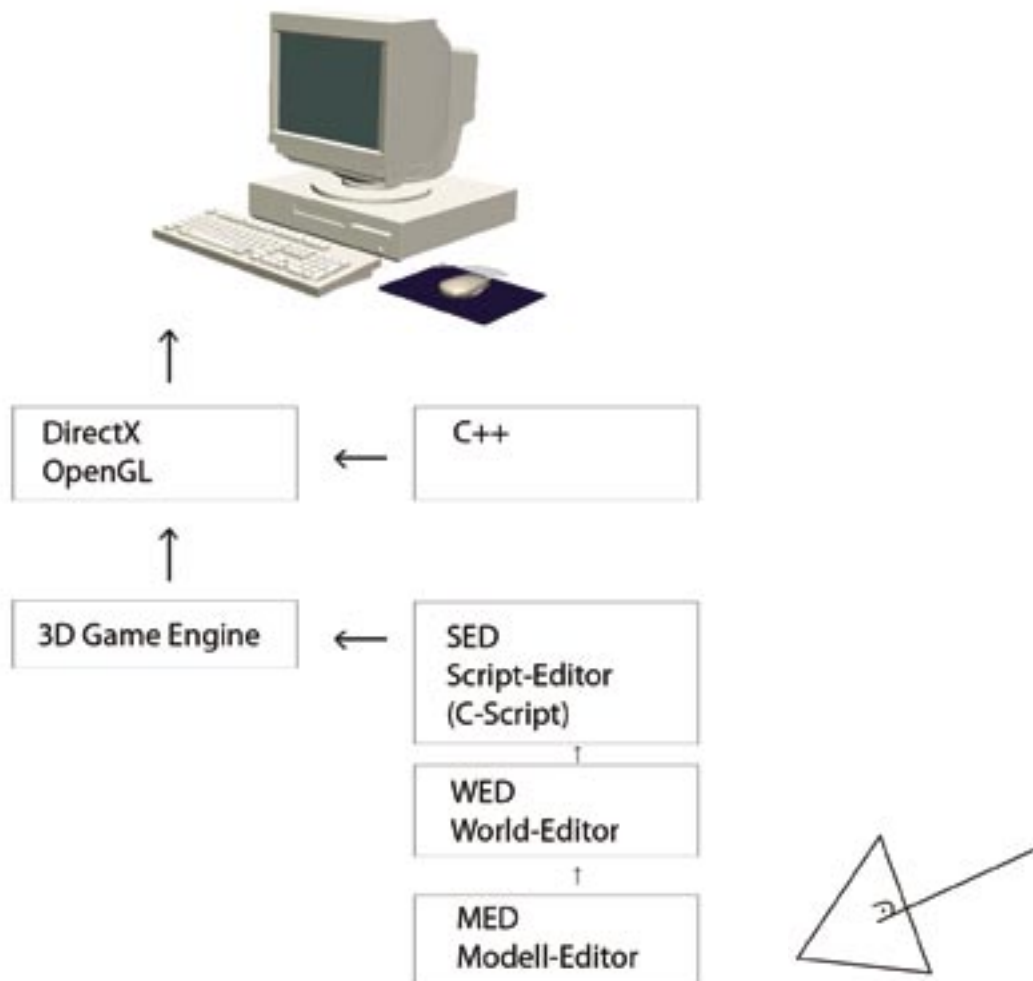
Weiter gibt es noch Direct Sound, Direct Play, Direct Music und Direct Show auf die aber hier nicht eingegangen wird.

### 3.2 3D Game Engine's

3D Engines sind ähnlich wie DirectX Bibliotheken für Grafikfunktionen. Sie stehen jedoch unter DirectX, bzw. OpenGL, da sie diese selbst benutzen. Sie erleichtern den Zugriff auf DirectX. Es gibt viele 3D Game Engines von vielen Entwicklern. (Eine sehr gute Game Engine ist Virtools, jedoch mit fast 5000 Euro nicht gerade preiswert)

### 3.3 Spielesoftware 3D Game Studio

Für die Entwicklung von Architektur\_Engine\_1.0 wurde die Game Engine der Firma *Conneticut Datensysteme GmbH* benutzt: kurz 3D GS. Diese teilt sich in 3 Komponenten: Der Script-Editor beruht auf C++, kann jedoch auch mit C# (sprich C Script) programmiert werden, welches eine vereinfachte Form von C++ ist. Im Model-Editor kann man 3D Objekte erstellen und im Worl-Editor werden die erstellten Modelle mit dem erstellten Script „verbunden“. Hier werden die Entities mit ihrem Script (*behaviour*) belegt. Am Ende entsteht eine Plattform unabhängige Anwendung, die unter DirectX läuft.



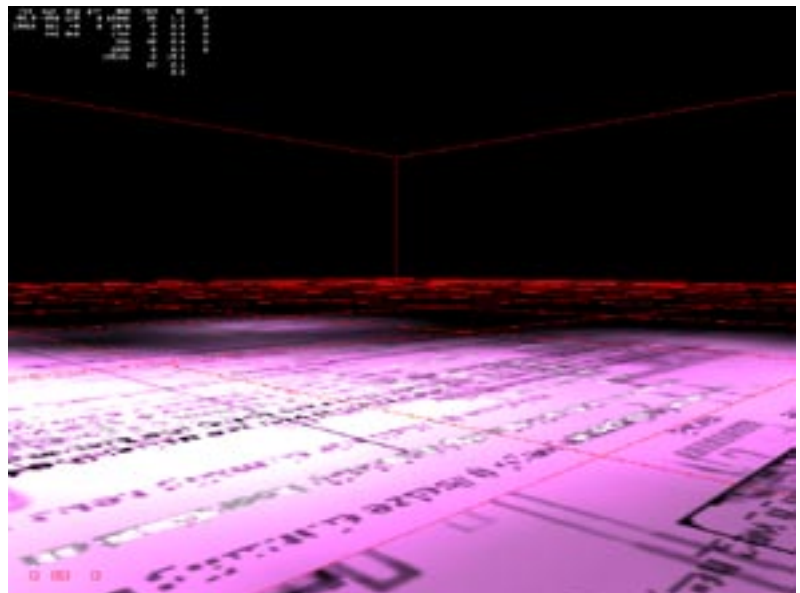
### 3.4 Aufbau eines 3D Spiels

Das räumlich virtuelle Grundgerüst eines Spiels ist ein dreidimensionales Koordinatensystem mit Ursprung. Die Einheiten werden mit Quants bezeichnet. Ein Quant ist beliebig groß, meist entspricht es ca. 3 cm. Auch die Zeit die während des Spiels verstreicht ist nicht leicht zu programmieren, vielmehr bezieht sich die Grundzeit auf die Dauer eines Bildaufbaus (=Frame), also ähnlich eines Films: Bilder pro Sekunde (Frames per Second - fps). Da jedoch jeder PC unterschiedlich lange für die Berechnung eines Bildes braucht, ist auch diese Annäherung ungenau.

Wichtig für jede Game Engine ist es zu wissen, wo die Grenzen des virtuellen Raums liegen. Zu diesem Zweck liegen alle 3D Welten in einer Art Schuhkarton. Diese von nun an *Nullraum* genannte Box, gibt diese Grenze an. Alles was nun innerhalb dieses *Nullraums* existiert, also der Spieler, seine Umgebung, Textfenster, Licht, Texturen, usw. , besteht aus jediglich zwei Typen von Objekten: aus Objekten die steuerbar und programmierbar sind und aus Objekte, die dies nicht sind.

Erstere nennt man Entitys und letztere Blocks.

Das sogenannte Script, geschrieben mit C++ regelt intern das Verhalten der Objekte und extern die 3D Game Engine, bzw DirectX.



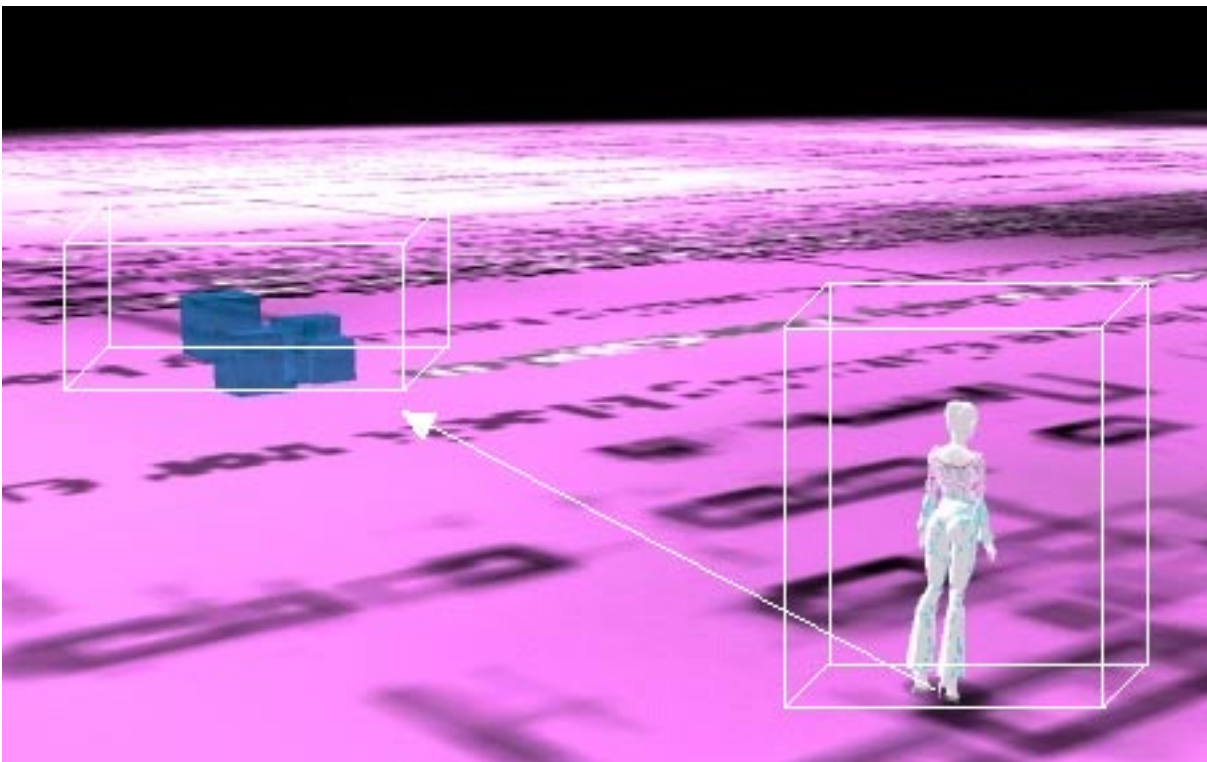
### 3.5 Umgebungserkennung

Entities sind in der Lage ihre virtuelle räumliche Umgebung zu erkennen.

Durch diese Eigenschaft kann der Spieler zum Beispiel erst mit seiner Figur über eine Ebene laufen und nicht durch sie hindurchgleiten. Es gibt verschiedene Arten von Kollisionserkennung, zwei wichtige seien hier kurz erklärt.

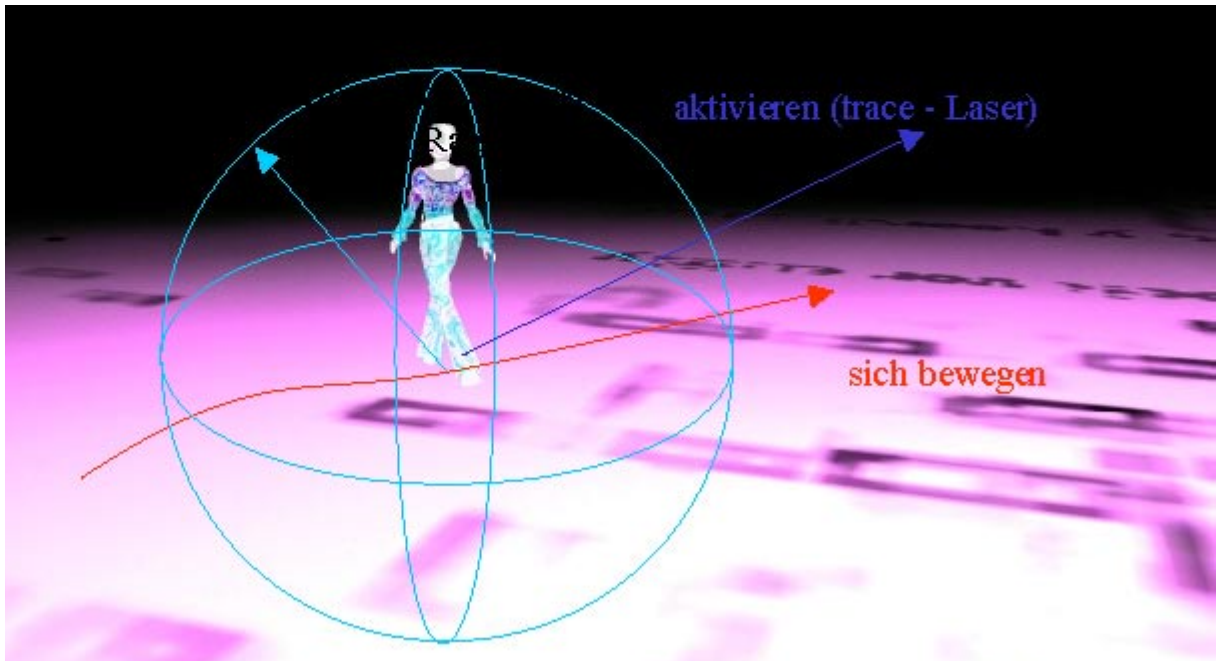
#### 3.5.1 Umgebungserkennung - trace (vectorfrom, vectorto):

Laserstrahl von einem Punkt („from“ Position) zu einem zweiten Punkt („to“-Position), der prüft ob eine Hindernis vorhanden ist. Wenn Hindernis eine Entity ist wird der you-pointer (C# Bezeichnung für die erkannte Entity) auf diese Entity gesetzt. (siehe Abb. 29)



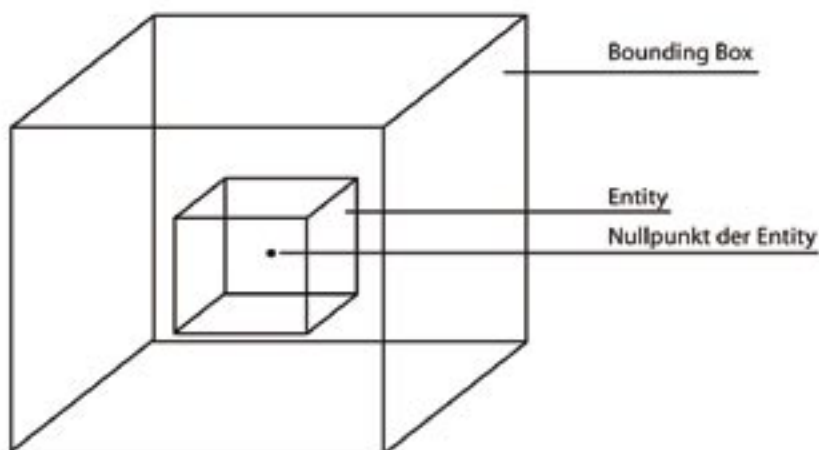
#### 3.5.2 Umgebungserkennung – scan\_entity (arrayfrom, vektorwidth):

Die Anweisung findet alle Entities innerhalb eines Kugelsegments von gegebener Breite und Richtung. Wird eine Entity, die ihr Zentrum innerhalb des Scansegments hat, mit gesetztem Enable\_scan (ermöglicht es gefunden zu werden) gefunden, wird deren Eventfunktion (Event kann eine Funktion sein wie zB: größer werden) ausgelöst. Die Größe der Scankugel wird über die trigger Variable festgelegt.



### 3.5.3 Kollision und Bewegung / Kollisionserkennung - ent\_move ();

Jede Entity und jeder Block hat eine Bounding Box. Diese ist ein 3D Körper einer bestimmten Größe, der um das Objekt liegt. Der Computer errechnet sich die Distanz zweier Objekte, addiert die Abstände der Mittelpunkte der Objekte zu ihrer jeweiligen Boundingbox und vergleicht beide Werte. Ist die Distanz kleiner als die Summe der Abstände, berühren sich für den Computer beide Objekte.



## 3.6 Entities

### 3.6.1 Definition Entity

„Die Entität (lat. ens : das Sein) oder Seinshaftigkeit (auch Wesenheit) (englisch: entity) bezeichnet ein individuelles Einzelelement aus einer Vielzahl von möglichen Elementen der realen oder der Vorstellungswelt. Entität reflektiert die „Seiendheit“ eines Dings, mit der Betonung darauf, „dass“ es ist, im Unterschied davon „was“ es ist.

In der klassischen Philosophie bezeichnet Entität alles, was überhaupt existiert und worüber etwas ausgesagt werden kann. Es wird dabei völlig davon abstrahiert, ob dieses Existierende materiell oder ideell, objektiv oder subjektiv existiert.“

(...)

„Als Entität werden in der Informatik unterscheidbare, in der realen Welt eindeutig identifizierbare Objekte bezeichnet. Die Objekte können sowohl eine physische („real“) oder eine konzeptionelle („abstrakte“) Existenz haben.“

(Wikipedia: Entity)

Entities in der 3D Game Engine können externe Dateien sein. Es gibt vier verschiedene Arten von solchen externen Dateitypen.

### 3.6.2 Modell-Entities - \*.mdl Dateien

Ein Modell ist ein animiertes 3D-Objekt, das als externe MDL-Datei gespeichert ist. Es besteht aus einem 3D-Gitter mit einer darübergespannten, elastischen Texturhaut.

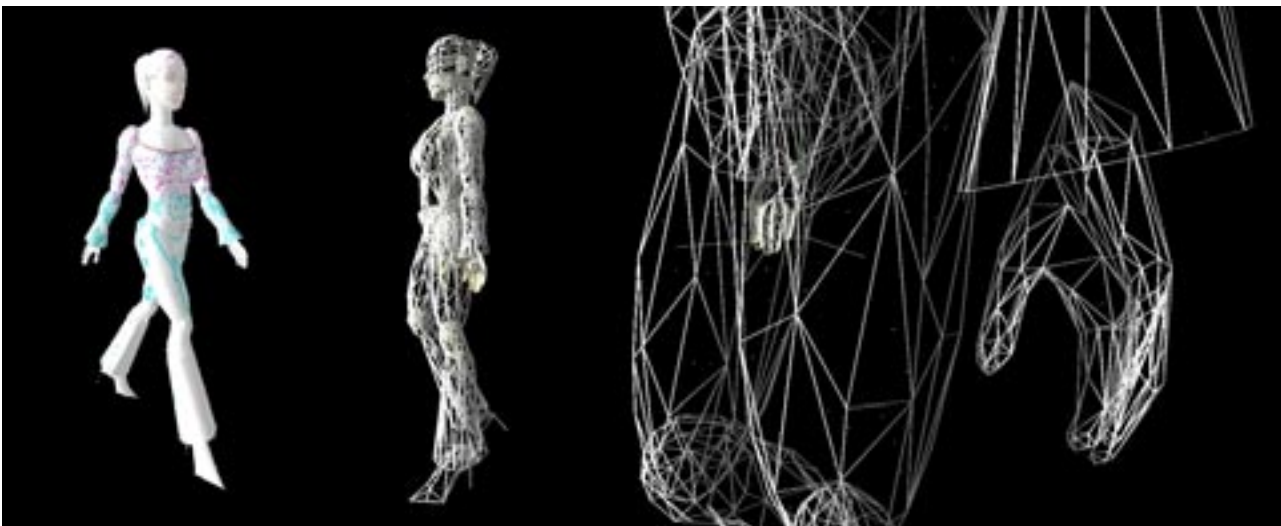


Abb. 31:  
Modell Entity

### 3.6.3 Map-Entities - \*.wmb

Eine Map-Entity ist ein starres 3D-Objekt, welches in einer externen WMB-Datei gespeichert wird. Der Vorteil einer wmb Datei besteht darin dass sie „begehbar“ ist.

