HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr

Robert Marass

Konzeption und prototypische Umsetzung eines Orientierungsspiels

2015

Fakultät Elektro- und Informationstechnik

BACHELORARBEIT

Konzeption und prototypische Umsetzung eines Orientierungsspiels

Autor:
Robert Marass

Studiengang:
Multimediatechnik

Seminargruppe:
MK08w1-B

Erstprüfer:
Prof. Dr.-Ing. Frank Zimmer

Zweitprüfer:
Dipl.-Ing. Norbert Göbel

Mittweida, März 2015

Faculty of Electrical Engineering
and Information Techhnology

BACHELOR THESIS

Conception and prototypical realization of a orientation game

author:
Robert Marass

course of studies:
Multimedia Technology

seminar group:
MK08w1-B

first examiner:
Prof. Dr.-Ing. Frank Zimmer

second eximiner:
Dipl.-Ing. Norbert Göbel

Mittweida, March 2015

Bibliografische Angaben

Nachname, Vorname: Marass, Robert

Thema der Bachelorarbeit:

Konzeption und prototypische Umsetzung eines Orientierungsspiels

47 Seiten, 16 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Elektro- und Informationstechnik

Bachelorarbeit, 2015

Abstract

Eine Verbindung zwischen Technik und Mensch wächst in den letzten Jahren immer weiter. Sie nimmt immer mehr Einfluss in unserem alttäglichen Leben ein. Orientierung mit mobiler Technik spielt dabei eine große Rolle. Die hier vorliegende Arbeit beschäftigt sich mit der Konzeption und prototypischen Umsetzung eines Orientierungsspiels. Mittels des Unreal Development Kits wird eine Abbildung des Hochschulcampus Mittweidas erzeugt. Die zentrale Fragestellung der Arbeit lautet: Ist es möglich sich einen visuellen Überblick über die Umgebung in einer nicht realen Welt zu verschaffen?

INHALTSVERZEICHNIS

INHALTSVERZEICHNIS I

ABBILDUNGSVERZEICHNIS III

LISTINGVERZEICHNIS IV

1 Einleitung 1

1.1 Motivation 1

1.2 Zielsetzung 1

2 Vorstellung der Lösungsvariante 3

2.1 Erläuterung einiger Begiffe 3

2.2 Unreal Engine 5

2.3 Programmieroberfläche des Unreal Development Kit 7

3 Umsetzung der Konzeption und Prototypisierung 11

3.1 Idee und Konzeption 11

3.1.1 Die wichtigsten Inhalte 12

3.1.2 Der Leitfaden 12

3.1.3 Planung technischer Umsetzung 17

3.2 Technische Umsetzung des Prototyp 17

3.2.1 Autodesk 3Ds Max 17

3.2.1.1 Spielerfigur 17

3.2.1.2 Gebäude 20

3.2.2 Photoshop CS2 21

3.2.3 UDK 23

3.2.3.1 Importieren der Objekte 24

3.2.3.2 Importieren der Spielfigur 26

3.2.3.3 Unreal Scripting 27

3.2.3.4 Third-Person-Perspektive 30

3.2.3.5 Zusammenfügen der Einzelteile 31

4 Testphasen und Änderungen 35

4.1 Erster Test 35

4.2 Zweiter Test 36

5 Zusammenfassung 40

5.1 Ausblick 40

5.2 Resümee und kritische Betrachtung 40

LITERATURVERZEICHNIS 42

ANHANG A: Verwendete Software 44

ANHANG B: Inhalt der mitgelieferten CD 45

Selbstständigkeitserklärung 47

ABBILDUNGSVERZEICHNIS

Abbildung 1: Struktur des UDK 6

Abbildung 2: UDK Oberfläche 8

Abbildung 3: UDK Content Browser 9

Abbildung 4: 3Ds Max Annie Objekt 18

Abbildung 5: Annie Rigging/Animation 19

Abbildung 6: Mensa Erstellung 20

Abbildung 7: Mensa Collision Box 21

Abbildung 8: Texture Mapping 22

Abbildung 9: UDK Benutzeroberfläche 23

Abbildung 10: Content Browser 24

Abbildung 11: UDK Bibliothek + Mensa 26

Abbildung 12: Unreal Material Editor 27

Abbildung 13: UDK Klassen 29

Abbildung 14: Austausch Spielermodel 29

Abbildung 15: UnrealKismet 31

Abbildung 16: Annie und Mensa 32

LISTINGVERZEICHNIS

Listing 1: Dateiauszug DefaultEngine.ini 28

Listing 2: Alter Dateiauszug UDKUI.ini.....37

Listing 3: Neuer Dateiauszug UDKUI.ini 38

1 Einleitung

Diese Arbeit beschäftigt sich mit der Konzeption und prototypischen Umsetzung eines Orientierungsspiels. Ziel ist es am Ende, mit einem Computerprogramm Namens *Unreal Development Kit*, den Prototypen eines Computerspieles zu entwickeln. Aufgabe des Spiels ist es, den Spieler über den Campus der Hochschule Mittweida zu informieren und ihm Orientierung zu geben. Man muss somit nicht unbedingt physisch in Mittweida anwesend sein, um sich einen Eindruck vom Campus zu machen.

1.1 Motivation

Mittweida, „der Campus der kurzen Wege“.

Vielen Studenten fällt es zum Start des neuen Lebensabschnitts Studium, oft schwer sich in der neuen ungewohnten Umgebung zu orientieren. Ein neues Umfeld und so viele fremde Menschen überfordern einige zum Anfang sehr. Durch diese Erkenntnis wäre es doch toll sich im vornherein einen Überblick über den Campus machen zu können d.h. sich schon einmal Informationen vom heimischen Sofa aus, auf spielerischer Weise zu besorgen. Mit der stetig wachsenden Technologie und Mobilität heutzutage wäre es ein Einfaches. In unserem Zeitalter des „State of the Art“[1] wäre es ein Vorteil auf einem mobilen Endgerät eine Orientierungs-App[2] zu besitzen. Ob auf dem Weg zum ersten Studientag oder kurz vor dem Abschluss des Abiturs sich schon einmal einen Überblick von seiner zukünftigen Arbeitsumgebung zu verschaffen, wäre eine hilfreiche Möglichkeit der Unterstützung.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es ein Konzept und einen Prototypen eines Orientierungsspiels zu erstellen, welches neuen Studenten erlaubt sich auf ihrem

[1] Bezeichnet einen hohen Entwicklungszustand einer Technologie

[2] Anwendungssoftware

mobilen Endgerät in einer virtuellen Umgebung von Mittweida umzuschauen und zu informieren. Dieser Prototyp soll getestet und ausgewertet werden.

2 Vorstellung der Lösungsvariante

Das Orientieren in einer nicht realen Umgebung wird in dieser Arbeit nicht neu erfunden. Es gibt diverse Internetseiten, die innovative Präsentationen eines Rundgangs für Unternehmen erschaffen und damit ihren Kunden ihr Gewerbe attraktiver gestalten wollen. Ein Beispiel hierfür ist die Internetpräsenz *www.virtueller-rundgang.eu* (gefunden am 11.02.2015) oder das bekannte Google StreetView (zu finden unter https://www.google.de/maps/).

Da das stupide Betrachten solcher Plattformen nicht sehr interaktiv gestaltet wird, wirkt es für den Zuschauer etwas langweilig. Ein aktives Eingreifen im Geschähen wäre da doch viel Interessanter. Bei diesem Projekt wird darauf mehr Wert gelegt. Deswegen wurde die Wahl getroffen ein Computerspiel zu entwerfen. Der Betrachter wird gleichzeitig zum interaktiven Spieler seiner Umgebung die betrachtet und erkundet werden kann.

Ein Computerspiel zu entwerfen, stellen sich die meisten einfacher vor als es in der Wirklichkeit ist. Das eine Person alleine ein Spiel entwickelt und programmiert, daran ist heutzutage nicht mehr zu denken. Dies war früher einfacher zu lösen. Die immer weiter wachsenden Anforderungen an ein Computerspiel von Seiten der Hersteller und Nutzer macht es eben so komplex und aufwändig. (vgl. [1] Blow 2004, S. 29) Es ist vergleichbar mit einer Produktion eines bekannten Hollywood-Film. (vgl. [4] Lorber 2008, Fokus_ 12/13)

2.1 Erläuterung einiger Begiffe

Engine (Spiel-Engine)

Engine bedeutet wortwörtlich Übersetzt „Antrieb“ oder „Motor“.

Sie ist ein eigenständiges Teil eines Computerprogramms in der Informationstechnologie. Eine Engine ist für unterschiedliche Aufgaben entworfen worden, wie zum Beispiel die Spiel-Engine.

Die ***Spiel-Engine*** ist für Computerspiele das „Herz“ und deswegen heutzutage

unumgänglich. Diese Engine ist für den Spielverlauf, für die visuelle Darstellung, die künstliche Intelligenz, Physik etc. verantwortlich. Aber dazu später mehr. Solch ein Baukasten wird auch als Entwicklungsumgebung genutzt, welche diverse Tools[3] enthält. Eine der bekanntesten Spiele-Engine ist die *CryEngine*, *Unity Engine* und die *Unreal Engine*. Es gibt noch diverse andere Spiele-Engine. Solch Spiele-Engine zu entwickeln ist sehr aufwändig, deshalb verkaufen die Entwickler solcher Engine ihr Endprodukt an anderen Studios weiter, welche dann ihr eigenes individuelles Computerspiel produzieren und verkaufen können. (vgl. [6] Mittler 2013, S. 1)

Eine Spiel-Engine besteht des Weiteren aus folgenden Systemen:

Grafik-Engine (graphic asset)

Die Grafik-Engine, wie schon im Wort „Grafik“ enthalten, ist für die visuelle und optische Darstellung verantwortlich. Diese Engine ist verantwortlich für die,

- Textausgaben
- Verwalten, Darstellen und Laden von Texturen[4]
- Erstellen von Wasser-, Nebel-, Feuereffekte etc. (Was meist mit der Physik-Engine kombiniert wird)
- Shader[5]-System, dieses System ist z.B. für die Darstellung realistischen Zusammenspiel von Licht und Schatten auf verschiedene Texturen oder Materialien verantwortlich.