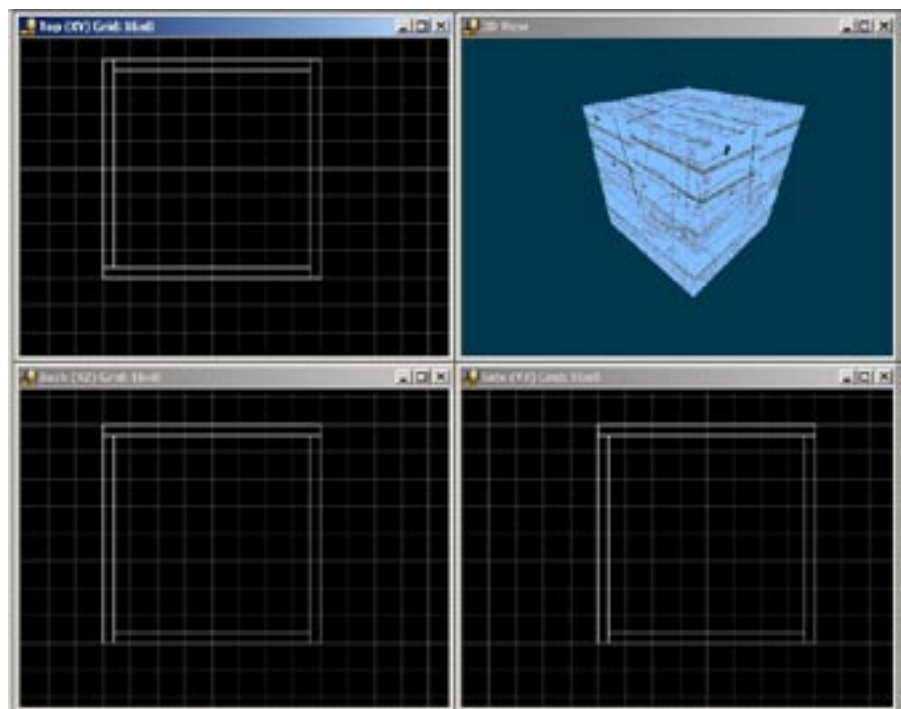


Abb. 32:  
Map Entity

### 3.6.4 Sprite-Entities

Ein Sprite ist ein flaches 2D-Objekt, das entweder wie ein Abziehbild (Decal) an eine Wand gesetzt werden kann, oder sich automatisch immer der Kamera zuwendet. Sprite Entities werden als externe PCX-, BMP- oder TGA-Dateien gespeichert und können mit Malprogrammen wie PaintShop Pro oder Adobe Photoshop erstellt werden. TGA-Sprites enthalten einen Alpha-Kanal, der jedem einzelnen Pixel einen Transparenzwert zuordnet. Häufig werden sie für Explosionen, Lichter, Flammen usw. benutzt.



Abb. 33:  
Sprite Entity

### 3.6.5 Terrain-Entities

Ein Terrain ist ein rechteckiges Gitter von Höhenwerten mit einer darüber gespannten Textur. Es ist in einer externen HMP-Datei gespeichert. Ein Terrain kann aus von Terrain-Generatoren produzierten BMP oder PCX Bitmaps in MED importiert und bearbeitet werden.

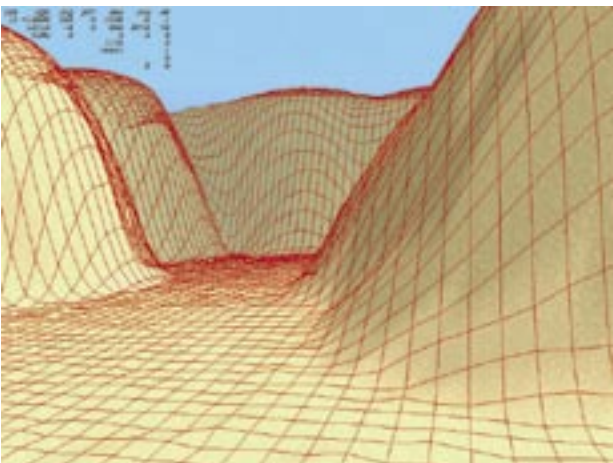


Abb. 34:  
Terrain Entity

### 3.7 Programmiersprachen

Als Einführung in das Thema der Programmiersprachen folgt nun eine kurze unvollständige Übersicht über Techniken und Ziele der Programmierung.

#### 3.7.1 Strukturierte Programmierung

Hierbei ist der Ausgangspunkt meist ein komplexes Problem, für das eine Lösung in Form eines Programms gefunden werden soll. Bei der strukturierten Programmierung versucht man, dieses komplexe Problem in viele kleinere Teilprobleme aufzugliedern. Die Teilprobleme werden dann einzeln bearbeitet und anschließend wieder zusammengesetzt, um die endgültige Lösung zu erhalten.

#### 3.7.2 Prozedurale Programmierung

Bei diesem Ansatz legt man besonderen Wert auf die effiziente und flexible Implementierung des Programms. Dazu entwickelt man möglichst universelle Lösungen für bestimmte Probleme, die man in Prozeduren oder Funktionen (Eine Funktion - als Begriff aus C++ - faßt eine Gruppe von zusammengehörigen Anweisungen unter einem Namen zusammen) zusammenfasst. Das eigentliche Programm besteht dann aus einer Reihe von Funktions- und Prozeduraufrufen mit den passenden Argumenten.

#### 3.7.3 Modulare Programmierung

Die modulare Programmierung geht noch einen Schritt weiter als die prozedurale Programmierung. Hier wird versucht, ein Programm möglichst so aufzuteilen, dass zusammengehörende Daten und Funktionen in einem Modul zusammengefasst sind. Ein Modul ist meist nichts anderes als eine separate Datei mit Quellcode. Mit dieser Methode erreicht man, dass Programmteile, die logisch zueinander gehören, in eine Datei ausgelagert und unabhängig von anderen bearbeitet werden können.

#### 3.7.4 Objektorientierte Programmierung (OOP)

Diese stellt das modernste Konzept bei der Entwicklung dar und vereint verschiedene Programmieransätze. Wichtige Merkmale der objektorientierten Programmierung sind Abstraktion und Kapselung von Daten, Vererbung und Polymorphie. Die objektorientierte Programmierung erlaubt es, Software zu entwickeln, die leichter zu warten ist und damit leichter wiederverwendet werden kann. Dadurch wird der Entwickler in die Lage versetzt, auf vorhandenem Code aufzubauen, und braucht nicht alles bis ins Detail von neuem zu entwickeln.

## 3.8 Begriffe der Programmiersprachen

### 3.8.1 Klassen

Eine Klasse kapselt Daten und Funktionalität in einer Blackbox. Beispiel: Klasse der geometrischen Objekte. Ein Objekt einer Klasse wird durch einen Konstruktor erzeugt. Destruktoren zerstören ein Objekt einer Klasse. Sie kümmern sich um das Freigeben des Speichers. Ein Destruktor wird automatisch aufgerufen, wenn das Programm beendet oder wenn der Gültigkeitsbereich eines Objekts endet, z.B. am Ende einer Funktion.

### 3.8.2 Abstraktion

Abstrakte Klassen sind Objekte die Eigenschaften zusammenfassen ohne selbst ein konkretes Objekt zu sein. Abstrakte Klassen können also Funktionen sein.

### 3.8.3 Kapselung von Daten

Das Prinzip, Daten und zugehörige Funktionalitäten in einem Objekt zusammenzufassen, nennt man Kapselung. Dies bietet viele Vorteile. Zum einen hat man auf diese Weise zusammengehörige Daten an einer Stelle im Programm. Außerdem haben nur die Objekte selbst direkten Zugriff auf ihre Daten. Auch das "Verbergen von Information" genannt, sorgt dafür, dass Objekte den internen Zustand anderer Objekte nicht in unerwarteter Weise lesen oder ändern können; nur den eigenen Methoden eines Objektes soll es erlaubt sein, auf den internen Zustand direkt zuzugreifen. Alle Klassen präsentieren nach außen Schnittstellen, die darüber bestimmen, wie andere Objekte mit ihnen interagieren können.

### 3.8.4 Vererbung

Kurz übersetzt bedeutet Vererbung das Übertragen von Eigenschaften. Mit Hilfe der Vererbung versucht man, verwandte Klassen in Kategorien einzuteilen. Man sucht nach Attributen, die Klassen übergreifend geltend sind. Sie bietet die Möglichkeit, auf die Funktionalität und die Daten von bereits bestehenden Klassen zurückzugreifen, diese anzupassen oder zu verändern und gegebenenfalls um neue Attribute und Methoden zu erweitern.

Eine Klasse die von einer anderen erbt, wird als die abgeleitete Klasse (*derived class*) oder Kindklasse bezeichnet. Die Klasse von der geerbt wird, ist die Basisklasse (*base class*) oder Elternklasse.

Die Vererbung organisiert und erleichtert Polymorphie, indem neue Objekte definiert und erzeugt werden können, die Spezialisierungen schon existierender Objekte sind. Solche neuen Objekte können das vorhandene Verhalten übernehmen und erweitern, ohne dass dieses Urverhalten neu implementiert werden muss. Typischerweise wird das dadurch erreicht, dass Objekte zu Klassen und zu Hierarchien von Klassen gruppiert werden, in denen sich die Gemeinsamkeiten im Verhalten ausdrücken.

### 3.8.5 Polymorphie

Verschiedene Klassen können auf die gleiche Nachricht unterschiedlich reagieren. Polymorphie ist die Eigenschaft einer Variablen, für Objekte verschiedener Klassen stehen zu können, was bei typisierten Sprachen normalerweise nicht möglich ist. Der Sender der Botschaft muss daher nicht wissen, zu welcher Klasse das Objekt gehört.

Polymorphie ist die „Vielgesichtigkeit“ eines Zeigers bzw. einer Referenz. d.h. ein Zeiger oder eine Referenz kann vom Typ her ein Pointer auf eine Elternklasse (*base class*) sein, aber trotzdem auf Objekte der Kindklassen (*derived class*) zeigen.

## 3.9 C# / C-Script

Ein C-Script Programm besteht aus Definitionen und Funktionen. Definitionen erzeugen Objekte und geben ihnen Namen und Anfangs-Eigenschaften; Funktionen legen das Verhalten dieser Objekte im Spiel fest, indem sie diese Eigenschaften dynamisch verändern.

### 3.9.1 Funktionen (abstrakte Klassen)

Eine Funktion besteht aus dem Funktionskopf und dem Funktionsrumpf. Jede Funktion gibt einen Wert zurück. Die folgende Zeile zeigt den theoretischen Aufbau einer Funktion.

(Rückgabe-) Datentyp Funktionsbezeichnung (Datentyp Argument1, Datentyp Argument2, ...)

Argumente stellen Variablen dar, die an eine Funktion übergeben werden können. Auch sie müssen zu einem bestimmten Datentyp gehören. Der Funktionsrumpf steht wieder zwischen den beiden Klammern.

```
{  
Anweisung1;  
Anweisung2;  
Anweisung3;  
}
```

### 3.9.2 Objekte (Klassen)

C-Script kennt folgende Arten von Objekten: BMAP, Entity, Function, Panel, String, Material. Jedes Objekt hat eine bestimmte Anzahl von Eigenschaften, welche über den Punkt-Operator „.“ angesprochen und ausgelesen werden können:

OBJEKTNAME.EIGENSCHAFT

In der ARCHITECTURE\_ENGINE\_1.0 sind die Entities Objekte dieser Art. Ihre 3D-Datenstruktur ist jedoch begrenzt auf das .wmb Dateiformat, welches auf Quadern basiert. Dies erklärt sich aus der internen Kollisionserkennung der 3D Game Engine.

Den Entities werden Funktionen zugewiesen (können als abstrakte Klasse betrachtet werden), welche nun das Verhalten in der Anwendung steuern.

### 3.9.3 Scripting / Pseudocode

Beispiel aus einem Script für die ARCHITECTURE\_ENGINE\_1.0

```
function_a
```

```
{  
    x-mal mache folgendes:  
    schaffe_entity (AE_1, Position, function_b);  
}
```

```
function_b
```

```
{  
    verhalte dich nach Code_1 und Code_2 (werden weiter unten erklärt)  
    Code_1: Grundverhalten;  
    Code_2: Reaktionsverhalten;  
    Reaktion auf FE:  
    1. in der Nähe von FE: bleibe dort stehen, warte x min, lösche dich und  
    rufe funktion_a auf  
    2. FE aktiviert dich: funktion_c  
    Reaktion auf AE:  
    wenn ich Entity erkenne, warte x min, lösche dich und rufe funktion_a auf  
}
```

```
function_c
```

```
{  
    verhalte dich nach Code_3 und Code_4 (werden weiter unten erklärt)  
    Code_3: setze dich von FE weg in einer Richtung immer wieder neu, x-mal  
    Code_4 :sei fähig zu reagieren auf FE  
    Reaktion auf FE:  
    1. FE aktiviert dich :lösche dich selbst  
}
```

## //4\_ Anwendungen////////////////////////////////////

*„It is extremely relevant that the designers don't just talk about the process, but that they actually make it work. You must run the process and work in the process.“*

(Oosterhuis 2003: 74)

### 4.1 Allgemein

Im folgenden wird nun versucht zu zeigen wie die oben erklärte Technologie der programmierbaren Game Engine für Entwurfsprozesse genutzt werden kann, wobei sich die Arbeit eher als Annäherung an Entwurfsprozesse versteht und nicht als fertiges Entwurfstool. Die operative Komponente im Entwurfsprozess bekommt dadurch einen höheren Stellenwert als die Originalität

Das Ergebnis sind reaktive virtuelle dreidimensionale Umgebungen, die der Nutzer per Interface in Echtzeit beeinflussen kann.

Kas Oosterhuis stellt die richtigen Fragen:

*„The challenging issue here is: who sets the rules for the playful building? What are the boundaries of the playground? Who decides what formulas are used? Who writes the procedures? Who creates the scripts, who writes the genetic code? And who is authorized to change the rules of the game?“*

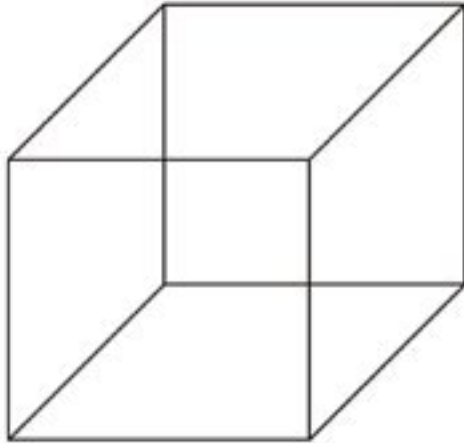
(Oosterhuis 2003: 76)

### 4.2 Anwendungsmöglichkeiten oder „Was soll das ganze?“

Durch die Nutzung der Technologie der programmierbaren 3D Engine, insbesondere durch die oben definierten Entitäten und deren Code, also deren programmiertes Verhalten (behaviour), entsteht eine Gleichwertigkeit in der Entwurfspraxis. Es gibt keinen Unterschied mehr zwischen Subjekt und Objekt. Alle Entitäten sind Dinge mit einem programmierten Verhalten (=behaviour), die in einem dreidimensionalen Raum theoretisch alles können. Sie erkennen sich dabei gegenseitig (!) und der Bildschirm zeigt uns einen Ausschnitt (frame) aus dieser virtuellen Welt. Dazu ist der Anwender in Lage selbst zu einer Entity zu werden und sich in dieser Welt zu bewegen. Der Anwender hat im Vergleich zu Simulationen immer die Möglichkeit, in den laufenden Prozess einzugreifen und zwar nicht nur im Sinne der definierten Regeln, sondern auch gegen diese, also zielführend oder nicht.

Die Begrenztheit liegt wenn, dann in der Regelmäßigkeit (Algorithmen) der Entwurfsparameter, soll heißen, zu Beginn müssen erst, auf der Entwicklungsebene, entwurfsrelevante Verhalten definiert werden, die dann gewissen Entitäten zugewiesen werden. Die Form der Entities, also die Form der in der Anwendung vorhandenen 3D Datei, ist wiederum veränderbar. Das bedeutet eine Entity kann gescalt, gemorphet oder mit anderen Entities durch boolesche Operationen in Beziehung gebracht werden. In Verbindung mit anderen Softwareumgebungen wie z.B. 3DS Max ist auch hierbei keine Grenze zu setzen.

Durch das Prinzip der Vererbung, also durch die Kapselung von Daten in externen Abschnitten, ist der Zugriff auf diese Daten, wie z.B. die Form (die sich ja aus Punkten und deren Verbindung ergibt) möglich. Das bedeutet es können neue Entities entstehen, Entities können Eigenschaften von anderen Entities vererbt bekommen. (siehe Abb. 35)



Nullraum: wird gebraucht aus internen technischen Gründen.

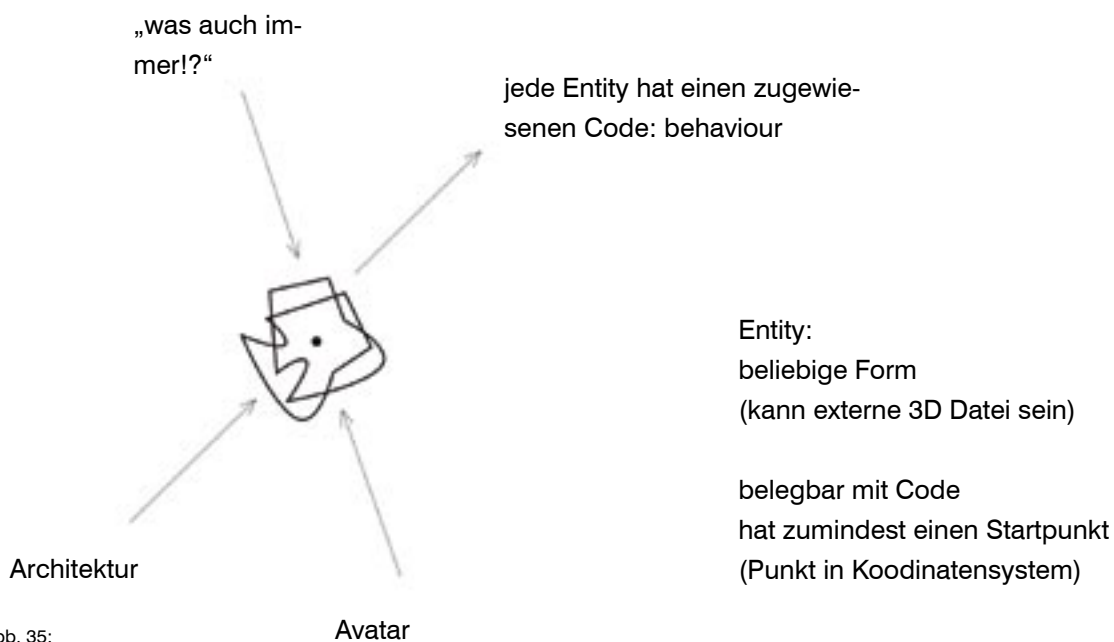


Abb. 35:  
Nullraum und Entity

Zur kurzen Wiederholung:

Eine Entity in der Anwendung hat immer genau eine Funktion (behaviour), also einen Code – programmiert im Script. Dieser Code entspricht inhaltlich der vorher definierten Verhaltensregel, die je nach Entwurfparametern anders aussieht. Egal was dieser Code auch auslöst, sind zwei Eigenschaften grundlegend sehr wichtig:

1. Eine Entity kann sich selbst oder andere entfernen.
2. Sie kann eine neue Entity schaffen und ihr eine Funktion zuweisen.

Der Befehl in C-Script heisst:

zu 1: `ent_remove (me)` bedeutet: entferne Entity (Name der Entity)

zu 2: `ent_create (entity, position, function)` bedeutet : schaffe Entity (Name, Koordinaten, zugewiesene Funktion)

(siehe Abb. 36)

Entity hat eine Funktion → Funktion (=Code )

- zwei wichtige Eigenschaften:
- o kann sich selbst löschen
  - o kann weitere Entity entstehen lassen und dieser eine Funktion/Code zuweisen

Abb. 36:  
Entity und Funktion

## 4.3 Struktur einer Anwendung

### 4.3.1 Entwicklungsebene

Hier müssen entwurfsrelevante Algorithmen festgelegt werden. Diese werden dann in einen Pseudoscript übertragen und schliesslich in eine Programmiersprache übersetzt.

### 4.3.2 Anwendungsebene

In der Anwendungsebene passieren mehrere Aktionen räumlich nebeneinander. Die Entities „handeln“ entsprechend ihrer Programmierung, der Anwender kann manipulierend durch direkte Eingriffe per Interface, also meist Tastaturbefehlen, eingreifen oder per Avatar am „Spiel“ teilnehmen. Vorher definierte Umgebungsbedingungen, wie z.B. die feste Umgebung (entspräche einer städtebaulichen Situation), Zeitdruck oder Ressourcenknappheit beeinflussen das Ergebnis. (siehe Abb. 38)

Architektur ist also ein Standbild (freeze frame) eines Prozesses. Dieser Prozess besteht aus interagierenden Entitäten und der Interaktion des Anwenders auf der Ebene des Eingreifens (per Tastatur) und der Ebene des „Mitspielens“ (als Entity). (siehe Abb. 39)

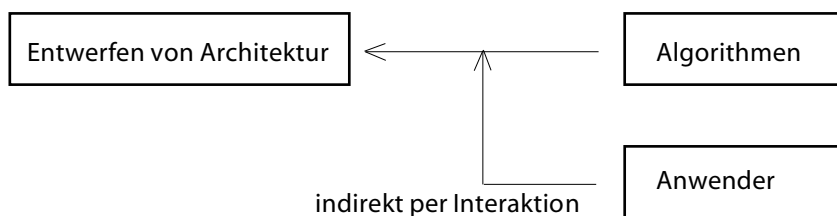


Abb. 37:  
Entwerfen von Architektur

---

ENTWICKLUNGSEBENE

Scriptentwicklung

Entwickeln entwurfsrelevanter Algorithmen

---

ANWENDUNGSEBENE

Block

Nullraum (braucht die Engine)  
statische Umgebung

Entities

max Anzahl zur Zeit ca 300

Funktionen

views  
Verhaltenscodes  
(Algorithmen)

Codes  
(Script-Ebene)

Umgebungsbedingungen

Ressourcen, Punktekonto  
Zeit / Dynamik  
Handlungsdichte

Anwender / Nutzer - Ebene

Eingriff per Interface / (Tastatur)

---

Abb. 38:  
Entwicklungs- und Anwendungsebene allgemein

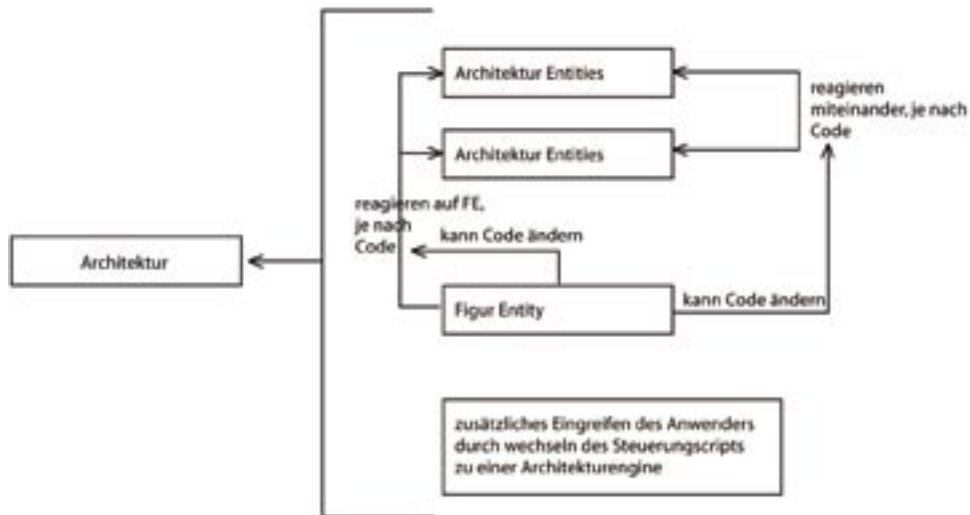


Abb. 39:  
Architektur als Ergebnis eines Prozesses

#### 4.4 Mögliche Anwendungen

Ähnlich wie bei Simulationen müssen im Vorfeld Aufgaben erst einmal definiert werden. Das bedeutet entwurfsrelevante Parameter müssen abstrahiert und zu Regeln übersetzt werden.

Zu unterscheiden sind Umgebungsbedingungen oder Randbedingungen und inhaltliche Aktionen. Diese Randbedingungen können wie oben schon erwähnt in der Architektur die Umgebungsbebauung sein, Bestimmungen des Bebauungsplans wie Höhe oder Baumasse, Zeit- oder Ressourcenmangel und vieles mehr. Die Entities in der Anwendung verhalten sich nach ihrer Programmierung. So sind z.B. folgenden Algorithmen denkbar:

##### 4.4.1 Transfer eines Raumprogramms

Transfer eines Raumprogramms mit unterschiedlichen Größen und Abhängigkeiten:

Verschieden große Entities bekommen Verhaltensregeln nach denen sie bestimmte Entities anziehen, bestimmte Entities ignorieren und von anderen abgestoßen werden. Dabei werden sie alle von Attraktoren angezogen, die in ihrer Lage den Bauplatz berücksichtigen. Durch zeitliches Begrenzen und wiederholen von einzelnen Entities, läuft ein Prozess immer weiter und immer neue Ergebnisse entstehen.

#### 4.4.2 Transfer eines Nutzungsschemas

Durch Programmierung von Entities, die menschliche Aktivitäten simulieren, zum Beispiel Arbeitsabläufe, entstehen Raumstrukturen durch räumliche Entities, die auf diese Figuren reagieren.

#### 4.4.3 Arbeiten mit Künstlicher Intelligenz (KI)

Anfangen von path-finding Algorithmen bis hin zu komplexen Verhaltensmustern von Menschen gibt es schon Codes (behaviour), die für unterschiedliche Zwecke benutzt werden. So ist theoretisch für jedes Vorhaben ein Algorithmus zu finden. Durch die Überlagerung mehrerer Entities, die sich ja gegenseitig erkennen, entstehen hoch komplexe Ergebnisse.

Der Anwender ist nun in der Lage einmal extern auf eine laufende Anwendung einzuwirken oder intern, indem er eine der Entities selbst steuert. Gerade durch die interne Steuerung kann er weitere Situationen herbeiführen und in „Echtzeit“ das Ergebnis überprüfen.

Dabei ist keine Unterscheidung mehr zwischen Objekt und Subjekt notwendig. Der Anwender kann den Nutzer spielen oder zu Architektur werden. Er kann also derjenige sein, der Prozesse auslöst oder der auf Prozesse reagiert.

---

ENTWICKLUNGSEBENE

Scriptentwicklung

Entwickeln entwurfsrelevanter Algorithmen

---

ANWENDUNGSEBENE

Block

Nullraum

Entities

View (VE)

Figure Entity (FE)

Architektur Entity (AE)

Funktionen

1st person view  
3rd person view  
god view

Steuerungsfunktion  
(= Steuerungsscode)

Verhaltensfunktionen  
(=Verhaltenscodes)

Codes  
(Script-Ebene)

Code für :  
1st person view  
3rd person view  
god view

Code für  
Steuerungsfunktion  
(ermöglicht die Steuerung  
per Interface)

Code\_1 : Grundverhalten  
Code\_2: von Entity weg  
Code\_3: um Entity herum  
Code\_4: unsichtbarer Starter  
Code\_5: Garten  
Code\_6: Besucher  
usw.

Umgebungsbedingungen

Energie

Zeit / Dynamik  
Handlungsdichte

Anwender / Nutzer - Ebene

Stiegen / Rampen setzen

View wechseln

Standbild / Pause /

---

Abb. 40:  
Entwicklungs- und Anwendungsebene ARCHITECTURE\_ENGINE\_1.0

# //5\_Architecture\_Engine\_1.0////////////////////////////////////

ARCHITECTURE\_ENGINE\_1.0 ist eine Annäherung an einen Entwurfsprozess und versucht die Methodik der beschriebenen Technologien an einem Beispiel darzustellen.



In der von mir entwickelten Anwendung wurde versucht möglichst viel der technischen Aspekte zu zeigen. So wurde in der ARCHITECTURE\_ENGINE\_1.0 von einer figürlichen Präsenz des Anwenders in Form eines Avatars (Spielfigur) ausgegangen. Es gibt eine Schwerkraft und die Figur bewegt sich menschenähnlich. Das bedeutet sie kann in alle Richtungen gehen, Treppen steigen und eine gewissen Höhe hüpfen. Im weiteren gibt es von Start an vier unterscheidbare Architektur Entities die sich durch ihre Größe und in ihrem Verhaltenscode auf die Figur beziehen. Der Anwender ist somit in der Lage durch seine Bewegungen im virtuellen Raum Architektur entstehen zu lassen. Im weiteren kann er durch „Aktivieren“ (LMT = linke Maus Taste) einer Entity deren Code austauschen, sodass diese sich nun anders verhält.

Um die verschiedene Möglichkeiten der Views (Wahrnehmungsräume in die virtuelle Welt) zu zeigen, kann der Anwender von dem 1st person view in den god-view wechseln, oder in den 3rd person view einer ihn umgebenden Architektur Entity (AE). Aber er kann auch seine Identität zu einer Architektur Entity wechseln, sodass er nun selbst zum Objekt wird.

Der Anwender kann im weiteren per Tastatur in die virtuelle Welt eingreifen, indem er Entities entstehen lässt. In der ARCHITECTURE\_ENGINE\_1.0 sind dies Stiegen und Rampen, die der Figur ein in die Höhe gehen ermöglichen.

Als Umgebungsbedingungen gibt es ein Punktekonto, das alle fünf Sekunden kleiner wird. Erst durch Aktivität des Anwenders bekommt dieses Punkte: Aktionszwang! Wenn das Punktekonto leer ist, ist das Spiel vorbei.

## 5.1 Gebrauchsanweisung

Der Nutzer der ARCHITECTURE\_ENGINE\_1.0 startet die Anwendung in der Ego-Perspektive einer virtuellen Figur. Er startet im sogenannten Nullraum der gewählten Bauaufgabe. Die Figur kann sich per Maus und Tastatur (Interface) in alle Richtungen bewegen. Von Start an existieren ebenfalls vier unterschiedliche Architektur Entities (AE) auf die später noch eingegangen wird. Diese AE's haben einen Code (behaviour) der sie die Figur suchen lässt und sich auf diese zubewegt.

Es gibt kein Ziel und keine konkrete Aufgabe, jedoch wird man zu Aktivität gezwungen, denn durch Fortschreiten der Zeit werden der Figur Punkte abgezogen und bei null Punkten ist die Anwendung vorbei. Game Over. Erst durch Aktivitäten erhält die Figur (FE) wieder Punkte.

Die Figur ist nun in der Lage durch Bewegung Räume entstehen zu lassen. Durch aktivieren einer AE kann sie deren Code ändern, so dass sich diese AE nun von ihr wegbewegt und sich dabei selbst kopiert. Durch Tastatureingaben ist der Nutzer in der Lage direkt vor der Figur eine Stiege oder eine Rampe entstehen zu lassen (siehe Abb.), sodass die Figur auch in die Höhe steigen kann. Weitere Tasten ermöglichen es die Perspektive des Nutzers auf die virtuelle Welt zu ändern. So kann er in die Gott Perspektive wechseln oder in der Perspektive einer auf die Figur zusteuern AE. Ebenfalls kann der Nutzer selbst zu Architektur werden, indem er zu einer AE wird.

## 5.2 Funktionsweise der ARCHITECTURE\_ENGINE\_1.0

Wie oben bereits erwähnt ist einer der Hauptmerkmale der ARCHITECTURE\_ENGINE\_1.0 die Gleichstellung von Architektur Elementen auf der Ebene der Programmierung. Die Architektur wird zur Architektur Entity.

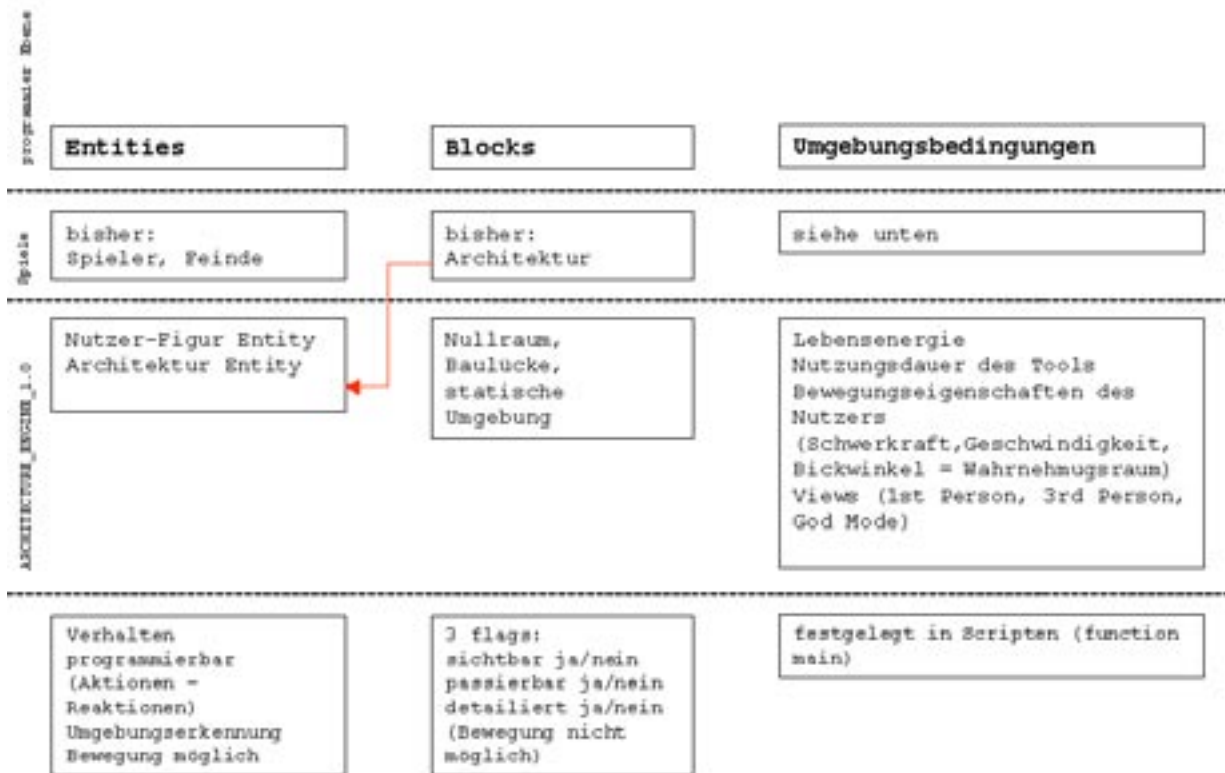


Abb. 41:  
Tabelle Vergleich der Nutzung von Entities und Blocks

Die Anwendung Architecture\_Engine\_1.0 hat zu Beginn die Figur Entity (FE) und vier Architektur Entities (AE).

Der Nutzer sieht die virtuelle Welt aus den Augen der FE (1st Person View). Die vier AE haben ein sogenanntes Grundverhalten (Code\_1). Der Nutzer ist in der Lage das Verhalten der AE, also deren Code zu ändern (indem er mit der linken Maustaste eine AE anklickt). Alle entwickelten Bewegungsalgorithmen setzen die AE räumlich in Bezug auf die FE. Durch den räumlichen Bezug, entsteht ein Raumgefüge welches durch die Bewegung der Figur-Entity definiert ist. (In weiteren Entwicklungen für Entwurfsprozesse könnte man nun die Bewegungen der FE inhaltlich bzw. programmatisch leiten. Vorstellbar sind räumliche Handlungsabläufe wie zb. in einem Herstellungsbetrieb oder in einem Krankenhaus.)

### 5.3 Anwendungsverlauf

Anhand eines möglichen Anwendungsverlaufs werden nun chronologisch verschieden mögliche Ergebnisse beschrieben. Danach werden die in der Anwendung benutzten Verhalten- und Bewegungsalgorithmen erläutert.