Physik-Engine (physic asset)

[3] Werkzeuge

[4] Eine Textur bezeichnet ein Bild, das auf der Oberfläche eines virtuellen Körpers dargestellt wird

[5] „Shader sind mathematische Algorithmen, welche die Aufgabe haben, bestimmte Effekte beim Berechnen einer dreidimensionalen Computergrafik zu erzeugen.“ (Rösler, Ronny: Hochschule Fulda, Projekt Einführung in die Shader Programmierung unter OpenGL 2.0, Seite 3, online zu finden unter: http://www2.hs-fulda.de/caelabor/inhalte/OpenGL/Projekte/roesler/shaderprog.pdf (gelesen am 26.01.2015)

Bei neueren Spielen wird immer mehr Wert auf visuellen Realismus gesetzt. Fällt zum Beispiel ein Apfel vom Baum auf den Boden, so bleibt er nicht gleich starr liegen, sondern springt kurz auf und rollt eventuell ein Stück zur Seite, je nach physikalischer Form des Apfels.

Die Physik-Engine bekam erst in den letzten Jahren der Spiele-Entwicklung eine größere Aufmerksamkeit mit der stetig wachsenden und immer stärker werdenden Computer-Hardware.

Sound-Engine (sound asset)

Dieses Packet ist zuständig um akustische Stimmungen zu erzeugen und den räumlichen Eindruck der Spielwelt zu verstärken. Heutzutage gibt es verschiedene Technologien um realistische Klangeffekte zu erzeugen. Zum Beispiel das von dem Unternehmen „Creative Labs“ entwickelte EAX (Environmental Audio Extensions) mit dem man Geräusche je nach physikalischen Umgebungen virtualisieren kann. Ein kleines Beispiel ist der Hall in einem leeren Raum oder in eine Höhle.

Scripting

Ohne irgendwelche vorprogrammierten Spielabläufe funktioniert kein Computerspiel. Egal ob zur Erstellung einer Story, das Laden und Speichern von Speicherständen oder das Programmieren von Netzwerkcodes (z.B. für Mehrspieler Spiele) etc. ist das Scripting unumgänglich. Ein Beispiel für eine Scriptsprache ist das Java-Script, welches zur Erstellung von kleinen Applikationen dient. Auch das PHP-Scripting, was in der Webprogrammierung benutzt wird, ist den meisten bekannt. Die meisten Spiel-Engine besitzen ihre eigene Skriptsprache, welche für den Programmierer eine vereinfachte Modifikation von Spielen ermöglicht.

2.2 Unreal Engine

In diesem Abschnitt wird genauer auf die Unreal Engine eingegangen. Der Prototyp wird voraussichtlich mit dieser Engine realisiert. Der Grund warum genau die Unreal Engine infrage kommt, wird im weiteren Text klar.

Das erste Mal wurde die Unreal Engine 1998 kostenpflichtig veröffentlicht. Aber der größte Meilenstein in deren Geschichte war im Jahr 2009. Da veröffentlichte der Entwickler *Epic Games* eine kostenlose, auch nicht-kommerziell zu verwendende Variante der Unreal Engine, das Unreal Development Kit (UDK). Auch die Preise der kommerziellen Verwendung dieser Unreal Engine haben die Entwickler angepasst und reduziert. Der zweite Unterschied zwischen der Unreal Engine und das UDK besteht darin das in der kommerziellen Variante das Entwicklungstool und der Entwickler- C++ Quellcode zur Verfügung steht, was bei dem UDK nicht der Fall ist. Hier gibt *Epic Games* den Programmierer nur das Entwicklungstool zur Verfügung und die Möglichkeit Scripts über die integrierte UnrealScript Sprache zu bearbeiten.

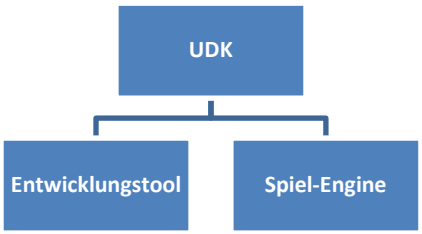


Abbildung 1: Struktur des UDK (vgl. [8] Thorn 2012, S. 4)

Wofür kann das UDK eingesetzt werden?

Hier müssen wir erst einmal Unterscheiden! Wir wissen, dass das UDK dazu da ist um 3D Anwendungen und 3D Animationen zu erstellen. Aber es gibt 2 unterschiedliche Verfahren eines 3D Modeling.

Als erstes die *pre-rendered*[6] Variante:

Hierbei wird ein Bild von einem Objekt, aus einer festen Kameraperspektive mit Schatten, Spiegeleffekten und sämtlichen Einstellungen erstellt. Dies nimmt viel Zeit in Anspruch, da das Bild Pixel pro Pixel einzeln berechnet werden muss. Diese Variante wird meisten zur Erstellung von hochauflösenden Animationsfilmen oder realitätsnahe/animierte Bilder verwendet. Da zum Abspielen solcher vorgerenderten Medien dann kein großer Aufwand auf die Hardware des Abspielgerätes voraussetzt wird. (ebd., S. 10)

Und als zweites die *real-time-rendered*[7]Variante:

[6] Vorgerendert

In diesem Verfahren werden die Bilder in Echtzeit gerendert. Das heißt, bewegt man seine Spielfigur oder die Kamera in irgendeiner Richtung wird dies sofort berechnet und gleichzeitig am Bildschirm ausgegeben. Bei dieser Variante ist Geschwindigkeit oberste Priorität. Das rendern der einzelnen Bilder in einer Sekunde wird den Zuschauern nur hochauflösend vorgegaukelt. (ebd., S. 10) Da diese Methode sehr umfangreich ist, will ich hier nicht weiter darauf eingehen.

Fazit:

Dieses Kit wird meistens zur Erstellung von Computerspielen genutzt. Egal ob 2D oder 3D Anwendungen. In Wirklichkeit wird einen die Dreidimensionalität nur optisch vorgegaukelt, aber den Spieler fällt es nicht wirklich auf. Man schaut ja auf einen Bildschirm der nur eine Horizontale und Vertikale besitzt. Aber zum Beispiel durch das nach vorn und zurück, links und rechts, springen und ducken einer Figur, die man gerade spielt, sieht es aus als würde man sich virtuell in einen dreidimensionalen Raum bewegen. (ebd., S. 7f)

Es kann aber auch anderweitig eingesetzt werden. Wie ich es vor kurzer Zeit aus eigener Erfahrung erlebt habe. Was früher schon einmal vorgestellt wurde aber nie wirklich den Markt erobert hat ist eine Virtual Reality Brille. Aber jetzt hat die Firma *Oculuc Rift VR* ein neues Modell auf dem Markt gebracht Namens: *Oculus Rift*, diese Brille ist vollkommen kompatibel mit der Unreal Engine. (vgl. [6] Mittler 2013, S. 6)

Das ist auch ein weiterer Grund warum das Unreal Development Kit eingesetzt wird. Mit solch einer Technik kann man einen virtuellen Rundgang auf dem Campus sehr gut simulieren. Man fühle sich als wäre man mitten im Geschehen.

2.3 Programmieroberfläche des Unreal Development Kit

Das UDK ist eine Kombination aus Grafischer Benutzeroberfläche (GUI – graphical user interface) und das vorher schon erwähnten, einfach zu benutzende, Entwicklerprogramm. Es ist möglich Spiele für verschiedene Plattformen zu

[7] Echtzeitgerendert

programmieren wie zum Beispiel für einen Windows PC, einer Spielekonsole, einen Macintosh oder man Entwickelt ein Spiel für ein mobiles Endgerät.

Schön bei diesem Kit ist die relativ einfache Handhabung, einfach mit der Maus und per drag-and-drop Funktion einen Würfel in die virtuelle Welt zu bringen und somit sich seine eigene Landschaft zu bauen. Das ist aber nicht alles, weitere wichtige Parameter müssen noch ergänzt werden. Auch die Echtzeit-Vorschau ist eine hervorragende Funktion in dem UDK. Der Programmierer sieht dadurch gleich ob der Würfel der gesetzt wurde auch gleich am richtigen Platz sitzt. In diesem ganzen Packet sind noch unterschiedliche Editoren darin implementiert z.B. der Material Editor, der Content Browser, ein Mesh[8] Editor und vieles mehr.

Das UDK unterstützt auch das Importieren/Exportieren von Dateien die mit einen Industriellen-Standard-Format kompatibel sind. Somit ist es möglich Gebäude, Grafiken oder Sounds aus einer Drittpartei-Anwendung einzubinden.

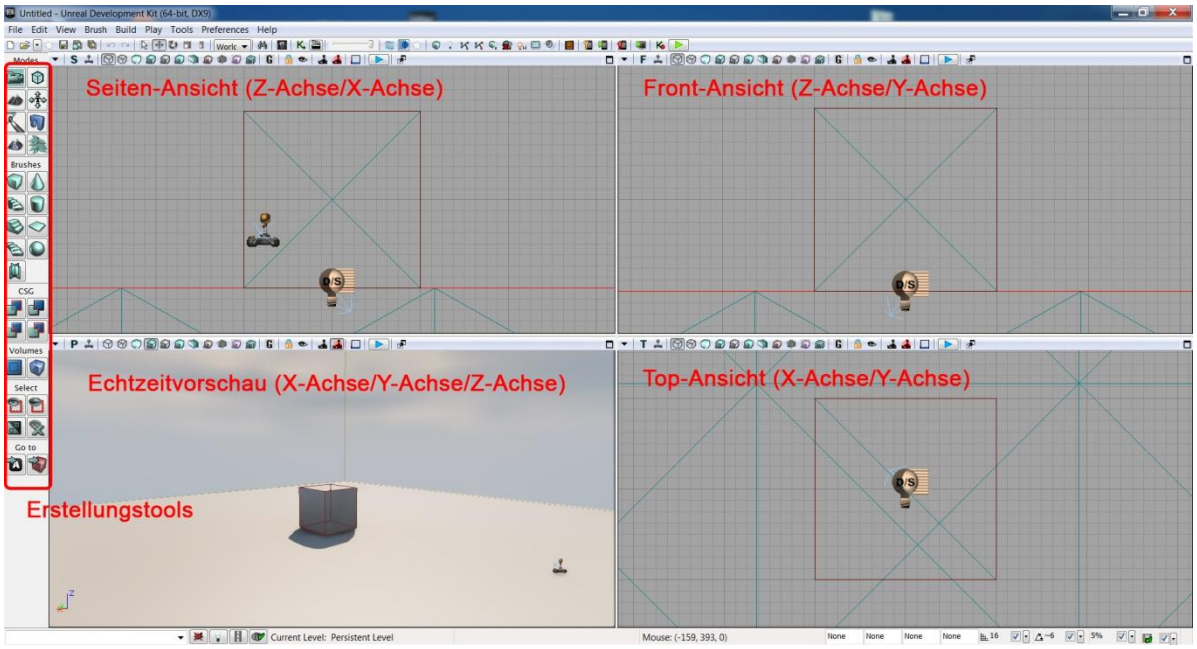


Abbildung 2: UDK Oberfläche

Auf Abbildung 2 ist die Benutzeroberfläche des Unreal Development Kits zu

[8] Ein Mesh ist ein aus mehreren Ecken, Kanten und Flächen zusammengestelltes Objekt (z.B. ein Würfel)

erkennen. Es erklärt ganz Grob den Aufbau dieser Oberfläche. Dieses doch relativ umfangreiche Programm beinhaltet mehrere Funktionen. Es würde aber den Rahmen sprengen jedes einzelne Detail hier zu erklären. Zu erkennen sind die 4 Hauptansichten mit dem man sich gut bei der Erstellung der Karte Orientieren kann. Links im Bild sind die sogenannten Erstellungstools, wie Brushes[9], Mode[10] etc. Mit diesen kann man verschiedene geometrische Figuren erstellen und individuell in Größe und Form bearbeiten.

Noch zu erwähnen wäre der interne Browser des UDK. Der sogenannte Content Browser. Eine kurze Erklärung hierfür ist auf Abbildung 3 zu finden.

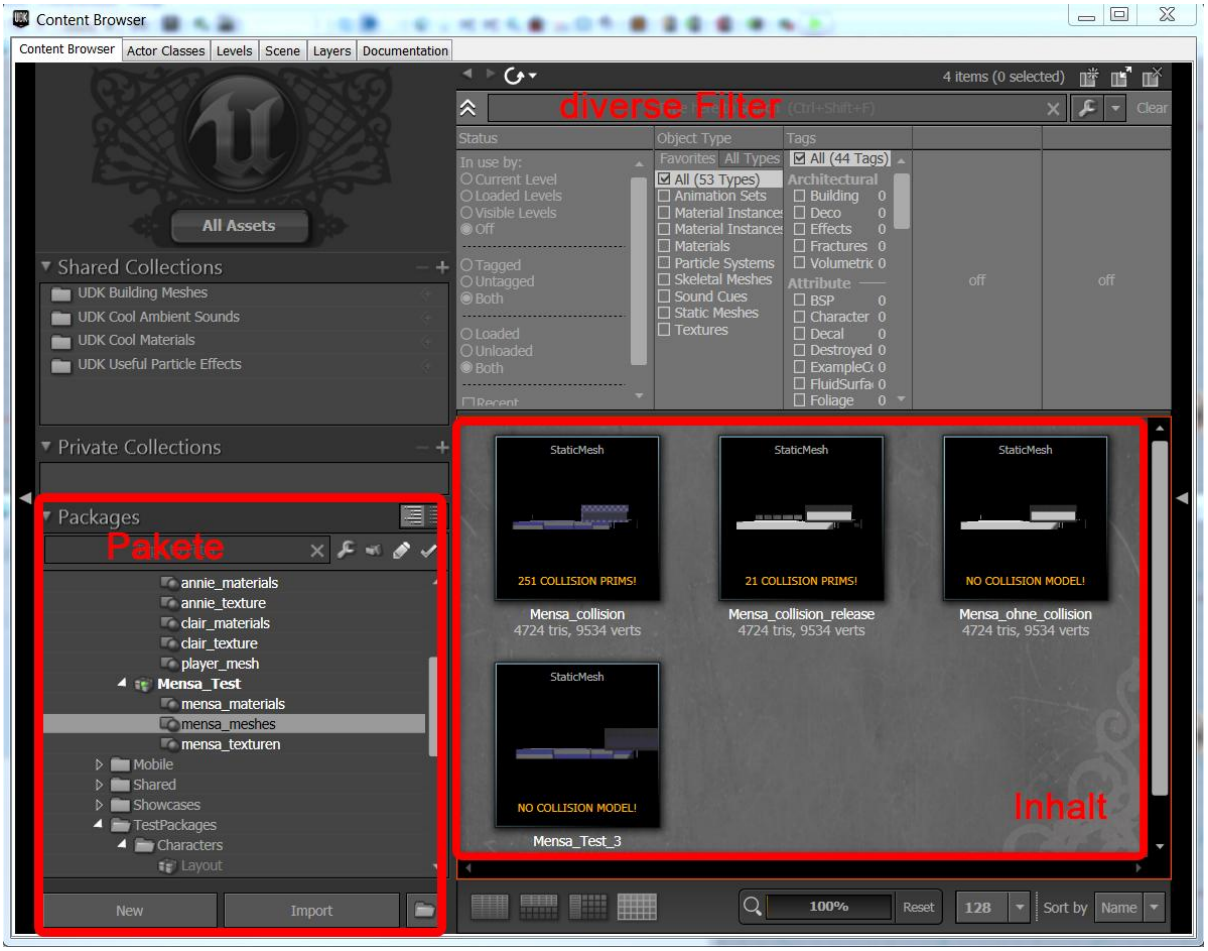


Abbildung 3: UDK Content Browser

[9] Wortwörtlich übersetzt Bürsten – Sind Vorgefertigte, geometrische Formen

[10] Modus

Im linken Teil dieses Browsers sind die Pakete zu finden. Dort werden sämtliche Meshes, Texturen, Materialien, Animationen etc. in beliebig angelegte Pakete gespeichert. Hier gibt das UDK einen auch die Möglichkeit externe Objekte, Texturen zu importieren und abzuspeichern. Rechts daneben wird der Inhalt der Pakete angezeigt. Hier gibt es noch diverse Einstellungsmöglichkeiten.

Das erzeugen von Abläufen im erstellten Spiel, wie z.B. das Erstellen eines Schalters der betätigt werden kann, wird über das UnrealKismet gesteuert. Im Punkt 3.2.3.5 (Zusammenfügen der Einzelteile) wird darauf etwas näher eingegangen und es wird ein Beispiel gezeigt, was für dieses Projekt von Bedeutung ist.

3 Umsetzung der Konzeption und Prototypisierung

Die richtige Planung ist das Alpha und Omega für große Projekte. Der grobe Ablauf einer Produktion eines Computerspiels ist wie folgt:

1. Idee und Konzept (2 - 4 Monate)
2. Finanzierung und Kostenplan (1 - 2 Monate)
3. Produktionsphase (mehrere Monate im zweistelligen Bereich)
    a. Entwickeln und Programmieren des Spiels
    b. Testphase und Fehlerbehebung
4. Release/Werbung/Verkauf (ungefähr 2 Monate)

(vgl. [5] Lorber 2011, Onlinebericht)

Bei diesem Projekt wird implizit auf Punkt 1 und 3 eingegangen, wobei es sich bei dem Punkt: Produktionsphase nur um einen Prototypen handeln wird. Die anderen Punkte sind hier zu vernachlässigen, da dies aus zeitlichen Gründen, mangelnder Mitarbeiter und finanzieller Lage nicht zu realisieren ist.

3.1 Idee und Konzeption

Wie schon oft in dieser Arbeit erwähnt soll ein Computerspiel programmiert werden, indem sich ein Spieler virtuell in einer Nachbildung vom Campus Mittweida frei bewegen kann. Wegweiser soll einen das finden der Hauptgebäude vereinfachen. Eine Art Story (Leitfaden) soll einen den ersten Rundgang vereinfachen. Diese Geschichte ist im Punkt 3.1.2 zu finden. Die Wahl der Programmierapplikation ist getroffen. Das *Unreal Development Kit* wird hierbei eingesetzt. Auch zum Einsatz soll das 3D Modeling Programm „3Ds Max“ von der Firma Autodesk kommen. Dieses Programm wurde gewählt, da es eine kostenlose drei Jahres-Lizenz für akademische Nutzer anbietet, die aber nur eine Nicht-kommerzielle Nutzung zulässt. Auch die Nutzung von Adobe Photoshop CS2 die später zum Einsatz kommt ist Kostenfrei.

3.1.1 Die wichtigsten Inhalte

Dieses Spiel soll mit einer Third-Person-Perspektive ausgestattet werden. Das heißt, dass der Spieler die Umgebung über die Schulter seiner Charakterspielfigur betrachten kann. Da diese eine bessere Atmosphäre vermittelt als bei einer First-Person-Perspektive. Diese „Ich“-Perspektive wird oft bei Ego-Shootern eingesetzt. Diese gewaltverherrlichenden Spiele haben nicht den besten Ruf. Viele Betrachter, die ein Spiel mit der First-Person-Perspektive sehen, denken gleich es sei ein gewalttätiges Spiel. Was aber bei diesem Projekt nicht der Fall sein wird. Die Spielfigur (das Model) sollte auch ausgetauscht werden, da in dem Unreal Development Kit nur Roboter oder Marine Kämpfer enthalten sind, die für diesen Prototyp eine unpassende Wahl wären. Ein Problem bei der Erstellung des Levels ist das Einhalten der Größenverhältnisse. In der Programmierapplikation UDK wird nicht mit einen metrisches System gearbeitet, sondern mit sogenannten UnrealUnits[11] (UU). Dies macht es zum Beispiel bei der Erstellung eines Mülleimers oder einen Baum schwer die Größenverhältnisse zu behalten. Oder ein Haus was im wahren Leben 20 Meter hoch ist, sollte im Spiel auch so wirken als wäre es so groß und nicht nur gefühlte 5 Meter. Deshalb werden Skizzen angelegt wo man die Größenverhältnisse einigermaßen umsetzten kann.
Das Erstellen und Programmieren aller Hauptgebäude vom Campus Mittweida ist in dieser kurzen Zeit nicht machbar. Deshalb wird lediglich nur das Erstellen von Haus 1 und die Mensa festgehalten.