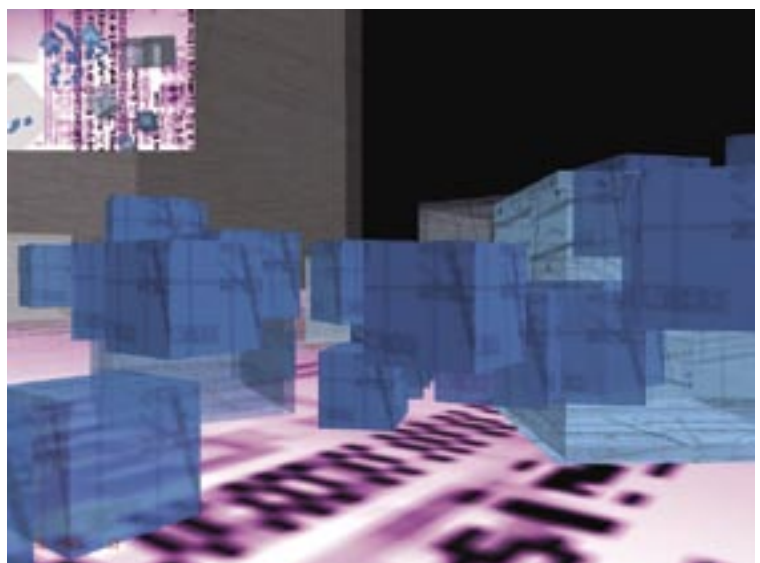
#### 5.3.1 Spielstart

Der Nutzer startet als Figur Entity und sieht auf dem Bildschirm aus der Ego Perspektive (1st Person view) um sich vier verschieden Architektur Entities auf sich zukommen. Er hat 100 sogenannte Energie Punkte. Durch Drücken der Taste „T“ erscheint oben links der Grundriss Blick (Top view). Ungefähr jede fünf Sekunden werden der Figur fünf Energie Punkte abgezogen. Die auf die Figur zusteuenden AE's bleiben vor der Figur stehen, verweilen dort eine Zeit und löschen sich danach selbst. Je kleiner eine AE ist, desto näher kann sie an die FE herankommen. In die drei größeren AE's kann sich die Figur hineinbewegen und auf oder in ihr laufen. Die kleinste AE ist nur begehbar.

Abb. 42:  
Spielstart



Abb. 43:  
Sicht des Anwenders bei Spielstart



### 5.3.2 Stiege / Rampe

Der Anwender ist nun in der Lage durch Drücken der Taste „S“ eine Stiege direkt vor der FE entstehen zu lassen oder eine Rampe durch Taste „R“. Diese sind begehbar und löschen sich nach einiger Zeit von selbst. Ebenfalls durch Aktivieren, also durch Drücken der linken Maustaste, sind sie zu entfernen.

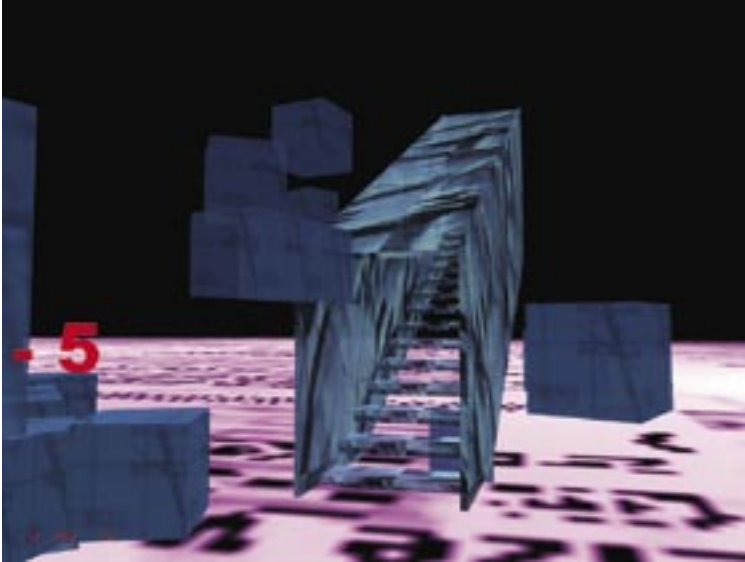


Abb. 44:  
Taste „S“ = Stiege

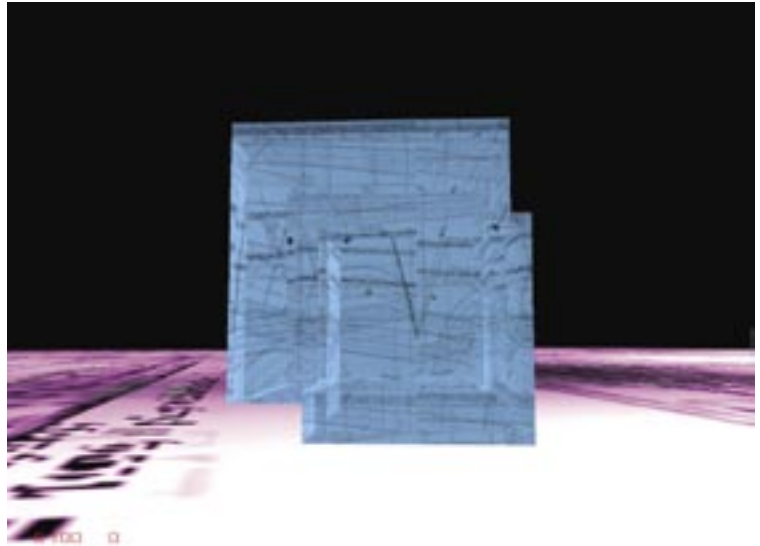


Abb. 45:  
Taste „R“ = Rampe

### 5.3.3 Vererbung

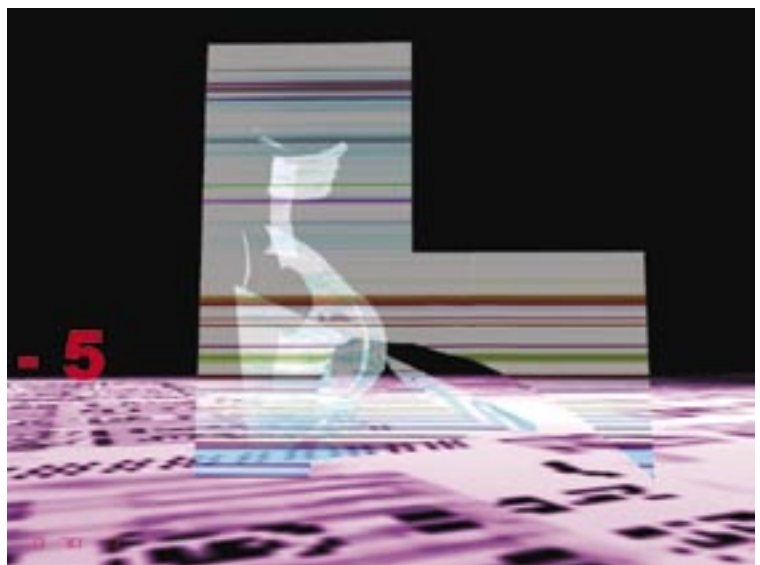
Nach kurzer Zeit haben die Architektur Entities (base class) ihre Form, die sie durch Überlagerung um die Figur Entity erlangt haben, an die neu entstehenden AE's vererbt. Sodass die nun zweite AE Generation (derived class) eine andere Form hat. Dieser Vorgang wiederholt sich nun laufend.

Abb. 46:  
AE der zweiten Generation, mit vererbter Form-Eigenschaft entstanden aus zwei AE's



Durch Überlagerung der Architektur Entity mit der Figur Entity entstehen ebenfalls neue Entities. Diese vererben sich auch die Texturinformation. Die neu entstehende Textur ist eine Überlagerung der AE Textur und der FE Textur.

Abb. 47:  
Beispiel einer Entity, entstanden aus AE und FE



### 5.3.4 Aktivieren

Durch Aktivieren einer AE (LMT = Linke Maus Taste) wird deren *behaviour*, also der per Script zugewiesene Code geändert. Die AE die sich bisher auf die FE zubewegt hat, kopiert sich nun in eine Richtung von dieser weg. Die FE erhält dadurch zehn Lebenspunkte.

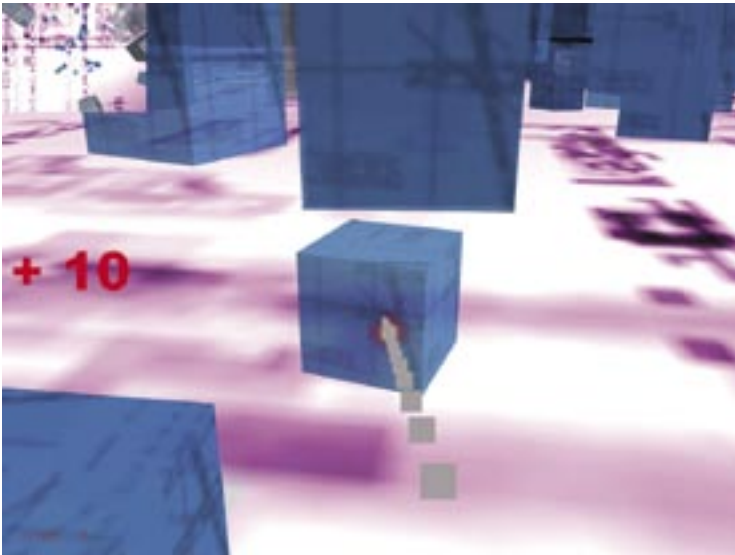


Abb. 48:  
Aktivieren einer AE Typ S (Small)  
und deren Verhalten ändert sich zu...

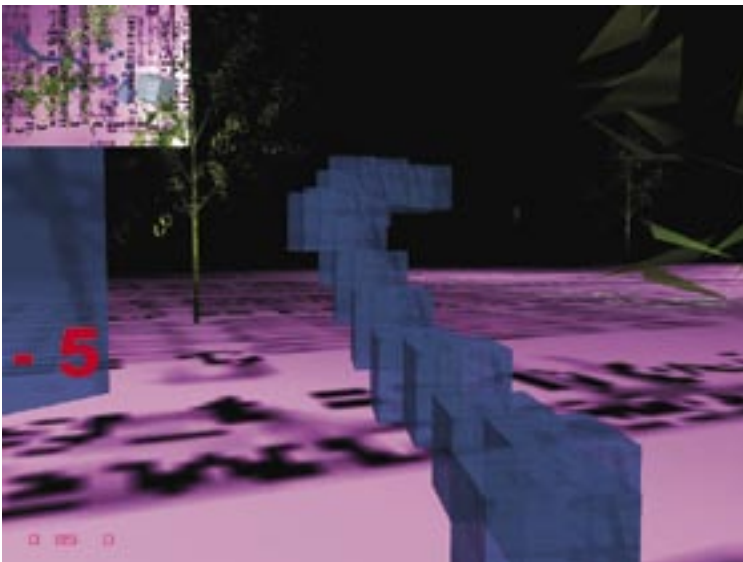


Abb. 49:  
AE die sich von FE weg kopiert  
(Verhaltens Code\_2)

Beim Aktivieren einer AE des Typs L (Large) oder XL (X-Large) wird nicht nur deren Code geändert, sondern zudem noch ein nicht sichtbarer „Starter“ gesetzt.

Dieser unsichtbare Starter hat die Form eines Würfels, der, wenn eine Entity in seine Nähe gekommen, ein Event auslöst. Events nennt man Ereignisse die durch Aktionen ausgelöst werden. Dieses Event (Code\_Besucher) lädt eine weitere Figur Entity in die Anwendung. Diese Figur hat eine einfache KI (Künstliche Intelligenz) Programmierung, die sie durch die virtuelle Umgebung laufen lässt und dabei Hindernissen ausweichen kann.

Ein weiteres Event ist der Code\_Garten. Er lädt zwei Typen von Bäumen um die Figur Entity herum. Beide Events sind zeitlich begrenzt.

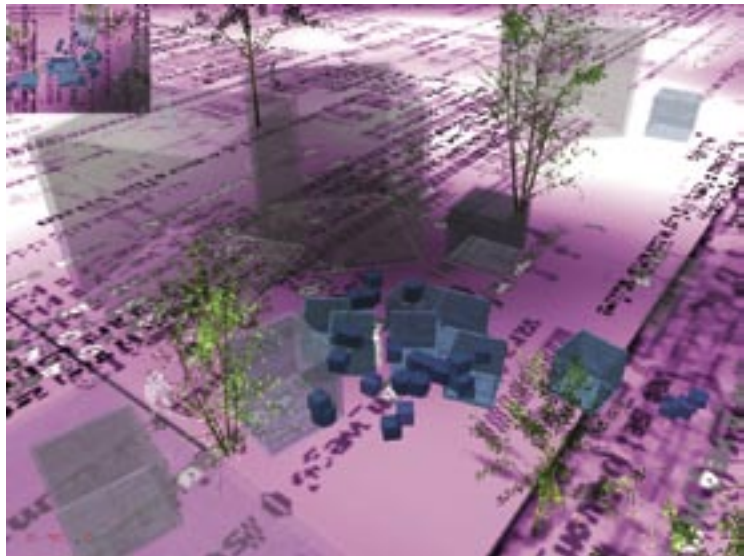


Abb. 50:  
Code\_Besucher und Code\_Garten

## 5.4 Wahrnehmungsräume

### 5.4.1 1st Person View

Der Bildschirm der Anwendung ARCHITECTURE\_ENGINE\_1.0 gibt einen Blick in die virtuelle Umgebung der laufenden Anwendung wieder. Er zeigt 24 Standbilder pro Sekunde (24 fps). Dieser Blick, auch View genannt, ist auf der Programmier Ebene ebenfalls eine Entity. Das bedeutet der View ist steuerbar. Bei Anwendungsstart simuliert er den Blick der Figur Entity (1st Person View). Mit der Taste „V“ kann der Nutzer den Blick einer AE annehmen. Das bedeutet die Figur Entity bleibt erhalten und weiterhin auch per Maus steuerbar, jedoch zeigt der Bildschirm nun den Blick einer auf die FE zu steuernden AE.

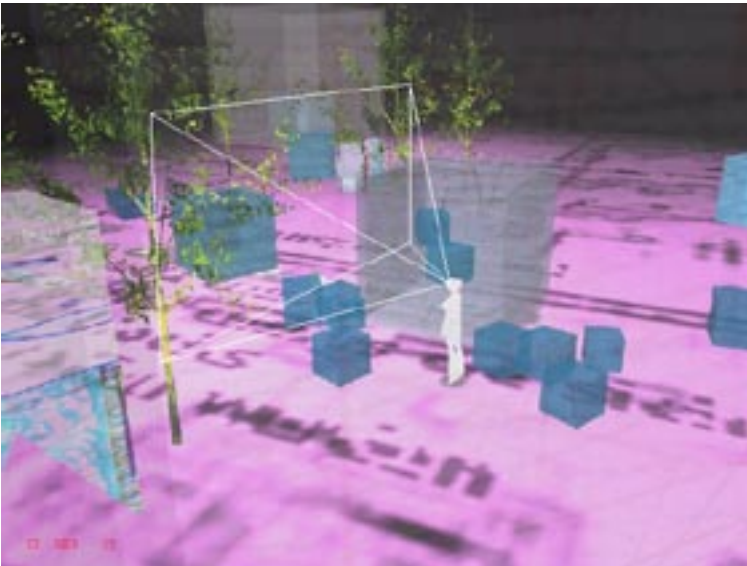


Abb. 51:  
Ego-Perspektive (1st Person View)  
Blick aus der Figur-Entity (FE)

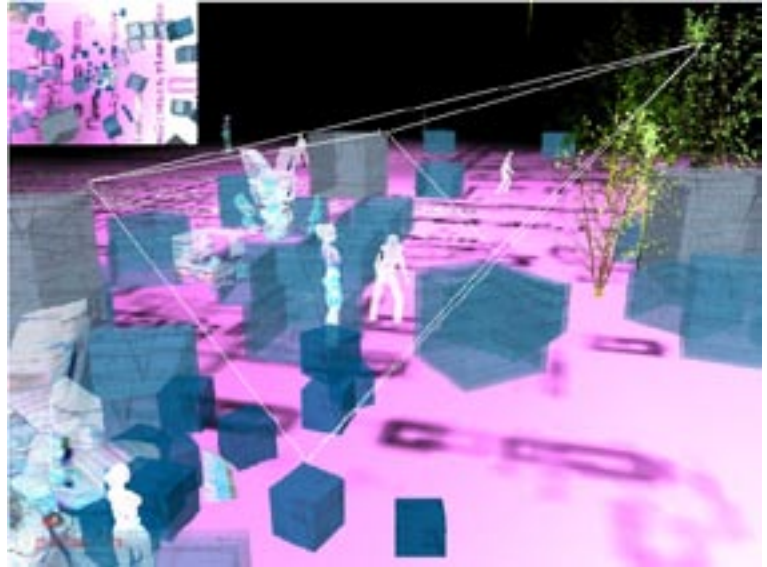


Abb. 52:  
Ego-Perspektive (1st Person View).  
Blick aus einer Architektur-Entity (AE) (FE bleibt steuerbar)

#### 5.4,2 God View

Die sogenannte Gott Perspektive (God View) ermöglicht es dem Anwender aus der „Haut des FE“ zu schlüpfen und gottgleich durch den Raum zu schweben. Die Figur Entity jedoch ist zu dieser Zeit nicht mehr steuerbar.

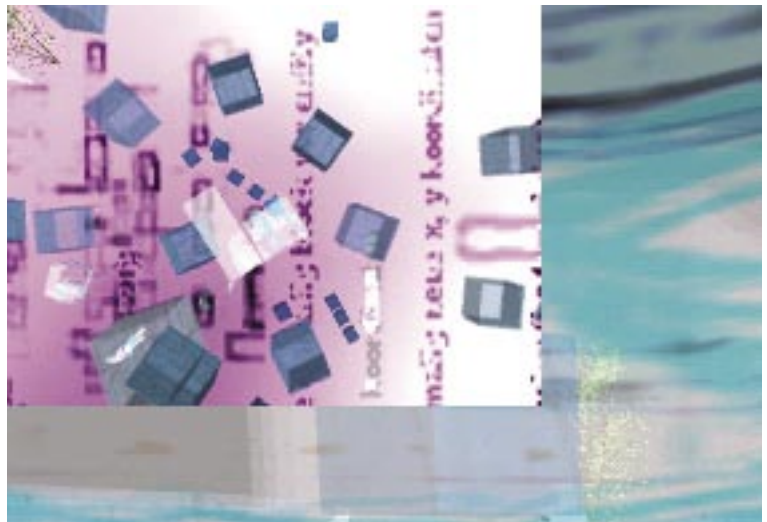
Abb. 53:  
God View  
alles umfassender Blick, frei bewegliche Kamera  
(FE nicht steuerbar)



#### 5.4.3 Grundriss View

Durch Aktivieren der Taste „T“ erscheint am oberen linken Rand ein Fenster, mit dem Blick einer „Kamera“ die genau über der agierenden Entity plziert ist. Sie bewegt sich mit dieser Entity mit.

Abb. 54:  
Grundriss View



## 5.5 Subjekt = Objekt / Objekt = Subjekt

Die Möglichkeiten der ARCHITEKTUR\_ENGINE\_1.0 ausnutzend ist der Anwender auch in der Lage selbst zu Architektur zu werden. Durch die Taste „u“ wird er zu einer Architektur Entity, die sich verhält wie vorher die Figur Entity. Diese bleibt in der virtuellen Umgebung erhalten und bekommt eine einfache künstliche Intelligenz, wie die oben erwähnten Besucher Entities (Non Playing Character - NPC). Durch Drücken der Taste „o“ wird diese Aktion wieder rückgängig gemacht und der Anwender ist wieder in der Rolle der FE. Alle Änderungen der Views sind auch als AE anwendbar.