3.1.2 Der Leitfaden

Aktuellere Titel legen sehr viel Wert auf eine spannende und fesselnde Geschichte, wie zum Beispiel das Spiel „Heavy Rain“ von Quantic Dream. Dem Spieler wird eine große Freiheit gelassen wie er in verschiedenen Situationen reagieren kann. Aber im Großen und Ganzen verläuft das Spiel nach einem Haupt-Leitfaden, der sogenannte rote Faden, nach dem der Spieler gehen muss, um ans Ende zu

[11] UnrealUnits ist die Größeneinheit mit der das UDK arbeitet.

gelangen. (vgl [3] Lewalter 2010, S. 1) Auch in dieser Arbeit wird eine kleine Geschichte entworfen um den Spieler einen Rundgang auf dem Campus zu ermöglichen und ihm die ersten wichtigen Wege zu zeigen. Die Informationsgewinnung ist bei diesem Projekt die oberste Priorität. Dem Spieler darf es aber trotzdem nicht langweilig werden. Kleinere Erfolgserlebnisse spielen dabei eine Rolle. Im nächsten Absatz wird diese geplante Geschichte kurz erläutert und diverse Dialoge hinzugefügt.

Im folgenden Abschnitt wird die Spielfigur die der Spieler spielt „Annie“ genannt. Auch sogenannt NPC[12] (non playable character) sind hier mit enthalten:

Annie kommt ca. 12:30 Uhr vom Bahnhof in Richtung Campus an. Der erste Stopp ist die Mensa, wo sie sich gleich zum Mittag etwas Zu essen kauft. Sie entdeckt kleine Automaten womit man vermutlich mit einer kleinen Karte bezahlen kann.

Die „Kassiererin“ (NPC) spricht zu Annie.

Person	***Dialog***
Kassiererin	*Du kannst auch mit deiner HSMW-Card bezahlen!*
Annie	*Was ist das? Wo bekomme ich die her?*
Kassiererin	*Das ist der Studentenausweis hier in Mittweida. Den bekommst du gleich hier um die Ecke!*
Annie	*Ach cool! Danke für die Info!*

Annie isst ihr Mittagessen. Danach macht Sie sich auf dem Weg dorthin und holt sich ihre HSMW-Card. Annie schaut sich beim Verlassen der Mensa ein bisschen um und entdeckt einen PC-Pool. Außerhalb der Mensa entdeckt sie einen Wegweiser. Darauf steht „Sekretariat“. Sie folgt den Wegweisern. Angekommen am Haus 1, betritt sie das Sekretariat.

[12] Nicht-Spieler-Charakter – ist eine Figur mit der man im Spiel agieren kann, die aber nicht vom Spieler eigenhändig gesteuert wird

Annie spricht die „Sekretärin“ (NPC) an.

Annie	*Hallo, ich bin Annie! Ich bin heute das erste Mal hier in Mittweida. Ich hab keinen richtigen Plan wo ich überhaupt hin soll. *grins** *Ich gehöre zur **Fakultät XY**!*
Sekretärin	*Hallo Annie, kein Problem. Hat doch jeder Mal neu angefangen. Hier hast du eine kleine Karte vom Campus. Damit kannst du dich vielleicht ein bisschen besser orientieren.*
Annie	*Oh danke, das ist sehr nett!*
Sekretärin	*Also, deine Fakultät ist im **Haus XY**!* *Hast du noch weitere Fragen?*
Annie	*Oh wie nett!* *Nein danke! Auf Wiedersehen!*
Sekretärin	*Tschüss Annie!*

***Fakultät XY**, hier soll eine Auswahl vom Spieler getroffen werden zu welcher Fakultät man gehört. Danach richtet sich die Antwort der Sekretärin „im **Haus XY**“. Dementsprechend wird dem Spieler auf der erhaltenen Campuskarte das Haus markiert und angezeigt die der Spieler jederzeit abrufen kann.*

*Annie begibt sich zum entsprechenden Gebäude. Unterwegs dorthin gibt es die Möglichkeit sich ein bisschen umzuschauen. Annie hat das **Haus XY** gefunden und betritt dies.*

Annie spricht mit der „Fakultätssekretärin“ (NPC).

Annie	*Guten Tag, mein Name ist Annie und ich bin auf der Suche nach meinem Stundenplan. Bekomme ich den hier?*

Fakultätssekretärin	*Hallo Annie, leider nicht. Stundenpläne sind nur Online abrufbar, auf der Hochschul-Internetseite. Hast du deine Online-Zugangsdaten schon abgeholt?*
Annie	*Achso! Nein, das habe ich noch nicht. Wo bekomme ich die denn her?*
Fakultätssekretärin	*Da gehst du zu Haus 3! Da gibt es das Netz- und Kommunikationszentrum. Das wird auch NCC genannt. Steht sogar draußen dran.* *Da kannst du dir deine Online-Zugangsdaten besorgen.*
Annie	*Ah, Okay! Da mach ich mich doch gleich mal auf den Weg dorthin *grins** *Vielen Dank!*
Fakultätssekretärin	*Kein Problem! Schön Tag noch!*

Annie schaut auf die Campuskarte. Haus 3 ist markiert und Annie kann sich daran orientieren. Am NCC angekommen wird es betreten. Ab jetzt besitzt sie ihre Zugangsdaten für den Onlinezugriff der Hochschule Mittweida. Sie erinnert sich an den PC-Pool in der Mensa. Sie will endlich ihren Stundenplan, um zu wissen was sie so für Vorlesungen und Seminare hat. Sie geht zurück zur Mensa und sieht das der PC-Pool gleichzeitig auch die Bibliothek ist. Sie geht Online und besorgt sich ihren Stundenplan. Erleichtert denkt jetzt Annie, dass sie alles erledigt hat und will sich auf den Heimweg machen. Beim Verlassen der Mensa kommt „Fred“ (NPC), ein ehemaliger Klassenkamerad, ihr entgegen und spricht sie an.

Fred	*Hey Annie, schön dich hier zu treffen! Hab gar nicht gewusst dass du dich hier in Mittweida auch beworben hast.*
Annie	*Hi Fredi! Ja, das hab ich! Weißt doch dass meine Interessen in Richtung **Fakultät XY** gehen. *grins**

Fred	*Ja, stimmt! *grins** *Hast du dir eigentlich schon deinen BAföG-Antrag geholt?* *Da komme ich nämlich gerade her.*
Annie	*Ach ja!!! Das hab ich ja total vergessen. Wo hast du den denn her?*
Fred	*Den kannst du dir im Wohnheim 3 holen!*
Annie	*Danke für die Info! Das mach ich jetzt schnell! Ich wäre jetzt nach Hause gegangen!* *Wir sehen uns Fred! Tschüssi!*
Fred	*Mach das Annie, das dauert ja auch ein Stück mit dem BAföG!* *Bye Annie!*

Annie schaut auf die Campuskarte und sucht darauf wo Wohnheim 3 ist. Sie läuft los. Abhängig davon welchen Weg sie nimmt, kommt sie am Studentenclub vorbei oder nicht. Wenn sie da vorbei kommt, steht ein Student vorm Club. Er spricht sie an und erklärt ihr, dass es der Studentenclub sei und sie doch bei der nächsten Veranstaltung eingeladen ist. Annie kann entscheiden ob sie zusagt oder nicht (eine Art Zusatzaufgabe). Weiter in Richtung Wohnheim 3. Dort angekommen werden gleich BAföG Antrag und Informationen wie z.B. Öffnungszeiten eingeholt. Annie macht sich zurück auf den Weg Richtung Bahnhof. Sie will nach Hause.

Diese Geschichte wäre das 1. Kapitel für eine Vollversion des Spiels. Aber der große Aufwand der entstehen würde, diese Story zu programmieren, würde den Rahmen sprengen. Deshalb beschränkt sich die Story bei diesem Prototyp auf dem Weg von der Mensa zum Sekretariat im Haus 1.

3.1.3 Planung technischer Umsetzung

Die Erstellung des Prototyps muss in einzelnen Abschnitten erfolgen. Zuerst wird nach einer geeigneten Spielerfigur gesucht. Diese muss so bearbeitet werden das es möglich ist diese in das UDK einzubinden. Desweitern ist das Erstellen der zwei Hauptgebäude geplant. Auch hier ist geplant diese separat in das UDK einzubinden und zu testen. Das Erstellen der Third-Person-Perspektive sollte als eigenständiges Projekt erzeugt und probiert werden, um so viele Fehlerquellen wie möglich auszuschließen. Ganz zum Schluss sollen diese einzelnen, kleinen Projekte zu einem Endprojekt zusammengesetzt werden. Eventuelle Kamerafahrten sollen eingebracht werden, um die Orientierung für den Spieler zu vereinfachen.

3.2 Technische Umsetzung des Prototyp

Der zweite Hauptteil dieser Arbeit beschäftigt sich mit der prototypischen Umsetzung des Videospiels. In diesem Punkt wird die Erstellung des Prototyps dokumentiert. Hier wird gezeigt welche Programme verwendet wurden und welche Probleme entstanden sind.

3.2.1 Autodesk 3Ds Max

3Ds Max ist ein Programm zur Erstellung von 3D-Modellen, -Animationen und -Rendering. Viele Benutzer dieser Software kommen aus den Bereichen der Spiele- und Filmproduktion. Das Erstellen komplexer und hochauflösender Objekte ist damit möglich. Die Komplexität der enthaltenen Funktionen und Tools macht es für viele Benutzer so interessant. Es ist aber eine lange Einarbeitungszeit zur Benutzung dieser Software von Nöten.

3.2.1.1 Spielerfigur

Wie schon im Punkt 3.1.1 erwähnt, muss die Spielfigur (das Model) bei dem Prototyp geändert werden. Hierbei wurde entschieden, dass ein schon fertig gemodeltes Objekt aus dem Internet verwendet werden soll. Nach einiger Suche fiel

die Entscheidung auf das Spieler-Model Annie (Siehe Weitere Online-Quellen [9], eingereicht von 3dregenerator, Nicht-kommerzielle Lizenzrechte). Annie ist eigentlich eine Spielfigur aus dem Spiel „Dead Rising 3“, welche aber von luxox_18 nachgemodelt und bereitgestellt wurde. Diese Figur ist aber noch ein normales Objekt wie z.B. eine Laterne oder ein Baum. Siehe Abbildung 4.

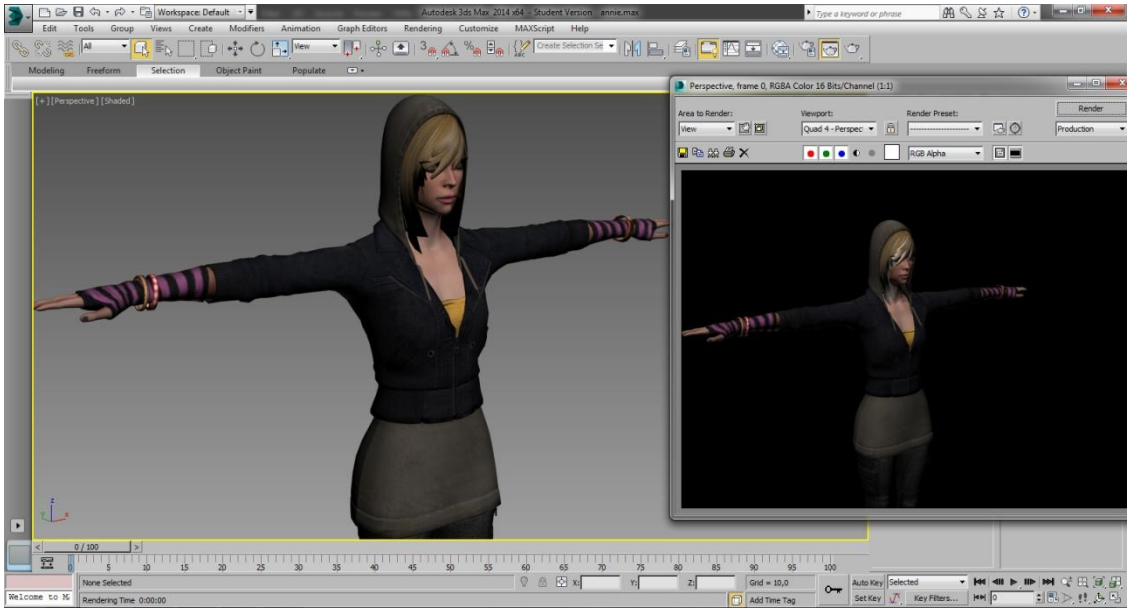


Abbildung 4: 3Ds Max Annie Objekt

Um dieser Figur sogenanntes „Leben“ einzuhauchen, muss ein Rigging[13] angewendet werden. Beim Rigging wird dieser Figur eine Art Skelett daruntergelegt. Füße, Beine, Gelenke, Arme, Hände werden dann miteinander verknüpft und konfiguriert. Diese zwei verknüpften Modelle bilden dann das Produkt einer beweglichen Figur.

[13] Wortwörtlich Übersetzt Takelwerke

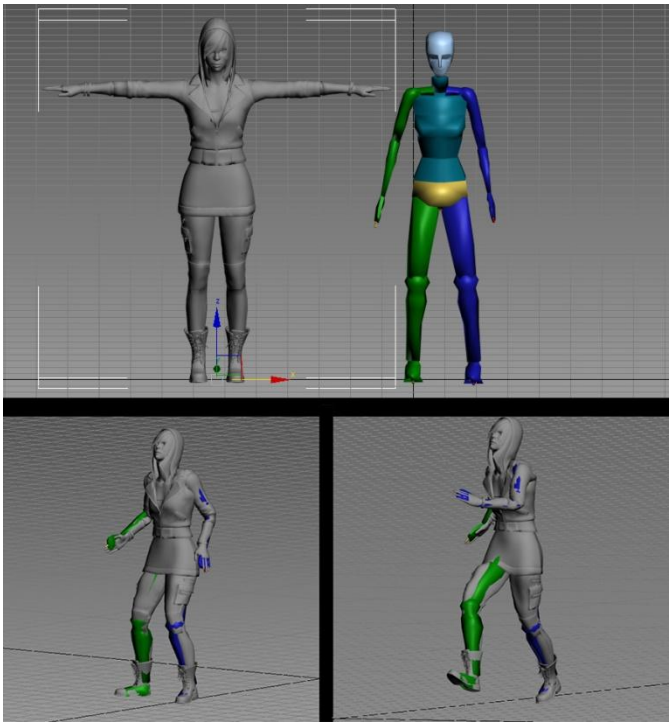


Abbildung 5: Annie Rigging/Animation

Auf Abbildung 5 ist gut zu sehen wie die zwei Modelle miteinander verbunden sind. Hier ist auch die erste Animation zu erkennen.

Um diese Spielfigur in das Entwicklerprogramm UDK zu bekommen musste sie Exportiert werden. Als erstes wurde Annie als FBX[14]-Datei exportiert. Da dies aber Probleme mit der Verbindung zwischen Mesh-ID[15] und Material-ID gab, musste eine alternative gefunden werden. Die Zuweisung der Materialien/Texturen war fehlerhaft. Dieses Model besteht aus verschiedenen zusammengesetzten Polygonen-Gitter. Zum Beispiel: Haare, Gesicht, Ärmel, Hose, Schuhe etc. Jeden dieser Teile wurde eine Identifikationsnummer (ID) zugeteilt, damit die UDK weiß welches Material wohin gehört. Nach dem Exportieren gab es aber nur noch eine ID für das gesamte Model.

Ab hier kam dann das Addon ActorX für 3Ds Max zum Einsatz. Damit war es

[14] Ist ein Dateiformat welches das UDK lesen und interpretieren kann

[15] ID - identification code – Identifikationsnummer

möglich Annie, samt „Skelett“ und richtiger Zuweisung der Materialien zu exportieren (Siehe mitgelieferte CD, Dateipfad „Spielfigur/Annie_16_Feb_ActorX_X_ModelInfo.txt“). Auch das Exportieren von Animationen beinhaltet dieses Addon.

3.2.1.2 Gebäude

Das 3D Programm 3Ds Max wurde auch für die Erstellung der Gebäude benutzt. Das erste gemodelte Gebäude ist die Mensa/Bibliothek. Hier wurde nur ein Teil der Außenfassade, der vom Spieler zu sehen ist, erstellt. Umso weniger Polygone ein Mesh besitzt, umso weniger Leistung wird von der Hardware des Abspielgeräts benötigt.

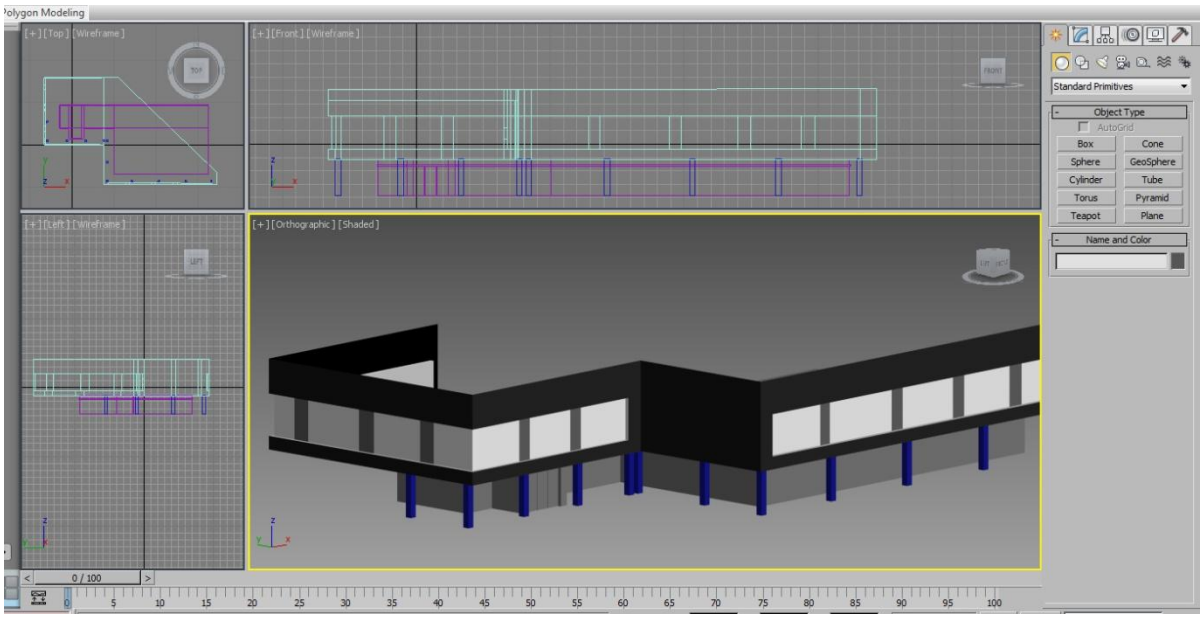


Abbildung 6: Mensa Erstellung

Auf Abbildung 6 sieht man den ersten Entwurf der Mensa. Auch gut zu erkennen ist das wirklich nur die Fassade erstellt wurde. Im Prototyp wird der Spieler nicht die Möglichkeit haben dieses Gebäude von oben zu sehen, deshalb braucht es nicht gemodelt zu werden.

Nach Fertigstellung der Mensa erfolgte ein erster kleiner Test mit Exportieren und Importieren in den UDK. Das Einbinden lief unter Berücksichtigung eines kleinen

Fehlers problemlos. Das Gebäude stand, aber die Spielfigur lief einfach durch das Gebäude durch. Es musste eine sogenannte Collision-Box[16] erstellt werden. Dieses Problem konnte ganz schnell in 3Ds Max gelöst werden. Überall wo der Spieler mit seiner Spielfigur gegen Wände laufen müsste, wurden Quader darunter gelegt. Ist jede Wand mit einem Quader versehen, dann werden diese zum Schluss zusammengeführt und genau so genannt wie das gemodelte Gebäude, nur mit den Zusatz „UCX_“ am Anfang.