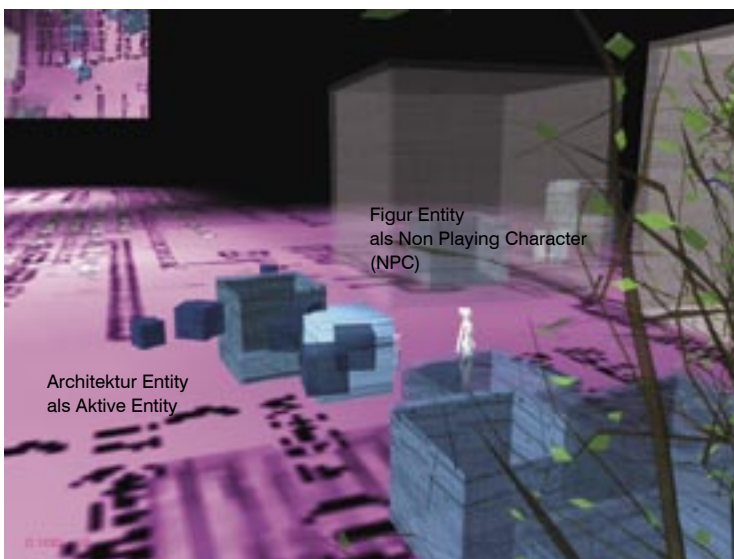


Abb. 55:  
Subjekt = Objekt / Objekt = Subjekt

## 5.6 Architektur Entities

Ausgangspunkt der ARCHITECTURE\_ENGINE\_1.0 sind neben der Figur Entity vier unterscheidbare Architektur Entities. Diese sind, wie die Verhaltenscodes, räumlich, nämlich durch ihre Größe, auf den Anwender, also meist die FE, bezogen.

Small (S), Medium (M), Large (L) und X Large (XXL).

Die Architektur Entities sind ihres Dateitypus nach Map Entities. Diese externen Dateien können also einfach und schnell ausgetauscht werden.

Andere, dem jeweiligen Entwurfsaugenmerk entsprechende, Formen sind vorstellbar.

AE Small (S)

Kleinste für die Figur noch nutzbare Einheit.

AE Medium (M)

Kleinster für die Figur noch nutzbarer Raum.

AE Large (L)

Kleinster Raum für zwei Figur Entities.

AE X Large (XXL)

Luxusraum.

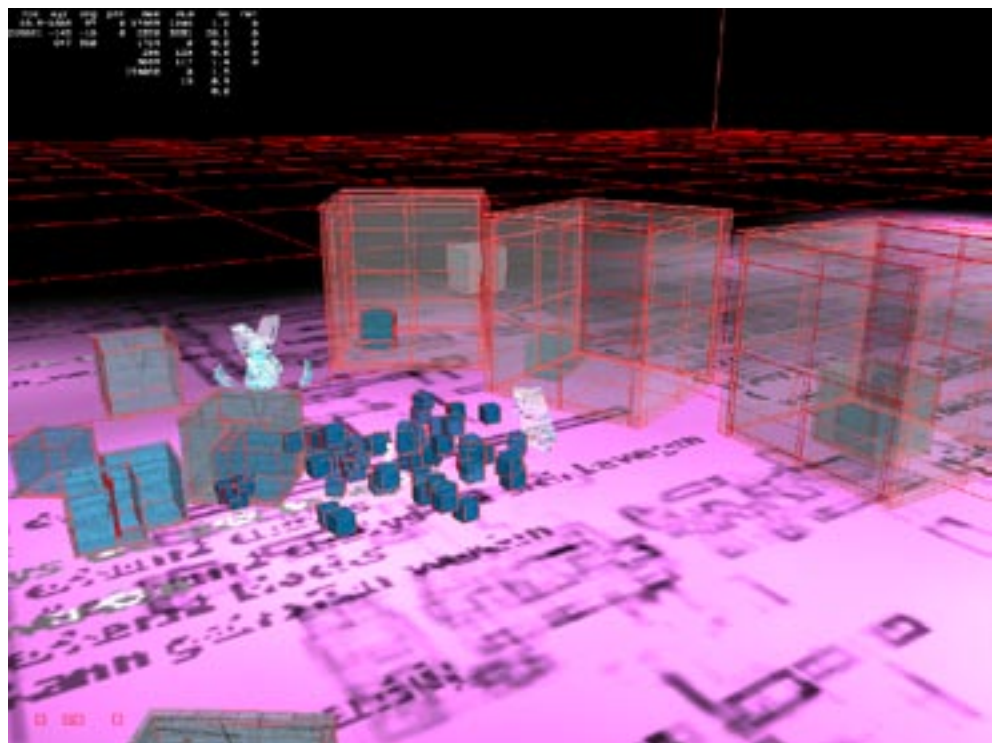


Abb. 56:  
vier unterscheidbare AE's und ihr Grössenverhältnis zur FE

## 5.7 Derived class Architecture

Im Anwendungsverlauf entstehen aus den ursprünglichen AE's (base class oder Elternklasse) neue AE's (derived class oder Kindklasse), die aus der Überlagerung mit anderen AE's entstanden sind. Diese Überlagerung kann aus Architektur Entities her führen aber auch aus Überlagerung mit der Figur Entity.

Hintergrund hierfür ist die Methode des Objektorientierten Programmierens, die es erlaubt Eigenschaften von Objekten in Klassen zu speichern. Solche Eigenschaften sind auch die Punkte (Vertices) die dreidimensionale Objekte bilden. Auf diese Eigenschaften kann man zugreifen und sie neuen Objekten (also auch neuen Entities) zuweisen.



Abb. 57:  
derived class Entity aus Überlagerung AE und FE

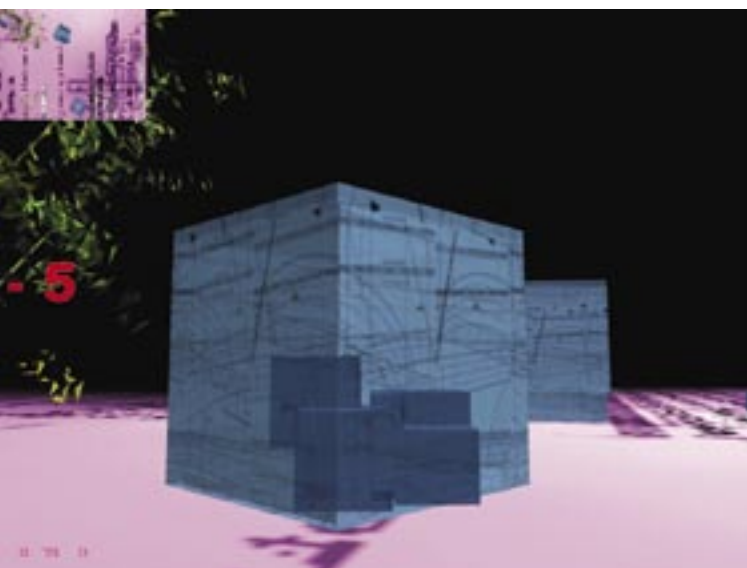


Abb. 58:  
derived class Entity aus Überlagerung AE und AE

## 5.8 Entwickelte Algorithmen - behaviour

### 5.8.1 Code\_1 - Grundverhalten

Base class Funktion: Architektur Entities starten mit diesem Verhaltenscode.

Pseudocode // „auf Figur Entity zu“

```
function code_1 ();  
    finde Figur Entity (FE);  
    drehe dich und bewege dich auf sie zu;  
    sei fähig auf Aktivierung zu reagieren;  
        wenn von FE aktiviert reagiere mit function code_2;  
    wenn in der Nähe von FE, bleibe dort für x sec;  
    danach lösche dich;  
    rufe dich selbst auf und plaziere dich an zufälligen  
    Koordinaten um die FE;  
    gebe dir selbst die function code_1; (Schleife)
```

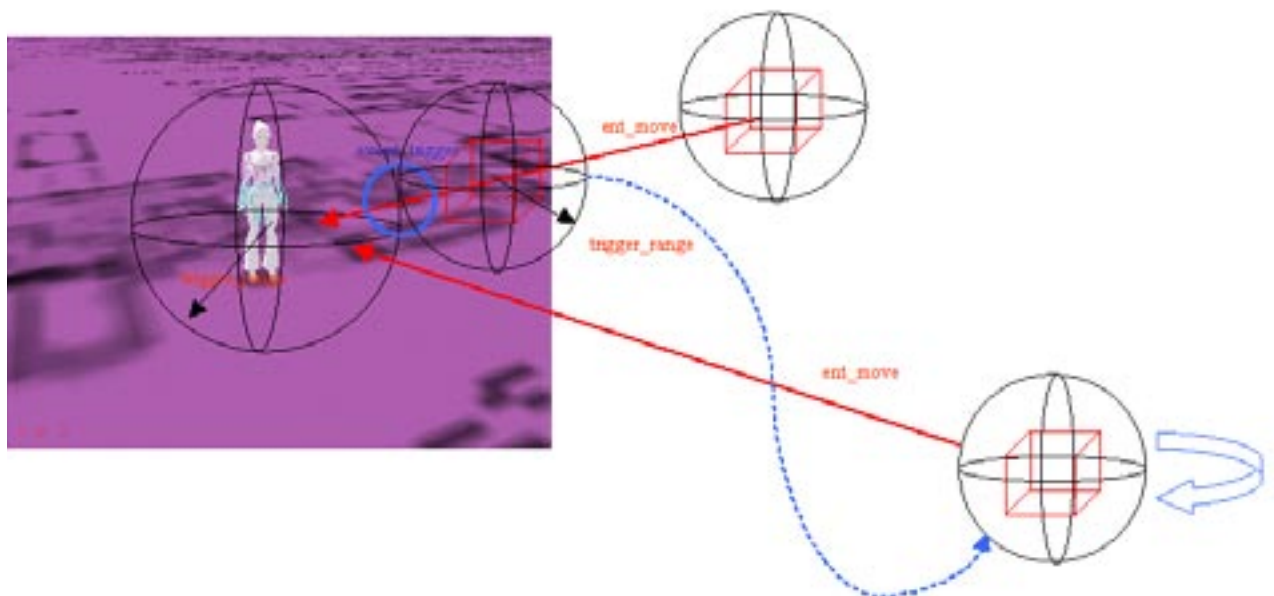


Abb. 59:  
Code\_1 - Grundverhalten





### 5.8.5 Code\_5 - Garten

#### Pseudocode

```
function code_5 ();  
    wird gestartet durch Code_4;  
    Variable bekommt zufällige xyz Koordinaten um FE;  
    Anzahl von x Baum Entities (*.mdl Dateien) werden geladen und bekommen  
    die oben genannten Koordinaten, jede Entity eine andere;  
    nach x sec wird die Baum Entity gelöscht;
```

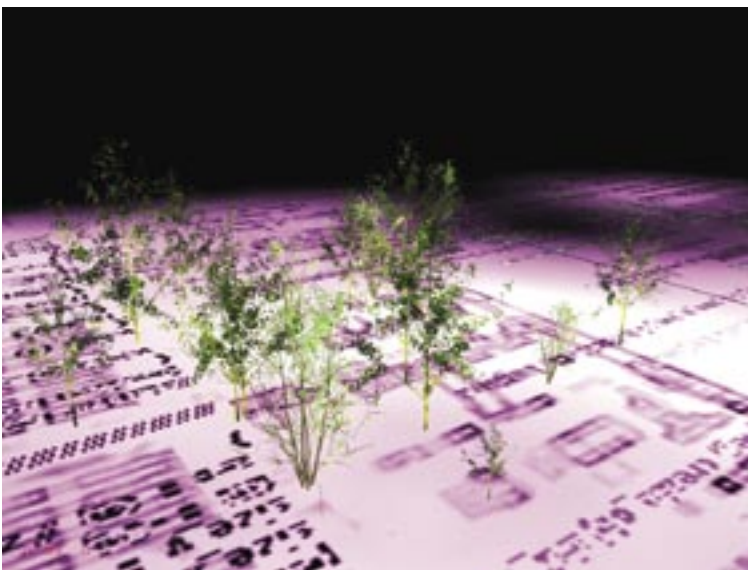


Abb. 64:  
Code\_5 - Garten

### 5.8.6 Code\_6 - Besucher

#### Pseudocode

```
function code_6 ();  
    wird gestartet durch Code_4;  
    Variable bekommt zufällige xyz Koordinaten um FE;  
    Anzahl von x Figur Entities (*.mdl Dateien) werden geladen und  
    bekommen die oben genannten Koordinaten, jede Entity eine andere;  
    die Entity erhält einen einfachen Bewegungskode der sie Hindernisse  
    (Umgebung) erkennen lässt und sie darauf reagieren kann (einfache KI);  
    nach x sec wird die Baum Entity gelöscht;
```

Abb. 65:  
Code\_6 - Besucher



## 5.9 Beziehung Architektur / Funktion

Die ARCHITECTURE\_ENGINE\_1.0 trennt die Architektur Entity von ihrem Code. Dies bedeutet die Architektur wird getrennt von ihrer Funktion, bzw. die Funktion wird der Architektur durch ihren Anwender erst auferlegt. Und sie ist in der Lage mit ihrer Umgebung zu reagieren.

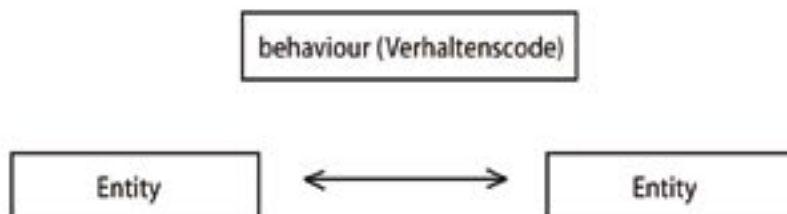


Abb. 66:  
Beziehung Entities untereinander

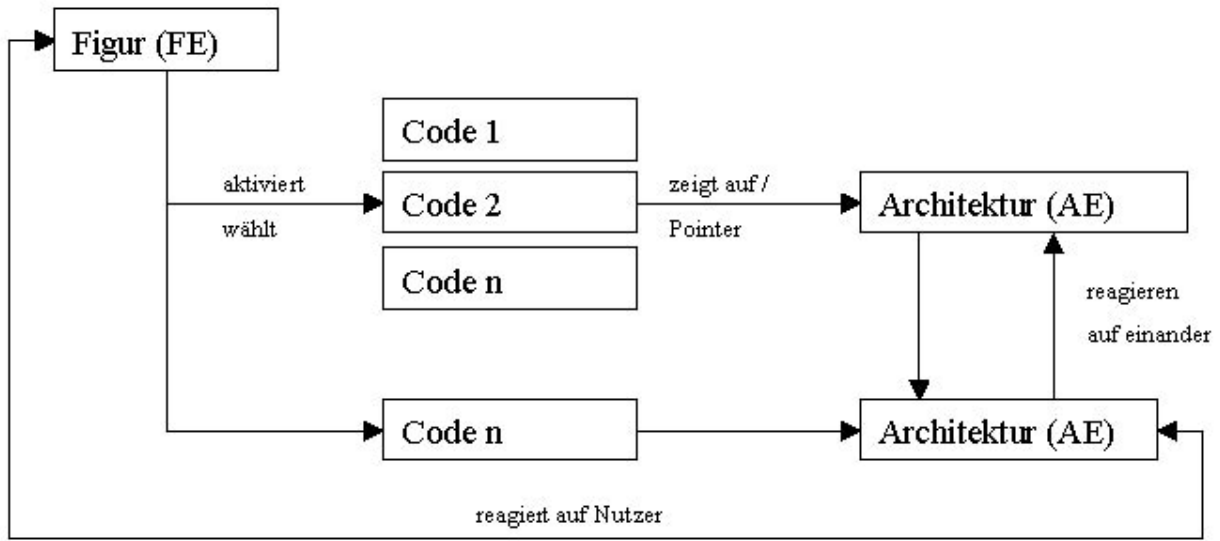


Abb. 67:  
Zusammenhang Architektur und Code

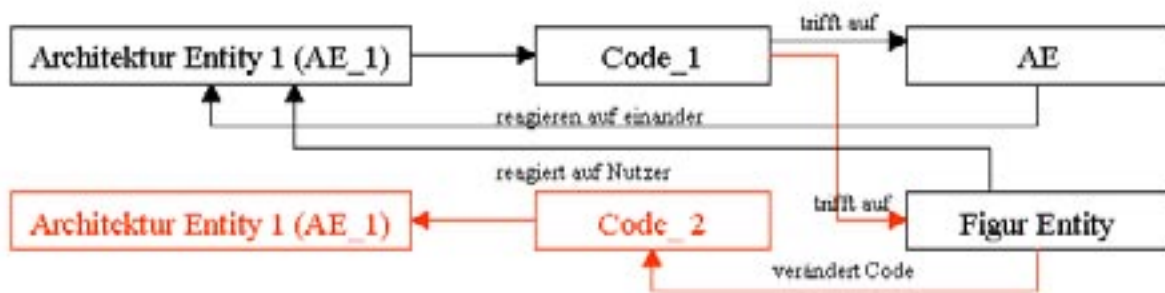


Abb. 68:  
Änderung des eigenen Codes aus der Sicht einer Architektur Entity

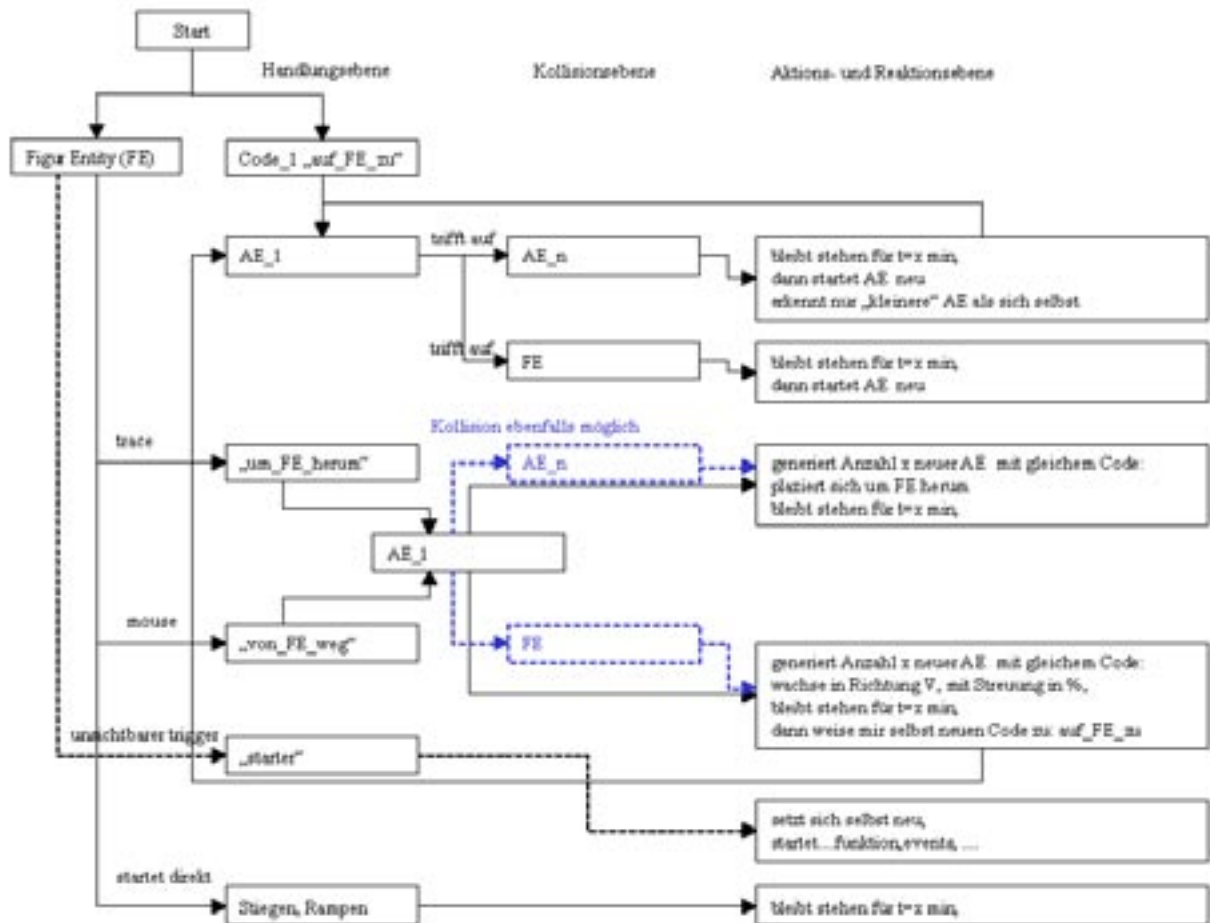


Abb. 69:  
Möglicher Verlauf der Anwendung



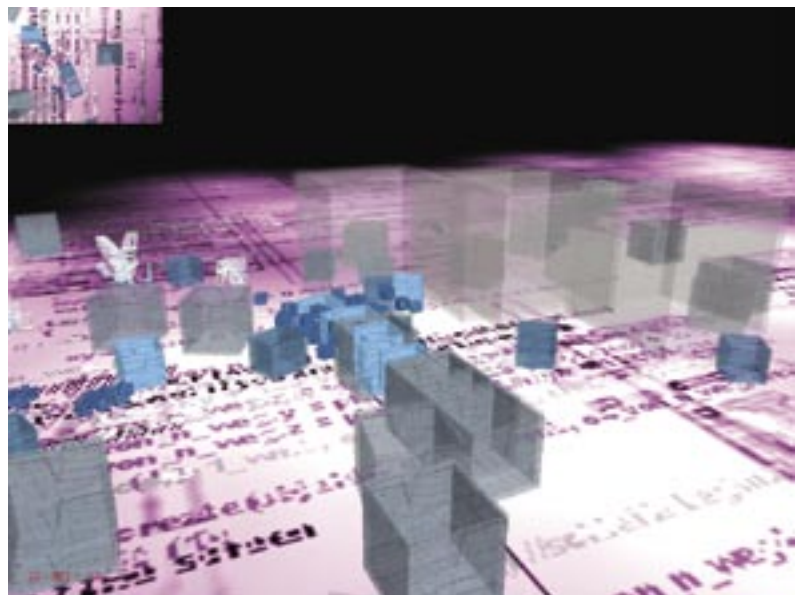
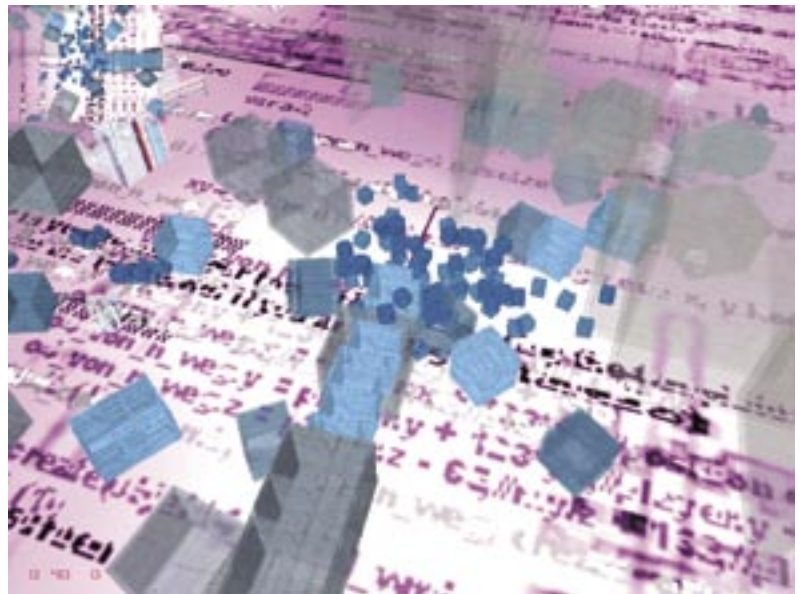
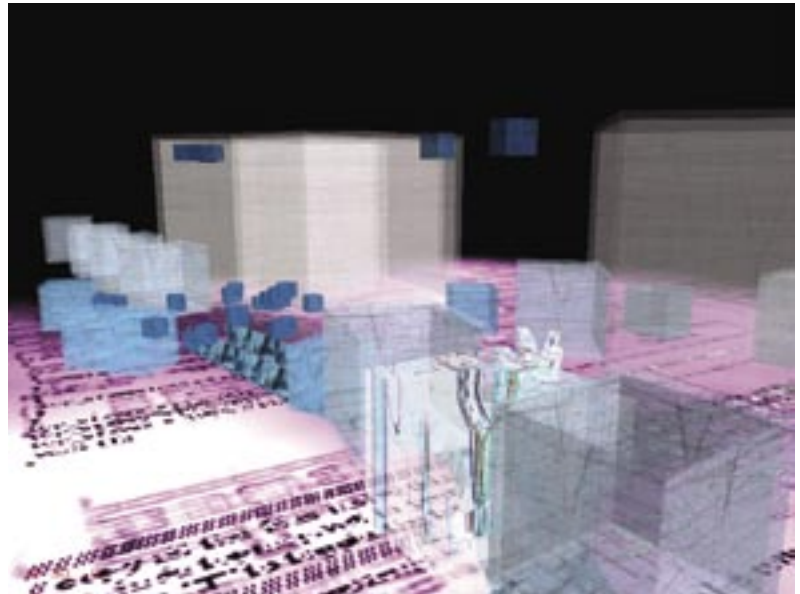


Abb. 72:  
Screenshots aus der  
ARCHITECTURE\_ENGINE\_1.0

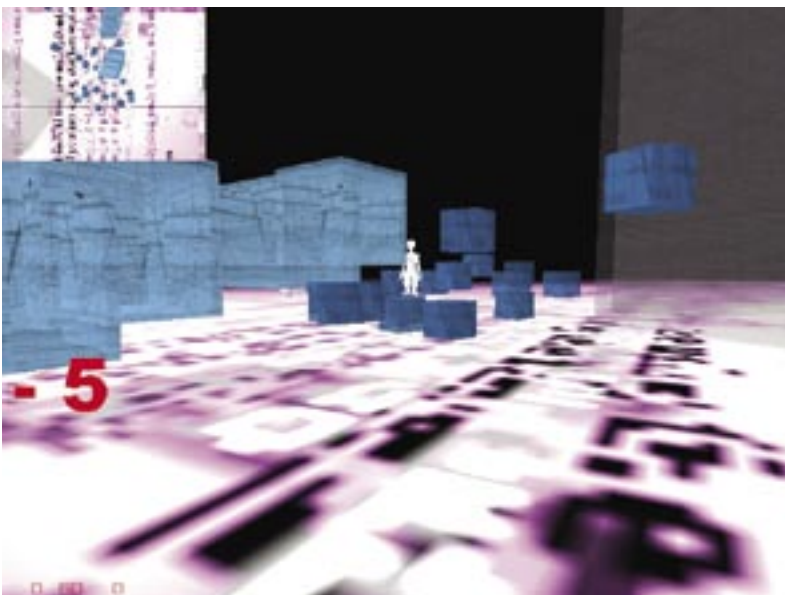
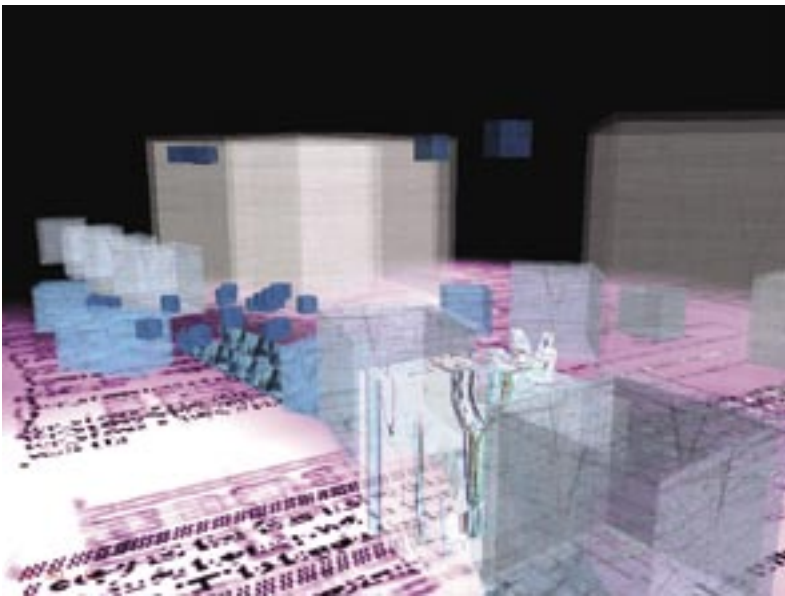
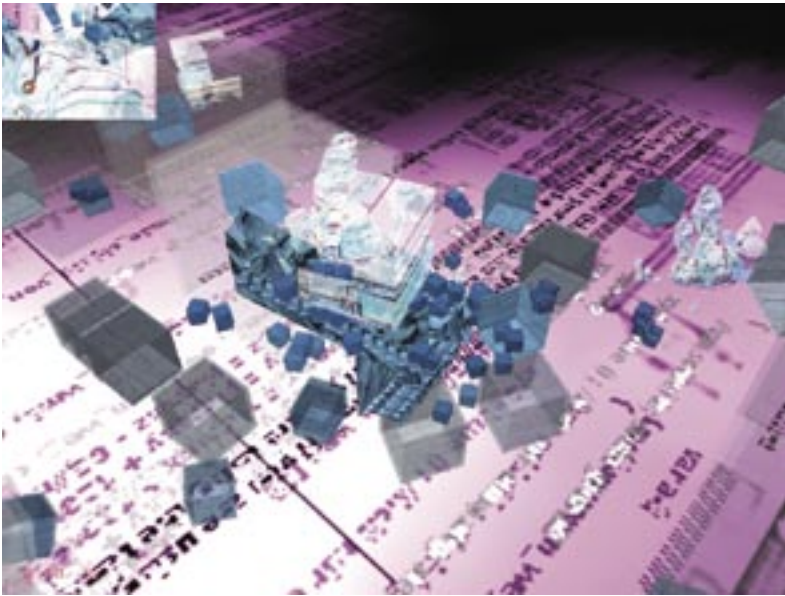


Abb. 73:  
Screenshots aus der  
ARCHITECTURE\_ENGINE\_1.0

## //6\_Zusammenfassung////////////////////////////////////

*„First you must see and feel how beautifully complex the procedures are, and how precisely and intuitively you must think and act to make it work for you. You must become as innocent and tricky as a child playing a game.“* (Oosterhuis 2003: 74)

### 6.1 Fazit

Im Unterschied zu bisherigen Computerspielen wird bei der ARCHITECTURE\_ENGINE\_1.0 die Umgebung, also insbesondere die Architektur, ebenfalls zu einer programmierbaren Entität, welche bei den Computerspielen meist nur die Spieler sind. Daraus ergibt sich, dass die Architektur, die die virtuelle Figur umgibt, mit dieser und mit sich selbst in Kontakt treten kann. Sie erkennt sich und ihre Umgebung und kann darauf reagieren. Die Architektur wird reaktiv.

Weiterhin ist die Architektur (Architektur Entity) und alle anderen entwurfsrelevanten Parameter nun programmierbar und dadurch steuerbar, mit Verhalten und Eigenschaften belegbar. Indem die architektonische räumliche Umgebung in der ARCHITECTURE\_ENGINE\_1.0 ebenfalls zu Entities gemacht wird, wird sie der virtuellen Figur auf der Ebene der Programmierung gleichgestellt. Den Entities kann Code (*behaviour*) per Script zugewiesen werden. Die Bewegungsmöglichkeiten oder die „künstliche“ Schwerkraft der Spielfigur z.B. sind solche Codes. Dieses *behaviour* ist, da es zuordenbar ist, auch während der laufenden Anwendung austauschbar. Das bedeutet die Figur kann den Code einer AE gegen einen andern Code austauschen, sie kann der AE aber nicht während der laufenden Anwendung einen neuen Code geben. Dies kann nur per Programmierung geschehen. Diese Gleichstellung zu Ende gedacht bedeutet nun, dass sich die Figur, die eigentlich den Nutzer der Anwendung im virtuellen Raum darstellen soll, in nichts mehr von der AE unterscheidet. Oder anders herum: wenn der Bewegungscode der FE einer AE zugewiesen wird, ist der Anwender kein Mensch mehr sondern selbst Architektur. (siehe Abb. 74 und 75)

*„Die Architekten werden Formen nicht direkt entwerfen, sondern die Bedingungen und Regeln, nach denen Formen und Verhaltensmuster entstehen. Aber die Unterschiede zwischen Innen und Außen, Nah und Fern, Virtuell und Real, Biologisch und Mechanisch, Gebäude und Körper, Körper und Geist werden schwinden, wenn wir methodisch alles mit allem verbinden.“* (Novak 2001)

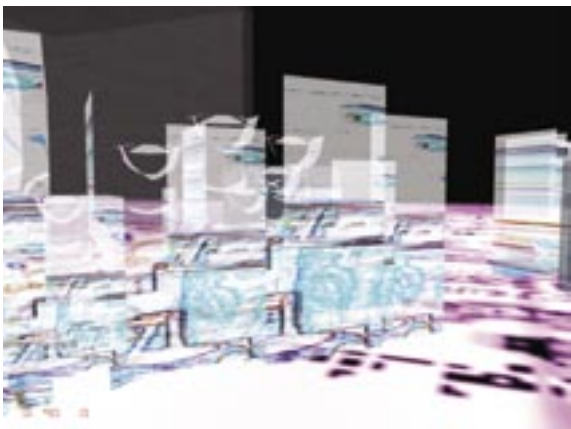


Abb. 74 und 75:  
Vergleich Beziehung Architekt und Architektur 1931 und 2004

## 6.2 Es gibt kein Spielziel. Warum soll man es dann spielen?

Die ARCHITECTURE\_ENGINE ist kein Spiel im klassischen Sinn. Es soll und kann den Spieler nicht lange binden oder gar faszinieren. Es ist eine Anwendung für Architekten. Es ist Planungshilfe, aber keine Maschine die ein fertiges Produkt ausdrückt.

Die Anwendung bietet jedoch dem oben angeführten Gedanken Manovich's und Pias' folgend die Möglichkeit einer Alternative zum narrativen Dialog. Wenn man davon ausgeht, dass die Objekte und Algorithmen eines Entwurfsprozesses Daten sind, so ist die Anwendung eine Interaktion zwischen Nutzer und Datenbank, die so bisher nicht möglich war. Der Nutzer der Anwendung der ARCHITECTURE\_ENGINE\_1.0 „spielt“ den Regeln entsprechend mit den Daten eines Entwurfes und versucht ein möglichst gutes Ergebnis zu erzielen.

Im Unterschied zu Simulationen ist man in der Lage jederzeit in den Prozess einzugreifen, ihn zu manipulieren. Dabei bietet die Technologie alle Möglichkeiten der bisherigen Planungstechniken.

## 6.3 Visuelle Gestaltung

Ein wichtiger Bereich ist die visuelle Gestaltung. Es gibt den Bereich der Bildschirmdarstellung und den Bereich der Gestaltung der virtuellen Welt. Die dreidimensionalen Objekte in einer Anwendung sind mit einer Textur zu belegen. Diese sollte am besten in einem thematischen Zusammenhang zum Konzept der Anwendung gewählt werden. Es ist zu beobachten, dass die Grafik eine wichtige Rolle spielt bei der Akzeptanz der Anwendung. Zur Zeit gibt es eine Tendenz hin zu realistischen Texturen, während abstrakte Texturgebungen eher auf Ablehnung in der Akzeptanz stoßen.

In der ARCHITECTURE\_ENGINE\_1.0 wurde, da es sich um eine abstrakte Darstellung handelt, jedoch versucht dies auch entsprechend umzusetzen. Bei konkreten architektonischen Bauaufgaben fällt die Texturbelegung sicherlich leichter.

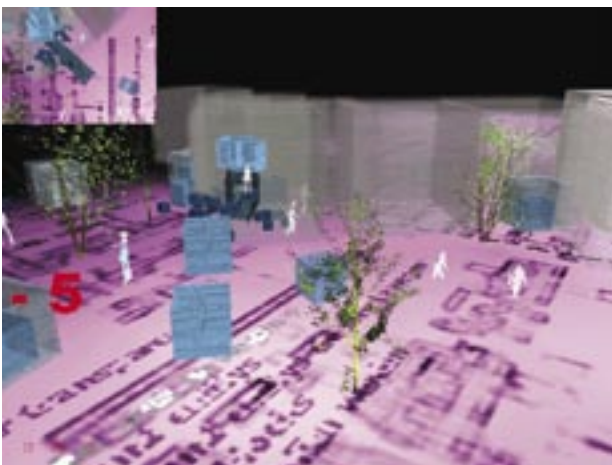


Abb. 76:  
Screenshot ARCHITECTURE\_ENGINE



Abb. 77:  
Screenshot CNN

Ebenfalls wichtig ist die Atmosphäre einer virtuellen Szene. Dabei spielt Licht und Schatten eine große Rolle.

Die Bildschirmdarstellung zeigt dem Anwender die gewählte Perspektive (View), den Punktstand, aktuelle Texteinblendungen (Panels), zusätzliche Views und die Menüführung. Man spricht von Multimedialität, da sich mehrere Informationsebenen überlagern. Man vergleiche hierzu aktuelle Nachrichtensender wie CNN oder NTV, die diese Multimedialität auch benutzen. (siehe Abb. 77)

(Anm. d. Verf.: Laut amerikanischen Studien sind Jugendliche vermehrt dazu in der Lage, sich mit dieser Multimedialität gut zu arrangieren. Während ältere Menschen sich überfordert fühlen, sind Jugendliche in der Lage selektiv ihre Wahrnehmung zu lenken, ohne dabei in ein Gefühl des Stress zu geraten.)