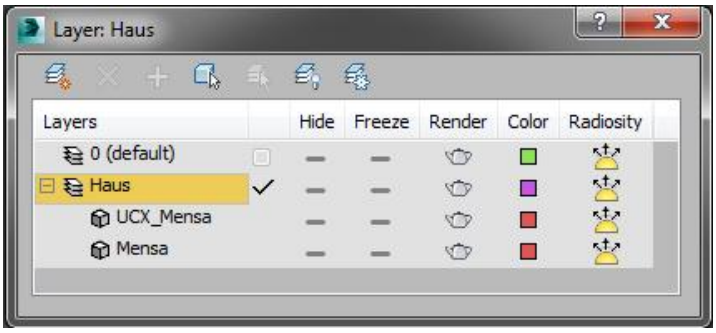


Abbildung 7: Mensa Collision Box

Dieser Zusatz ist für das UDK die sogenannte Collision-Box. Später beim Importieren ist dieses Objekt vom Spieler nicht mehr zu sehen. Es verhindert aber das durchlaufen von Wänden.

3.2.2 Photoshop CS2

Zurück zur Spielfigur Annie. Im Downloadpaket waren die Texturen dieses Models in DDS[17]-Dateiformat. Diese Dateien kann aber das UDK nicht interpretieren. Mit dem AddOn „NVIDIA Texture Tools“ für Adobe Photoshop war es möglich diese Texturen mit Photoshop zu öffnen und zu bearbeiten. In Abbildung 8 sieht man als Beispiel die drei Textur-Dateien für das Gesicht von Annie. Diese wurden als TGA[18]-Datei exportiert. In das UDK wurden diese drei Texturen dann importiert und zu einen

[16] Zusammenstoß-Kisten – Sie dienen zur Erzeugung von Kollisionen, nicht sichtbare Wände

[17] Direct Draw Surface – ein von Microsoft entwickeltes Dateiformat für Texturen

[18] Targa Image File – ist ein Bilder-Dateiformat

Material zusammengefügt, welches dann mittels der Material-ID dem Gesicht des Models zugeordnet wurde.

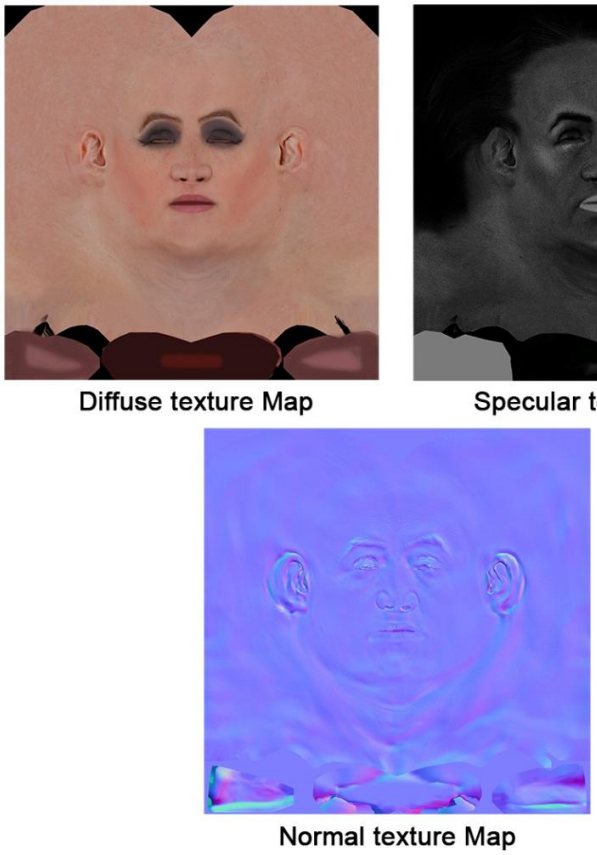


Abbildung 8: Texture Mapping

Diffuse texture Map Diese Textur ist da, um eine Hauptfarbe einer Oberfläche zu definieren. Um ein gutes Ergebnis zu erhalten sollten Schatten oder Beleuchtungen hier nicht vorhanden sein.

Specular texture Map Diese Textur ist für den Glanz und die Glanzlichtfarbe für die Oberfläche verantwortlich. Dunkle Stellen haben weniger Glanz als hellere Stellen.

Normal texture Map Diese Textur ist verantwortlich für den Störungswinkel des einfallenden Lichtes auf einer Oberfläche. Dieses RGB-Farbspiel simuliert eine detailreiche Oberfläche.

(vgl. [7] Splash Damage, Overview)

Ein Beispiel für dieses Material in dem UDK findet man in Abbildung 12.

3.2.3 UDK

Das Zusammensetzen dieser erstellten Puzzleteile wurde mit dem Unreal Development Kit realisiert. In diesem Punkt wird dokumentiert wie dies bewerkstelligt wurde, wie zum Beispiel das Importieren diverser Objekte, das Einbinden der Spielfigur Annie über UnrealScript, das Erstellen der Third-Person-Perspektive über UnrealKismet.

In Abbildung 9 bekommt man einen kleinen Überblick über die Benutzeroberfläche des UDK.

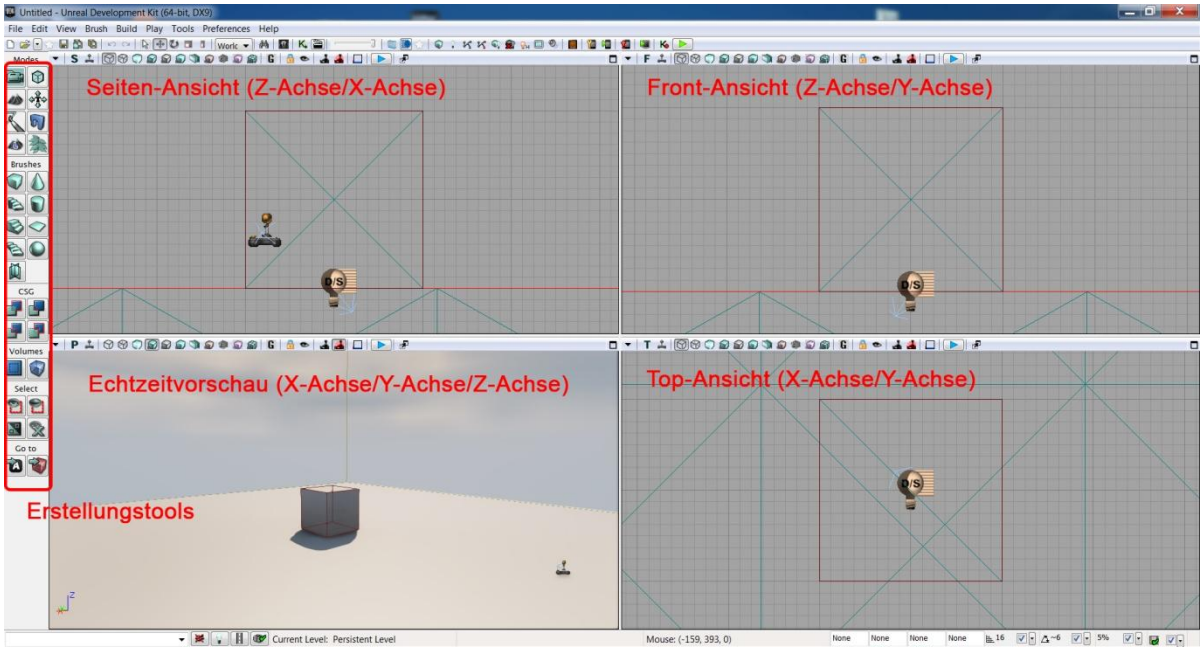


Abbildung 9: UDK Benutzeroberfläche

Auch wie bei 3Ds Max (Beispiel Abbildung 6) besitzt das UDK vier Ansichtsfenster zur Erstellung und Platzierung von Objekten. Sehr zum Vorteil ist die Echtzeitvorschau, wo der Programmierer gleich beim Erstellen der Karte ein vorgerendertes Bild bekommt.

In Abbildung 10 wird kurz der Content Browser vorgestellt.

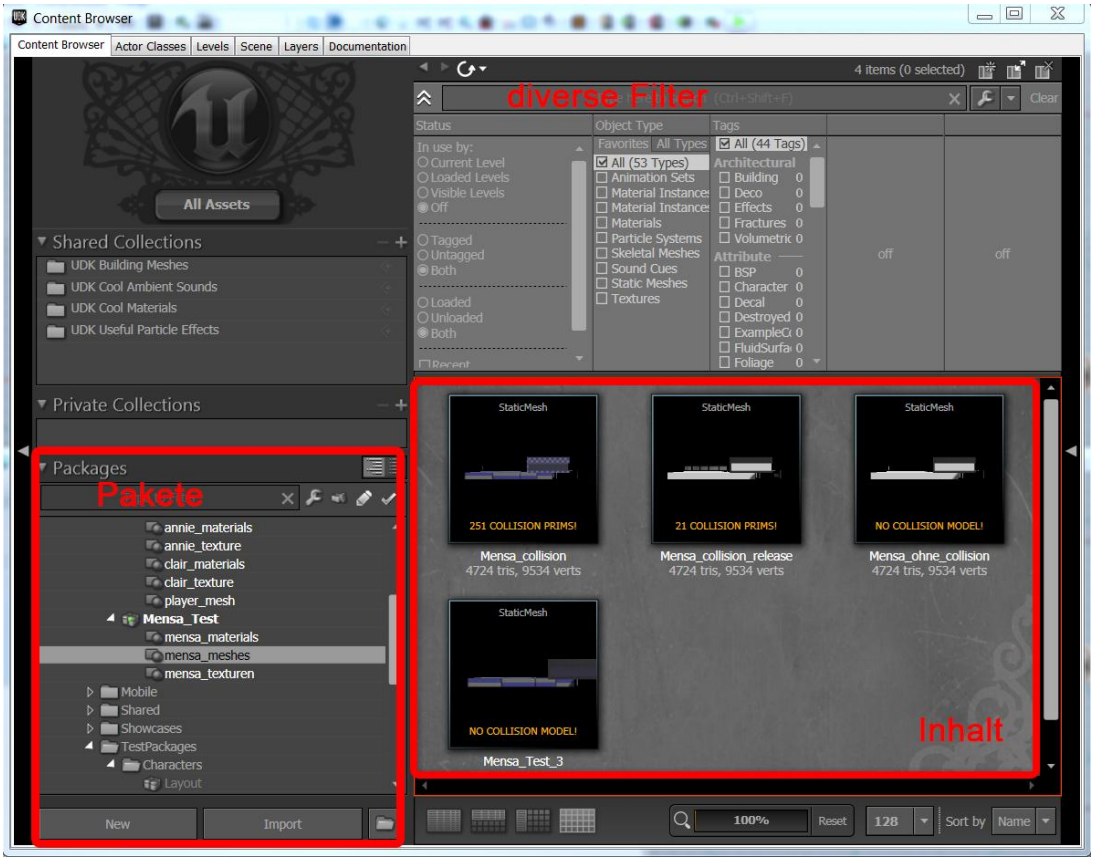


Abbildung 10: Content Browser

Das UDK speichert Objekte, Texturen, Materialien, Animationen etc. in sogenannten Packages (Pakete) ab. Für jedes Spiel was entworfen wird, sollte ein eigenes Paket angelegt werden. Der Inhalt der einzelnen Pakete wird im davor hergesehenen Fenster angezeigt. Das Importieren von Objekten, Texturen etc. erfolgt über diesen Content Browser.