#### 6.4 Zugriffsmöglichkeiten

Die Möglichkeiten der Interaktion des Nutzers per Interface auf den entstehenden Prozess einzuwirken sind im Unterschied zu den meisten Simulationen erweitert, da der Nutzer selbst als Entity am Prozess teilnimmt. Jedoch gibt es auch hier strukturbedingte Einschränkungen.

Zunächst einmal muss die Interaktion in der Programmierung schon mitbedacht werden. Das bedeutet man kann nur das tun was per Code auch möglich ist. Dies gilt für Aktionen per Entity (also in der virtuellen Welt) und auch für Interaktionen per Tastatur.

Noch wichtiger aber ist die durch die Programmierungsebene bedingte Einschränkung. Man kann behaviour nicht generell ändern, man kann es nur austauschen. So hat man während einer laufenden Anwendung nur die Möglichkeit auf Code bedingte Variablen einzuwirken, also sie zu verändern, jedoch nicht auf den Funktionscode selbst.

(Auf der Spielebene ist es eher einleuchtend wenn man von Spielregeln spricht, die nicht gebrochen werden dürfen. So kann man zwar falsch spielen, da dies ja noch regelbestimmt ist, aber man darf die Regeln nicht während dem Spiel ändern.)

Die Veränderung solcher Code bedingten Variablen könnten z.B. Umgebungsparameter für einen Entwurf sein, wie Zeit- oder Ressourcenmanagement, Bebauungshöhe oder Bebauungsdichte.

#### 6.5 Interface

Ein weiterer wichtiger Aspekt ist die Qualität der Teilnahme des Nutzers an solchen Entwurfsprozessen. Er kann vor einem Bildschirm sitzen oder über eine 3D Datenbrille teilnehmen, der Computer könnte per Kamera oder Sensor persönliche Zustände erkennen und diese mit in die Prozesse einbeziehen. Laut Ernest W. Adams in einem Interview liegt die Zukunft der Computerspiele in neuen Mensch Maschinen Interfaces.

*„Interfaces sind mehr als nur Schnittstellen; sie sind interaktive Kopplungen von Menschen und/oder Maschinen, mit denen ständig sich verändernde Problemstellungen bewältigt werden. Die Interfaces integrieren als temporäre Vernetzungen nicht nur Menschen, sondern auch Intelligente Agenten.“*

(Schmidt 1999)

## 6.6 Programmiererebene

Das Arbeiten mit Programmiersprachen erfordert die Auseinandersetzung mit einer für Architekten meist neuen Sprache. Syntax und Semantik müssen erst erlernt werden. Der Bereich der Informatik ist zwar faszinierend, jedoch auch oft zum verzweifeln logisch. Zu Beginn müssen Ziele der architektonischen Aufgabe definiert werden. Dazu müssen Entwurfsparameter abstrahiert werden um dann in Algorithmen übersetzt werden zu können. Gerade die Arbeit des Code Schreibens ist sehr zeitintensiv und null fehlertolerant.

Aber nicht nur die Entwurfsmethoden von Architekten ändern sich, auch die Grundzüge der unterschiedlichen Programmiersprachen wie Objektorientierte Programmierung über Aspektorientierte bis hin zu Generativen Programmierung unterliegen strukturellen Veränderungen. (vgl.: Veit 2001)

## 6.7 „Zuviel“ Architektur

Das Arbeiten mit algorithmischen Codes, die Architektur entstehen lassen hat, im Gegensatz zum Entwerfen vor dem weisen Blatt Papier, nicht das Problem der Leere, sondern eh er das Problem des „Zuviel“. Es entstehen erst einmal „ohne Ende“ Ergebnisse, sodass immer auch an die (vorzeitige) Beendigung oder Unterbrechung, zumindest aber Eingrenzung des Prozesses gedacht werden muss. So ist das Wegnehmen, das Entfernen mindestens genauso wichtig wie das Entstehen lassen.

Auch die Technologie der Game Engine, so schnell sie auch zur Zeit ist, haben das Problem nur eine begrenzte Anzahl von Entitäten verarbeiten zu können. Ansonsten verlangsamt sich die *framerate* und die Anwendung kommt zum Erliegen. Die Technologie fordert eine leistungsstarke Hardware.

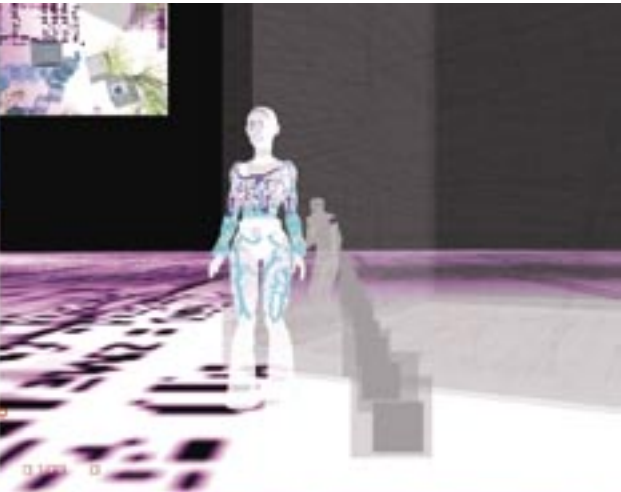
```
my_enable_impact = on // erkennt obj's die sich bewegen
my_enable_block = on // erkennt blocks
my_enable_choot = on //achn getroffen werden

my_obj = objekt_von_n_weg_event

sleep(0.3)
vec_set(a1_von_n_weg_neu, nullvector);

////////////////////////////////////
var a1;
wait(1);
randomize();
a1 = int(random(9)) //generiert zahl zwischen 0 und 9

//objekt 1 ist ca 40x40x40 groß
////////////////////////////////////
if (a1 == 8)
```



```
o1_von_n_weg_neu.x = my.x + 30 * x1/block_vec.x + 64;
o1_von_n_weg_neu.y = my.y + 30 * y1/block_vec.y;
o1_von_n_weg_neu.z = my.z + 20/block_vec.z;

if (a1 == 7)
{
o1_von_n_weg_neu.x = my.x + 30 * x1/block_vec.x + 64;
o1_von_n_weg_neu.y = my.y + 30 * y1/block_vec.y;
o1_von_n_weg_neu.z = my.z + 5/block_vec.z;
}

if (a1 == 6)
{
o1_von_n_weg_neu.x = my.x + 30 * x1/block_vec.x + 64;
o1_von_n_weg_neu.y = my.y + 5/block_vec.y;
o1_von_n_weg_neu.z = my.z + 5/block_vec.z;
}
```



```
o1_von_n_weg_neu.x = my.x + 30 * x1/block_vec.x + 64;
o1_von_n_weg_neu.y = my.y + 30 * y1/block_vec.y;
o1_von_n_weg_neu.z = my.z + 0/block_vec.z;

if (a1 == 4)
{
o1_von_n_weg_neu.x = my.x + 30 * x1/block_vec.x + 64;
o1_von_n_weg_neu.y = my.y + 30 * y1/block_vec.y;
o1_von_n_weg_neu.z = my.z + 20/block_vec.z;
}

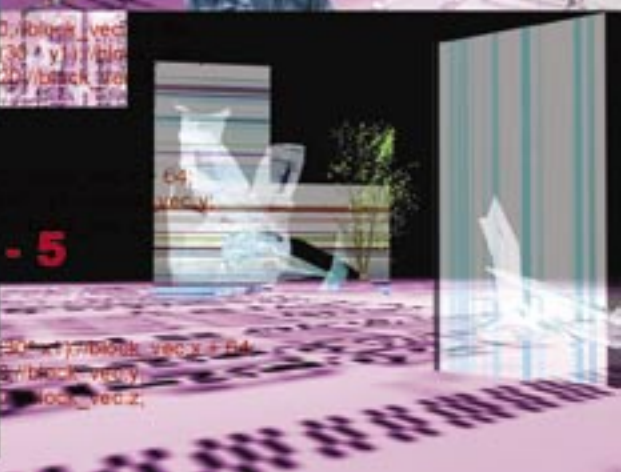
if (a1 == 5)
else/
```



```
o1_von_n_weg_neu.x = my.x + 30 * x1/block_vec.x + 64;
o1_von_n_weg_neu.y = my.y + 30 * y1/block_vec.y;
o1_von_n_weg_neu.z = my.z + 0/block_vec.z;

if (a1 == 1)
{
o1_von_n_weg_neu.x = my.x + 30 * x1/block_vec.x + 64;
o1_von_n_weg_neu.y = my.y + 30 * y1/block_vec.y;
o1_von_n_weg_neu.z = my.z + 0/block_vec.z;
}

if (a1 == 0)
{
o1_von_n_weg_neu.x = my.x + 30 * x1/block_vec.x + 64;
o1_von_n_weg_neu.y = my.y + 30 * y1/block_vec.y;
o1_von_n_weg_neu.z = my.z + 0/block_vec.z;
}
```



# //Abstract////////////////////////////////////

Diese Arbeit untersucht generelle Anwendungsmöglichkeiten für programmierbare Game Engines im Entwurfsprozess.

Hierfür wird zunächst der theoretische Kontext zum Themengebiet des Computerspiels abgegrenzt, dann ein kurzer Überblick über die Geschichte der Computerspiele gegeben, und schließlich eine Einführung in die Technologie der Game Engines sowie die damit verbundenen aktuellen Programmiersprachen.

Darauf aufbauend wird versucht zu zeigen, wie die Technologie einer programmierbaren Game Engine in architektonischen Entwurfsprozessen genutzt werden könnte, wobei sich die Arbeit eher als Annäherung an konkrete Anwendungen versteht und nicht als fertiges Entwurfswerkzeug.

Bei der Abstrahierung von entwurfsrelevanten Prozessen in programmierbare algorithmische Verhaltensregeln (behaviour), die dann den Entities zugewiesen werden, kann zwischen den Umgebungsbedingungen eines architektonischen Entwurfprozesses, wie Zeit-, Ressourcenmanagement oder den Bebauungsbestimmungen und aufgabenrelevanten Zielen, wie dem Transfer eines Raumprogramms oder Nutzungsablaufs, unterschieden werden.

Diese Entities interagieren dann entsprechend ihrer "behaviour" in einem dreidimensionalen virtuellen Raum. Es entstehen reaktive dreidimensionale virtuelle Architekturen, die im Unterschied zu Simulationen während des laufenden Prozesses vom Anwender interaktiv beeinflusst und erlebt werden können. Zuletzt wird eine solche Anwendung (Architecture\_Engine\_1.0) programmiert und beschrieben, die die Möglichkeiten einer programmierbaren Game Engine demonstriert.

The topic of this thesis is a research about the application possibilities for game engines in the architectural design process. Therefore a short theoretical and historical overview of computer games, an introduction to game engines, and a definition of recent scripts is given.

The aim is to show how a game engine can be used in architectural design processes. However, this thesis is meant to be an approach and not a ready-to-use tool. The abstraction of the relevant parameters informing design processes is very important in this context and can be divided into two parts:

First, environmental factors like resource- and time-management, planning regulations and budget. Second, specific design tasks like the abstraction of a building program and sequences of use. The result of these abstraction processes are algorithms (behaviour) which can then be assigned to entities. These entities interact in a three dimensional virtual reality. The result is a reactive three dimensional virtual architecture. The user can interact in runtime, which creates an important difference compared to the usual simulations.

The game software "architecture\_engine\_1.0", which has been developed based on the research mentioned above, demonstrates possible applications.

# //Glossar////////////////////////////////////

## Augmented Reality

erweiterte Realität, ist eine Form der Mensch Maschine Interaktion. Per Datenbrille werden kontextabhängige Informationen eingeblendet. Dies bedeutet die „Brille“ weist wo sie ist und erweitert das Blickfeld.

## Avatar

Ursprünglich stammt der Begriff aus der Hindu Mythologie und stand für den Körper dessen sich eine Gottheit bedient um die Erde zu besuchen. Heute wird der Begriff für ein virtuelles Modell in Spielen benutzt. (Spielfigur)

## behaviour

siehe Script / Code

## corridor games

siehe Container Räume

## Container Räume

ist die Bezeichnung für die Raumstruktur von Ego Shooter Spielen.

## Entity

Die Entität (lat. ens : das Sein) oder Seinshaftigkeit (auch Wesenheit) (englisch: entity) bezeichnet ein individuelles Einzelement aus einer Vielzahl von möglichen Elementen der realen oder der Vorstellungswelt. Entität reflektiert die „Seiendheit“ eines Dings, mit der Betonung darauf, „dass“ es ist, im Unterschied davon „was“ es ist. In der klassischen Philosophie bezeichnet Entität alles, was überhaupt existiert und worüber etwas ausgesagt werden kann. Es wird dabei völlig davon abstrahiert, ob dieses Existierende materiell oder ideell, objektiv oder subjektiv existiert.“

„Als Entität werden in der Informatik unterscheidbare, in der realen Welt eindeutig identifizierbare Objekte bezeichnet. Die Objekte können sowohl eine physische („real“) oder eine konzeptionelle („abstrakte“) Existenz haben.

## Funktion

in der Informatik:

Eine Funktion in der Programmierung von Computersystemen ist ein Stück zusammengehörenden Codes, der es erlaubt, eine bestimmte Aufgabe in wiederverwendbarer Art umzusetzen. Funktionen können einen oder mehrere Aufrufparameter haben und liefern nach ihrem Ende (im Gegensatz zu Prozeduren) einen Funktionswert zurück. Funktionen werden typischerweise in Bibliotheken thematisch gebündelt.

## framerate

auch Bildwiederholungsfrequenz. Gibt an wie viele Bilder pro Sekunde auf dem Bildschirm angezeigt werden. Ab 25 Bildern pro Sekunde erscheint das Angezeigte als flüssig. Die framerate ist stark hardware abhängig. Ansonsten kommt es zu frame drops, dem sogenannten Ruckeln des Bildes.

## Game Engine

koordiniert sämtliche Elemente die das Spiel ausmachen, wie Grafik, Sound, Steuerung, KI, usw.

## game play

fast nicht zu beschreibender Bewertungsmaßstab eines Spiels. Ist das game play gut, wird das Spiel gerne gespielt.

## MMORPG

Abkürzung für Massive Multiplayer Online Role Play Games.

## Nullraum

ein vom Verfasser eingeführter Begriff für die maßgebende Umgebung (Container), die die 3D Game Engine mindestens benötigt um die Grenzen dessen was gerendert wird festzulegen.

## Pointer

in der Informatik eine Variable, die auf eine Speicheradresse zeigt.

## Reaktive Systeme

sind Computer-Systeme, die in ständiger Interaktion mit ihrer Umgebung stehen, indem sie ihre Umgebung erkennen können und die Möglichkeit haben darauf zu reagieren.

## Script / Code

ist ein in einer Programmiersprache (z.B. C++ ) geschriebener Algorithmus. Bei Game Engines nennt man einen Code, der einer Entity zugewiesen ist behaviour.

# //Literaturverzeichnis////////////////////////////////////

- Adams, Ernest (1999): Three Problems for Interactive Storytellers, [http://www.gamasutra.com/features/designers\\_notebook/19991229.htm](http://www.gamasutra.com/features/designers_notebook/19991229.htm) (13.09.2004).
- Bruce, Thomas (2002): About The ARQuake Project , <http://wearables.unisa.edu.au/projects/ARQuake/www/> (14.09.2004).
- Charms (2003): Zwischenfälle. Luchterhand Literaturverlag, München.
- Costikyan, Greg (1994): I Have No Words & I Must Design, [http://www.costik.com/nowords.html#Not\\_puzzle](http://www.costik.com/nowords.html#Not_puzzle), (11.02.2004).
- Costikyan, Greg (2000): Where Stories End and Games Begin, <http://www.costik.com/gamnstry.html> (13.09.2004).
- Eigen, Manfred und Winkler, Ruth (1990): Das Spiel: Naturgesetze steuern den Zufall, München, Piper Verlag.
- Friedman, Ted (1999): Civilization and its Discontents: Simulation, Subjectivity and Space. <http://www.duke.edu/~tlove/germ/writing.htm> (28.03.2004).
- Fritz; Jürgen (o.A.): Wie virtuelle Welten wirken , <http://www.bpb.de/snp/index.html>, (11.09.2004).
- Fuller, Mary und Jenkins, Henry (1995): Nintendo® and New World Travel Writing: A Dialogue, [http://www.stanford.edu/class/history34q/readings/Cyberspace/FullerJenkins\\_Nintendo.html](http://www.stanford.edu/class/history34q/readings/Cyberspace/FullerJenkins_Nintendo.html) (02.06.2004).
- Funken, Christiane und Löw, Martina (2001): Ego-Shooters Container, Raumkonstruktionen im elektronischen Netz, Raum Wissen Macht, Frankfurt /M, Suhrkamp.
- Huizinga, Johan (1938=1956): Homo Ludens, 18. Auflage, Hamburg, Rowohlt Taschenbuch Verlag.
- Klages-Conti (2001), Viele, viele Spiele. <http://www.stud.uni-hannover.de/user/69332/texte/spiel.html#t4> (24.05.2004).
- Knauer (2001): Computerspiele-Industrie auf der Überholspur. <http://derstandard.at/?id=1628030> (26.05.2004).
- Kücklich, Julian (2002): Computerspielphilologie – Prolegomena zu einer literaturwissenschaftlich begründeten Theorie narrativer Spiele in den elektronischen Medien, Magisterarbeit, Universität München.
- Kuhn, Thomas (1967): Die Struktur wissenschaftlicher Revolution. Frankfurt/M, Suhrkamp.
- Lischka, Konrad (2002): Spielplatz Computer, 1. Auflage, Hannover, Verlag Heinz Heise GmbH & Co KG.
- Mactavish, Andrew (o.A.): Game Mod(ifying) Theory: The Cultural Contradictions of Computer Game Modding, [http://www.milgramreenactment.org/powerup/pages/prog\\_index.html](http://www.milgramreenactment.org/powerup/pages/prog_index.html), (28.03.2004).
- Manovich, Lev (2002): The Language of New Media. Cambridge, MA; The MIT Press.
- Martinez ( 2001): Über Schillers Begriff des Spieltriebs. <http://www.stud.uni-hannover.de/user/69332/texte/spiel.html#t2> (24.05.2004).
- netlexikon: Computerspiel - Definition, Bedeutung, Erklärung im Lexikon, <http://www.lexikon-definition.de/Computerspiel.html>, (25.09.2004).
- Newman, Beverly (o.A.): Forbidden Pleasures - Cheating in Computer Games, <http://www.watershed.co.uk> (28.03.2004).
- Novak, Marcos (2001): Fließend~, Trans~, Unsichtbar~: Der Aufstieg und die Artentstehung des Digitalen in der Architektur. Eine Geschichte, [http://www.a-matter.de/digital-real/dr\\_flash\\_final.swf](http://www.a-matter.de/digital-real/dr_flash_final.swf), (11.10.2003)
- Oosterhuis, Kas (2003): Hyper Bodies, Basel, Birkhäuser.
- Pias, Claus (2000): Computer-Spiel-WeltenComputer-Spiel-Welten, <http://www.uni-weimar.de/medien/kuenstliche-welten/forschung/forschung.htm>, (22.05.2004).
- Rorty, Richard (1994): Hoffnung statt Erkenntnis: Eine Einführung in die pragmatische Philosophie, Wien, Passagen-Ver.

- Schiller, Friedrich (1795): Über die ästhetische Erziehung des Menschen. Wohlhardt Henchmann. Fink, München.
- Schmidt, Artur P. (1999): Auf dem Weg zur digitalen Zivilisation - Von der virtuellen Architektur, über das Endo-Valley, hin zur Vision Lamspacus - [www.wissensnavigator.com/download/Schmidt\\_Auf%20dem%20Weg%20zur%20digitalen%20Zivilisation.pdf](http://www.wissensnavigator.com/download/Schmidt_Auf%20dem%20Weg%20zur%20digitalen%20Zivilisation.pdf), (14.08.2003).
- Sitarski, Piotr (o.A.): Why Do I Play and Cannot Stop? Pleasures of Computer Games, [http://www.milgramreenactment.org/powerup/pages/prog\\_index.html](http://www.milgramreenactment.org/powerup/pages/prog_index.html) (28.03.2004).
- Thiessen, Peter (o.A.): Klassische und moderne Spieltheorien, <http://www.soziales.fh-dortmund.de/asf/>, (22.05.2004).
- Veit, Matthias (2001): Seminararbeit Generative Sprachen - Komponentenkonzepte für das Continuous Engineering, Technischen Universität Berlin.
- Walz, Steffen (o.A.): Computerspiele als wissenschaftliches Paradigma, [http://www.game-face.de/article.php3?id\\_article=16](http://www.game-face.de/article.php3?id_article=16), (05.09.2004).
- Wikipedia: Entity, [http://de.wikipedia.org/wiki/Entit%C3%A4t\\_%28Philosophie%29](http://de.wikipedia.org/wiki/Entit%C3%A4t_%28Philosophie%29), (7.09.2004).
- Wikipedia, Spieltheorie, <http://de.wikipedia.org/wiki/Spieltheorie>, (26.05.2004).
- Wittgenstein, Ludwig (1951): Philosophische Untersuchungen. Schriften I.Ed. Rush Rees. Frankfurt a.M., Suhrkamp.
- Wolf, Mark J.P (2001): The Medium of the Video Game , University of Texas Press.

# //Abbildungsverzeichnis////////////////////////////////////

Alle nicht genannten Abbildungen stammen vom Autor.

Abbildung 1:

Tennis for two 1958, [www.pong-story.com/ 1958.htm](http://www.pong-story.com/1958.htm), (03.09.2004).

Abbildung 2:

Spaceware 1962, [www.ultimate.com/ phil/xy/spacewar.jpg](http://www.ultimate.com/phil/xy/spacewar.jpg), (03.09.2004).

Abbildung 3:

Pong 1972, [www.pong-story.com/ 1958.htm](http://www.pong-story.com/1958.htm), (03.10.2004).

Abbildung 4:

Doom 1993, <http://www.boilingpoint.com/~jasonyu/cs240/images/doom.jpg>, (03.10.2004).

Abbildung 5:

Star Raider 1979, <http://www.pong-story.com/>, (03.10.2004).

Abbildung 6:

Atari Konsole 2004, <http://www.atari.com>, (03.10.2004)

Abbildung 7:

Sony Eyetoy 2004, <http://www.sony.com>, (03.10.2004)

Abbildung 8,9 und 10:

ARQuake 2002, <http://wearables.unisa.edu.au/projects/ARQuake/www/> (14.09.2004).

Abbildung 13:

Ultima Online, (Autor nicht angegeben): Schöne neue Welten, GEE 07/2004, S. 59.

Abbildung 24:

machinima artistry, [www.oreilly.com/ catalog/1932111859/](http://www.oreilly.com/catalog/1932111859/), (03.10.2004).

Abbildung 33:

sprite entity, [www.thewall.de/tutorial. php?action=view&id=148](http://www.thewall.de/tutorial.php?action=view&id=148), (03.10.2004).

Abbildung 34:

terrain entity, [www.peacekeeper.com/ 3dgs/terrain.jpg](http://www.peacekeeper.com/3dgs/terrain.jpg), (02.10.2004).

Abbildung 75:

Rem Koolhaas (1978): Delirious New York. 2. Auflage 2002, Aachen, S.128.

# //Ludographie////////////////////////////////////

Adventure (Crowther and Woods 1976)  
Black & White (Lionhead 2001)  
Counterstrike (Mod zu Half-Life [1999])  
Civilization (Microprose 1992)  
Deus Ex (Ion Storm 2001)  
Donkey Kong (Nintendo 1981)  
Doom (id Software 1993)  
Enter the Matrix (Warner Bros 2003)  
Everquest (Ubisoft 1999)  
Final Fantasy I-XII (Squaresoft 1987-2002)  
Flight Simulator (Microsoft 1983)  
Game of Life (Conway 1970)  
Grand Theft Auto III (Rockstar Games 2001)  
HalfLife (Valve 1998)  
Max Payne (Remedy 2001)  
Metal Gear Solid (Konami 1998)  
Myst (Brøderbund 1993)  
Pac Man (Nintendo 1980)  
Pong (Atari 1972)  
Popolous (Krisalis 1992)  
Quake (id Software 1996)  
Quake III (id Software 1999)  
SimCity (Maxis 1989)  
Space Invaders (Taito 1977)  
Space Quest II (Sierra 1987)  
Spacewar (Russell 1962)  
Spider Man 2 (Activision 2004)  
SUPER MARIO BROS (Nintendo 1990)  
Tennis for two (Higinbotham 1958)  
Tetris (Nintendo 1989)  
Tomb Raider (Eidos 1996)  
Ultima Online (Origin 1997)  
Unreal (Epic 1998)  
Unreal Tournament (Epic 1999)  
Quake (Nintendo 1996)

# //Literaturhinweise////////////////////////////////////

Adams, Ernest (2001): Dogma 2001: A Challenge to Game Designers  
[http://www.gamasutra.com/features/20010129/adams\\_02.htm](http://www.gamasutra.com/features/20010129/adams_02.htm), (22.05.2004).

Adams, Ernest (o.A.): Gamedesign-Theorien, [http://www.game-face.de/article.php3?id\\_article=21](http://www.game-face.de/article.php3?id_article=21), (20.07.2004).

Allen, Stan (o.A.): Allgemeiner Instrumentalismus, 156 ARCH+, [http://www.baunetz.de/arch/archplus/156/allen\\_c.htm](http://www.baunetz.de/arch/archplus/156/allen_c.htm), (20.07.2004).

Atzler, Elisabeth (o.A.): Im Bann des Virtuellen, [http://www.zeit.de/2001/15/Hochschule/200115\\_c-gamesacademy.html](http://www.zeit.de/2001/15/Hochschule/200115_c-gamesacademy.html), (22.05.2004).

Baumgärtel, Tilman (2004): Alle Nazis werden Dreiecke - Computerspiele verwandeln sich in hintersinnige Kunstwerke, [http://www.zeit.de/2002/16/Kultur/print\\_200216\\_computerspielkun.html](http://www.zeit.de/2002/16/Kultur/print_200216_computerspielkun.html), (22.05.2004).

Baumgärtel, Tilman (o.A.): Und weg waren Sie - Computerspiele erzählen vom Seelenzustand unserer Gesellschaft, sie sind historische Zeugnisse. Archiviert aber werden sie nur selten, <http://www.zeit.de/archiv/2002/52/Spiele-Archiv>, (22.05.2004).

Bölte, Sven; Uhlig, Nora; Poustka, Fritz (o.A.): Das Savant Syndrom: Eine Übersicht, <http://www.psycontent.com/abstracts/hh/zkp/2002/04/body-zkp3104291.html>, (20.07.2004).

Bodmer, Marc (o.A.): Spielplatz der Zerstörung - Spielindustrie zeigte sich an der Messe E3 wenig innovativ, <http://www.nzz.ch/2004/05/21/em/page-article9LT5T.html>, (22.05.2004).

Chu, Karl S. (o.A.): Modal Space - The virtual anatomy of hyperstructures, [http://www.azw.at/otherprojects/soft\\_structures/allgemein/modal\\_space.htm](http://www.azw.at/otherprojects/soft_structures/allgemein/modal_space.htm), (20.07.2004).

Chu, Karl S. (o.A.): The Turing Dimension, [http://synworld.t0.or.at/level3/text\\_archive/the\\_turing.htm](http://synworld.t0.or.at/level3/text_archive/the_turing.htm), (20.07.2004).

Chu, Karl S. (o.A.): The Cone of Immanenscendence, [http://www.azw.at/otherprojects/soft\\_structures/allgemein/the\\_cone.htm](http://www.azw.at/otherprojects/soft_structures/allgemein/the_cone.htm), (20.07.2004).

Constant (o.A.): New Babylon: the world of Homo Ludens, <http://www.notbored.org/homo-ludens.html>, (06.02.2004).

Costikyan, Greg (1994): I Have No Words & I Must Design, [http://www.costik.com/nowords.html#Not\\_puzzle](http://www.costik.com/nowords.html#Not_puzzle), (11.02.2004).

Crawford Chris (o.A.): The Art of Computer Game Design, <http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html>, (11.02.2004).

Crombie, Duncan (o.A.): The Examination and Exploration of Algorithms and Complex Behaviour to Realistically Control Multiple Mobile Robots, <http://members.ozemail.com.au/~dcrombie/project/thesis/index.html>, (04.02.2004).

Dekke, Arne (o.A.): Körper und Geschlechter in virtuellen Räumen, [www.beziehungsbiographien.de/pub7.pdf](http://www.beziehungsbiographien.de/pub7.pdf), (20.07.2004).

Deissner, Kai; Virtel, Martin (o.A.): Schluss mit der Ballerei - Komplexe Erzählungen statt blinder Zerstörungswut: Wie sich die Welt der Computerspiele verändert, [http://www.zeit.de/2001/41/Kultur/200141\\_computerspiele..html](http://www.zeit.de/2001/41/Kultur/200141_computerspiele..html), (22.05.2004).

Dornbusch, Lutz (o.A.): Fraktale - Bildnisse der Natur aus dem Rechner?, [http://www.game-face.de/article.php3?id\\_article=21](http://www.game-face.de/article.php3?id_article=21), (20.04.2004).

Esser, Heike; Witting, Tanja (o.A.): Transferprozesse beim Bildschirmspiel, <http://www.bpb.de/snp/index.html>, (20.07.2004).

Franck, Georg (o.A.): Nichts ist nicht nur nichts, [http://www.iemar.tuwien.ac.at/publications/Franck\\_1997b\\_aufmerk\\_de.pdf](http://www.iemar.tuwien.ac.at/publications/Franck_1997b_aufmerk_de.pdf), (20.07.2004).

Fritz, Jürgen; Fehr, Wolfgang (o.A.): Videospiele in der Lebenswelt von Kindern und Jugendlichen, <http://www.bpb.de/snp/index.html>, (20.07.2004).

Glaser, Peter (o.A.): Nächstes Level, Alle tun so, als sei das Netz ein Informationsmedium. Dabei will jeder nur das eine: Spielen, [http://www.zeit.de/archiv/2000/16/200016.media\\_spiele\\_.xml](http://www.zeit.de/archiv/2000/16/200016.media_spiele_.xml), (22.05.2004).

Güttler, Christian (o.A.): Spatial Principles of Level-Design in Multi-Player First-Person Shooters, <http://www.gamasutra.com/education/theses/20031208/guttler.pdf>, (20.07.2004).

Hartmann, Knut (o.A.): GameAI (Künstliche Intelligenz in Computerspielen), [isgwww.cs.uni-magdeburg.de/~hartmann/GameAI-SS04/intro.pdf](http://isgwww.cs.uni-magdeburg.de/~hartmann/GameAI-SS04/intro.pdf), (20.07.2004).

Hermann, Marc (o.A.): Quo vadis, Machinima?, [http://www.game-face.de/article.php3?id\\_article=2](http://www.game-face.de/article.php3?id_article=2), (23.04.2004).

Herz, JC (o.A.): Gaming and the art of innovation, [http://flow.doorsofperception.com/content/herz\\_trans.html](http://flow.doorsofperception.com/content/herz_trans.html), (06.02.2004).

Jenkins, Henry (o.A.): Game Design as narrative architecture, <http://web.mit.edu/21fms/www/faculty/henry3/games&narrative.html>, (11.02.2004).

Knauer, Michael (o.A.): Computerspiele-Industrie auf der Überholspur, <http://derstandard.at/?id=1628030>, (26.05.2004).

Krell, Peter (2003): Alle Macht den Spielern - Ein Bericht vom Power up Symposium in Bristol [http://www.game-face.de/article.php3?id\\_article=29](http://www.game-face.de/article.php3?id_article=29), (19.07.2004).

Lischka, Konrad (o.A.): Warum Spieltheorie die Perspektive der Entwickler braucht, [http://www.game-face.de/article.php3?id\\_article=25](http://www.game-face.de/article.php3?id_article=25), (20.04.2004).

Maaß, Jürgen (o.A.): Kriterien für die Beurteilung von Computerspielen - Theoretische Aspekte der Bewertung, <http://www.bpb.de/snp/referate/maasacos.htm>, (20.07.2004).

Mantler, Otto (o.A.): Computerspiele (Referat für Eltern und Lehrer), <http://www.lernspiele.at/compspie.html>, (10.02.2004).

Müller, Jens (o.A.): Schneewittchen bei den Punks, Schneewitchen und der Hirschkäfer, <http://www.bpb.de/snp/referate/mueller.htm>, (20.07.2004).

Neuhaus, Wolfgang (o.A.): Software als (Mythen-)Material und Werkzeug, <http://www.heise.de/tp/deutsch/inhalt/sa/15630/1.html>, (20.05.2004).

Novak, Marcos (o.A.): An infinite symphony in space: liquid architecture, [http://www.staff.uni-marburg.de/~bohn/Virtualitaet/no\\_label\\_novak.html](http://www.staff.uni-marburg.de/~bohn/Virtualitaet/no_label_novak.html), (20.07.2004).

Ross, Kristen (o.A.): Henri Lefebvre on the Situationist International, <http://www.notbored.org/lefebvre-interview.html>, (06.02.2004).

Schartner, Christian (o.A.): Ästhetische Aspekte der Bewertung von Computerspielen, <http://www.bpb.de/snp/referate/schaacos.htm>, (20.07.2004).

Schindler, Friedemann; Wiemken, Jens (o.A.): „Doom ist invading my dreams“ - Warum ein Gewaltspiel Kultstatus erlangte, [http://www.bpb.de/snp/referate/schind\\_doom.htm](http://www.bpb.de/snp/referate/schind_doom.htm), (20.07.2004).

Schleiner, Anne-Marie (o.A.): Does Lara Croft wear fake polygons, <http://www.opensorcery.net/lara2.html>, (20.07.2004).

Schuhmacher, Patrik (o.A.): Architektur der Bewegung, in 134/135 ARCH+, [http://www.baunetz.de/arch/archplus/30475c\\_.htm](http://www.baunetz.de/arch/archplus/30475c_.htm), (14.05.2004).

Schwier, Jürgen (o.A.): Zur Soziologie des Spiels, <http://www.uni-giessen.de/~g51039/vorlesungV.htm>, (25.04.2004).

Simanowski, Roberto (o.A.): Nicholas Negropontes „Total Digital“, <http://www.dichtung-digital.org/2002/05-07-Simanowski.htm>, (20.07.2004).

Smith (2004): Deus Ex“-Produzent im STANDARD-Gespräch: „Der Spieler ist Gott“ <http://derstandard.at>, (22.05.2004).

Ströter-Bender, Jutta (o.A.): „Vor-Bilder zum Ein-Bilden“ - Zur ästhetischen Sozialisation durch Bildschirmspiele, [http://www.phil.uni-sb.de/fr/kunsterziehung/neue\\_medien/theorie/htmltheorie/str%F6ter.htm](http://www.phil.uni-sb.de/fr/kunsterziehung/neue_medien/theorie/htmltheorie/str%F6ter.htm), (20.07.2004).

Streibl, Ralph E. (o.A.): Krieg im Computerspiel - Krieg als Computerspiel, Spielend zum Sieg, <http://www.bpb.de/snp/referate/streibl.htm>, (20.07.2004).

Streibl, Ralph E. (o.A.): Krieg im Computerspiel - „Er war schon sechsmal getötet worden. Dabei war es gerade erst fünf Uhr.“, <http://www.bpb.de/snp/referate/streibl.htm>, (20.07.2004).

Virilio, Paul (o.A.): Auf dem Weg zu einem transeuklidischen Raum - Florence Michel und Nikola Jankovic im Gespräch mit Paul Virilio in 148 ARCH +, [http://www.baunetz.de/arch/archplus/148/virilia\\_c.htm](http://www.baunetz.de/arch/archplus/148/virilia_c.htm), (20.07.2004).

Walz, Steffen P. (o.A.): Computerspiele als wissenschaftliches Paradigma - Ein Kommentar zur Level-Up Konferenz, [http://www.game-face.de/article.php3?id\\_article=16](http://www.game-face.de/article.php3?id_article=16), (20.07.2004).

Walz Steffen P. (o.A.): Delightful Identification & Persuasion: Towards an Analytical and Applied Rhetoric of Digital Games, <http://www.playbe.com/stuff/games-rhetoric.htm>, (11.2.2004).

Warren, Lee E. (o.A.): Idiot Savant, <http://www.plim.org/2idiots.html>, (20.07.2004).

Wiemken, Jens (o.A.): Phänomen Bildschirmspiele: Counter-Strike, die Pädagogen [http://www.bpb.de/snp/referate/wiemk\\_cs.htm](http://www.bpb.de/snp/referate/wiemk_cs.htm), (20.05.2004).

Wolf, Mark J. P. (o.A.): Genre and the Video Game - Chapter 6 of The Medium of the Video Game, <http://www.robinlionheart.com/gamedev/genres.xhtml>, (20.02.2004).

Wurzenberger, Gerda (o.A.): Die Lerneffekte von «Ballerspielen» - Computer-Games und die Lust am Kriegerischen, <http://www.nzz.ch/dossiers/2002/schreiben-am-netz/2002.05.17-em-article85UUG.html>, (22.05.2004).

Zarze, r Brigitte (2003): Im Zeichen des Codes, Ars Electronica 2003: CODE zwischen Kunst, Justiz und Biologie, <http://www.heise.de/tp/deutsch/inhalt/sa/15577/1.html>, (20.07.2004).

Ziegler, Jens (o.A.): Autonome Agenten, <http://ls11-www.cs.uni-dortmund.de/people/ziegler/diplnode5.html>, (03.02.2004).

(o.A.): Komplexe Welten im Cyperspace - können Computersimulationen unser Denken verändern, <http://www.3sat.de/>, (Sendung vom 18.06.2004).

(o.A.): Radiomanuskripte zum Thema Spiel, <http://www.stud.uni-hannover.de/user/69332/texte/spiel.html#t2>, (24.05.2004).

DANKE !

an meine Familie, die mir das ermöglicht hat

an Carla

an Harald und Christoph  
und an Herrn Prof. Plottegg