3.2.3.1 Importieren der Objekte

Die exportierten Objekte aus 3Ds Max mussten nun in das UDK integriert werden. Dies bewährte sich zum am Anfang problematisch. Diverse Mesh's besaßen fehlerhafte Polygongitter die zu Löchern im Objekt führten. Ab hier kam das 3Ds Max wieder zum Einsatz. Durch das Entfernen überflüssiger Ecken und Kanten am Objekt, wurde dann nach mehreren Versuchen ein passendes Objekt ohne Polygonfehler im UDK importiert.

Die Zuweisung der gewünschten Materialien funktionierte auch. Aber das zweite Problem was auftrat war die Streckung der Materialien. Die Auflösung der Bilder am Objekt war nicht korrekt. Dieses Problem musste wieder im 3Ds Max gelöst werden. Hierzu musste dem zu bearbeiteten Objekt eine sogenannte „UVW[19] Map“ hinzugefügt werden. Diese ist zu finden über den Reiter „Modify“ und aus der Liste „Modifier List“. Hierbei musste der „Gizmo[20]“ so verändert werden, dass die Materialien die richtige Auflösung haben. Bei den zwei Gebäuden in diesem Projekt, fiel die Wahl auf einen Würfel-Gizmo mit den Seitenlängen von 100 Einheiten. Da die Gebäude zum größten Teil aus geraden Flächen und rechtwinkligen Ecken bestehen. Nach erneutem Exportieren der Objekte aus dem 3Ds Max waren die zwei erstellen Hauptgebäude nun bereit für das endgültige Importieren in das UDK.

Für diesen Prototyp viel die Wahl auf die Mensa und das Haus 1, welche erstellt und grob nachgemodelt wurden. In Abbildung 11 ist der Unreal Static Mesh Editor des UDK zu erkennen. Mit diesem wurde das Importierte Gebäude, die Mensa, geöffnet und angezeigt. In diesem Editor wird auch die Zuweisung der Materialen vorgenommen. Unter den Eigenschaftenpfad LODInfo → [0] → Elements sind die Materialien, in diesem Fall 9 unterschiedliche, zuzuordnen.

[19] UVW sind die 3D-Koordinaten einer Textur auf einem Objekt

[20] Wortwörtlich übersetzt Ding - Ist ein imaginäres Objekt welches ungefähr die Form des zu Texturierenden Objektes haben sollte.

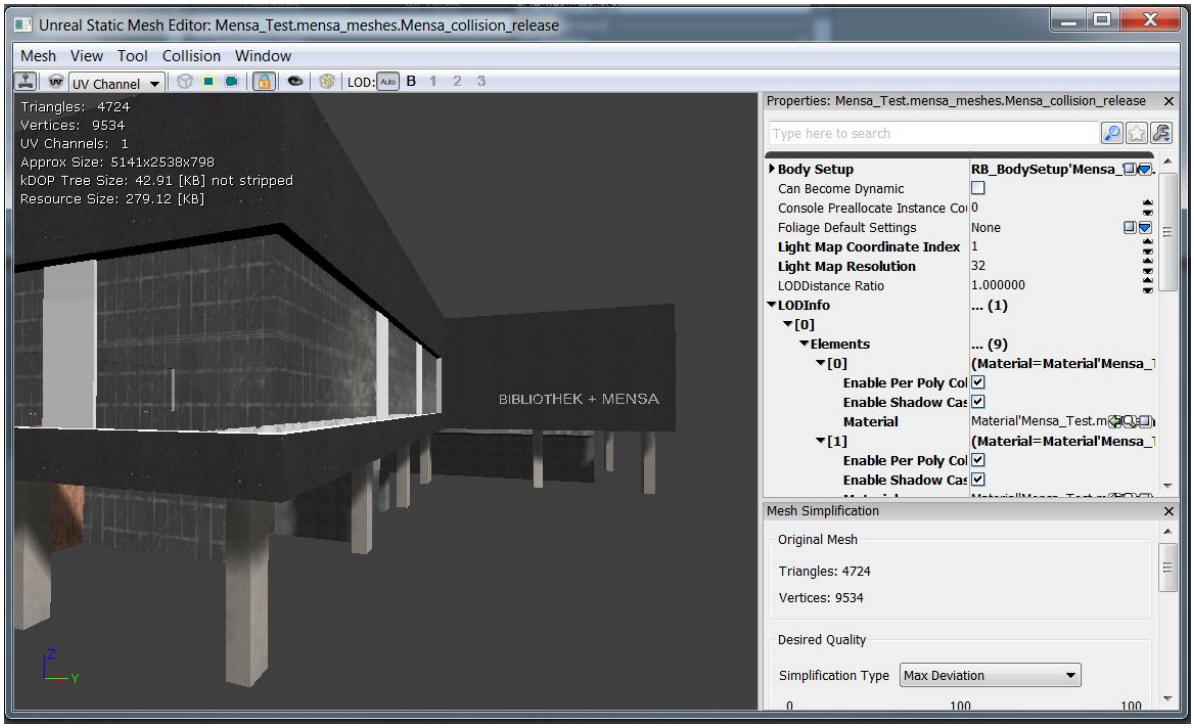


Abbildung 11: UDK Bibliothek + Mensa

Diverse Objekte wie einen Wegweiser (Siehe Weitere Online-Quellen [11], Nicht-kommerzielle Lizenzrechte) und einen Mülleimer (Siehe Weitere Online-Quellen [10], Nicht-kommerzielle Lizenzrechte) wurden ebenfalls mit mehreren Versuchen im 3Ds Max bearbeitet, dann exportiert und im UDK wieder importiert. Das Erstellen der endgültigen Map (Karte/Spielwelt-Umgebung) mit allen Objekten wurde in diesem Zeitpunkt noch nicht bewerkstelligt. Es wurde geplant erst alle wichtigen Eigenschaften in dem UDK einzubinden und separat kurz zu testen. Im nächsten Punkt wird kurz auf das Importieren der Spielfigur eingegangen.

3.2.3.2 Importieren der Spielfigur

Nach mehreren Fehlversuchen beim Importieren von Annie hat es nach geraumer Zeit doch noch funktioniert. Die Material-IDs wurden im FBX-Format nicht mit übernommen, deswegen konnte keine Zuweisung der Materialien erfolgen. Das schon oben genannte Addon „ActorX“ hat dieses Problem gelöst. Die Materialen konnten danach aus den drei Texturen (Diffuse, Specular, Normal) mit den Unreal Material Editor erzeugt werden. Ein Beispiel ist auf Abbildung 12 zu finden.

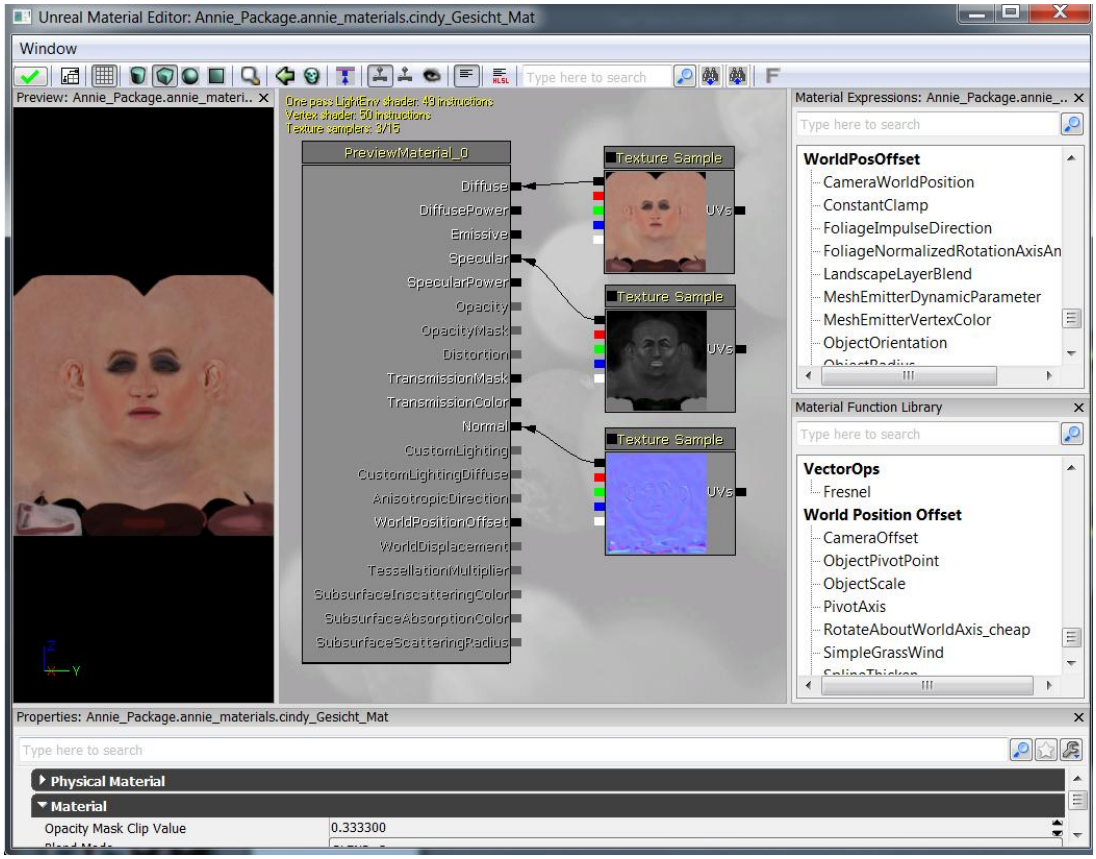


Abbildung 12: Unreal Material Editor

Nachdem Annie in das Unreal Development Kit eingebunden wurde, muss sie als Spielerfigur ausgewählt werden. Diese Aufgabe wurde per UnrealScript gelöst.

3.2.3.3 Unreal Scripting

Das erstellen solcher Scripts ist mit dem herkömmlichen „Editor“ von Windows oder mit dem „TextEdit“ von Mac OS möglich. Aber bei diesen Programmen ist die Syntax des UnrealScript nicht implementiert. Deswegen fiel die Wahl des Editors auf dem ConTEXT. Da dieser kostenlos ist und die Möglichkeit besitzt das UnrealScript zu interpretieren. Aber auch nur mit dem Zusatz Highlighter der offiziellen Homepage.

Als Erstes müssen einige Faktoren im UDK vorgenommen werden um eigenen UnrealScrip-Codes einzubinden. Dazu geht man ins Hauptverzeichnis des UDK und im Dateipfad \UDKGame\Config\ öffnet man die Datei DefaultEngine.ini. Unter dem Abschnitt [UnrealEd.EditorEngine] muss der Ordner des selbsterstellten Scripts

hinzugefügt werden und zwar mit dem Befehl +ModEditPackages=MittweidaGame. Der Ordner wurde bei diesem Projekt „MittweidaGame“ genannt. Hier ist der Abschnitt dieser veränderten DefaultEngine.ini zu sehen.

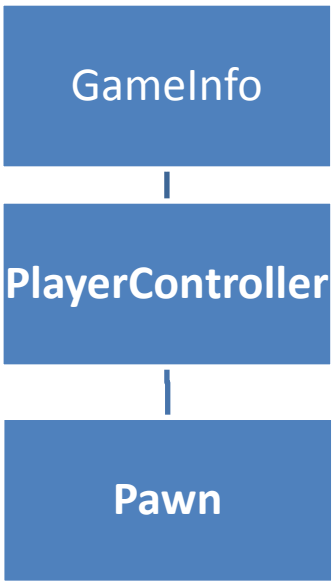
Listing 1: Dateiauszug DefaultEngine.ini

```
…
[Engine.DemoRecDriver]
DemoSpectatorClass=UTGame.DemoRecSpectator

[UnrealEd.EditorEngine]
+EditPackages=UTGame
+EditPackages=UTGameContent
+ModEditPackages=MittweidaGame

[Engine.Engine]
ConsoleClassName=UTGame.UTConsole

…
```

Jetzt werden die eigenen Scripts in diesem Ordner von der UDK ausgeführt.

Nun muss der Ordner „MittweidaGame“ und weitere Unterordner erzeugt werden. Man gehe ins Hauptverzeichnis des UDK und dort ins Unterverzeichnis \Development\Src\ und erstellt den Ordner MittweidaGame. Dieser wird wiederum 739mit einem Unterordner Namens Classes versehen. In diesem Ordner werden die Script abgespeichert. In diesem Fall sind es die drei Dateien mit dem Namen MittweidaGameInfo.uc, MittweidaPawn.uc und MittweidaPlayerController.uc. Sie müssen die Datei-Endung „uc“ besitzen, damit das UDK weiß, dass diese Dateien noch kompiliert werden müssen. Dies geschieht beim nächsten Starten des Unreal Development Kit.

Wie stehen diese drei Klassen zueinander?



Die GameInfo Deklariert welche Art von Spiel es sein soll. Diese greift auf die darunterliegenden Klassen zu und erbt dessen Inhalt beim laden

PlayerController ist die Schnittstelle für die Eingabe des Benutzer und dessen Umsetzung im Spiel. Auch die Änderung der Charakterfigur kann hier erfolgen.

Pawn ist für die Beziehung zwischen Spielfigur und Umfeld verantwortlich. Hier kann man z.B. die Laufgeschwindigkeit der Spielfigur vorgeben.

Abbildung 13: UDK Klassen (vgl. [8] Thorn 2012, S. 630)

Diese Dateien sind auf der mitgelieferten CD zu finden. In der nächsten Abbildung ist ein kurzer Auszug der MittweidaPlayerController.uc. Der Befehl der dafür verantwortlich ist das Spielermodel zu ändern wurde Gelb markiert.

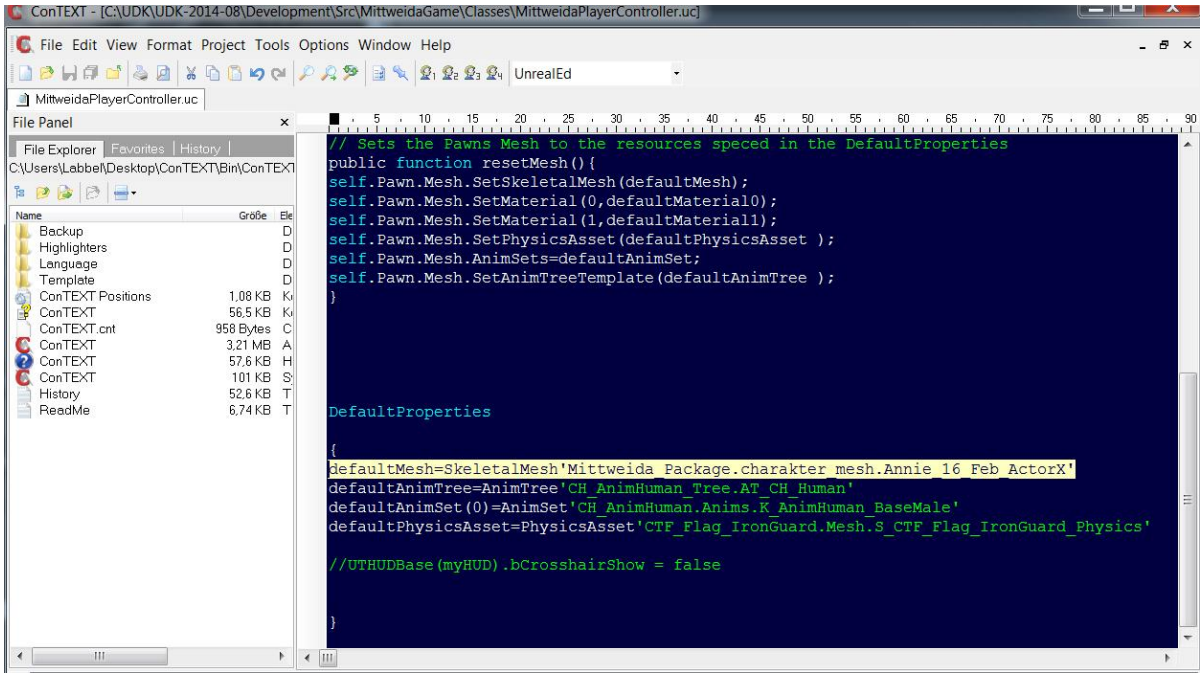


Abbildung 14: Austausch Spielermodel

Nach einigen Startschwierigkeiten und Schreibfehler hat es dann doch noch funktioniert zu kompilieren. Eine Abbildung ist in der mitgelieferten CD zu finden (/Abbildungen/Compile_Charakter.jpg).

3.2.3.4 Third-Person-Perspektive

Die Spieler-Perspektive wurde mit dem UDK internen UnrealKismet geändert. Aber zuvor musste in einer Test-Welt die zu erstellente Perspektive separat erzeugt und kurz getestet werden. Als erstes wurden die zwei benötigten Actor[21] erstellt. Ein Actor PlayerStart und ein Actor CameraActor. Der PlayerStart ist der Startpunkt der Spielfigur in dieser Spielwelt. Der CameraActor ist wie der Name schon sagt eine kleine Kamera mit der es möglich ist im Spiel verschiedene Perspektiven zu erzeugen. Dieser CameraActor wird in dieser Test-Welt mit Hilfe dem XYZ-Koordinaten so positioniert, sodass die Blickrichtung von hinten auf dem PlayerStart erzeugt wurde. Nun kam das UnrealKismet zum Einsatz. Siehe Abbildung 15.

Als erstes wird das Event Level Loaded erstellt. Dieses Event Startet sämtliche Aktionen wenn die Test-Welt geladen wurde. Nun mussten die zwei Actor, PlayerStart und CameraActor miteinander verbunden werden. Dazu wurde eine neue Aktion erstellt namens Attach to Actor. Diese Aktion wird aber erst ausgeführt wenn es eine Verbindung zwischen den Event Level Loaded (Loaded and Visible) und der Aktion Attach to Actor (In) gibt. Es wird zwei Variablen benötigt. Einmal die Variable Player und die Variable Objekt using CameraActor. Der Player Variable wird der PlayerIndex `0` zugewiesen. Das ist die Spielfigur in dieser Karte. Diese Player Variable wird als Target (Ziel) der Attach to Actor Aktion zugewiesen und das Objekt der Kamera den Attachment (Ansatz). Das die Kamera aber der Spielfigur beim Laufen folgt wird noch eine weitere Aktion benötigt: die Set Camera Target. Diese Aktion musste natürlich auch mit den Event Level Loaded (Loaded and Visible) verbunden werden. Als Target wurde wieder die Variable Player zugewiesen und als Cam Target (Kamera Ziel) das Objekt der Kamera.

[21] Wortwörtlich übersetzt Schauspieler

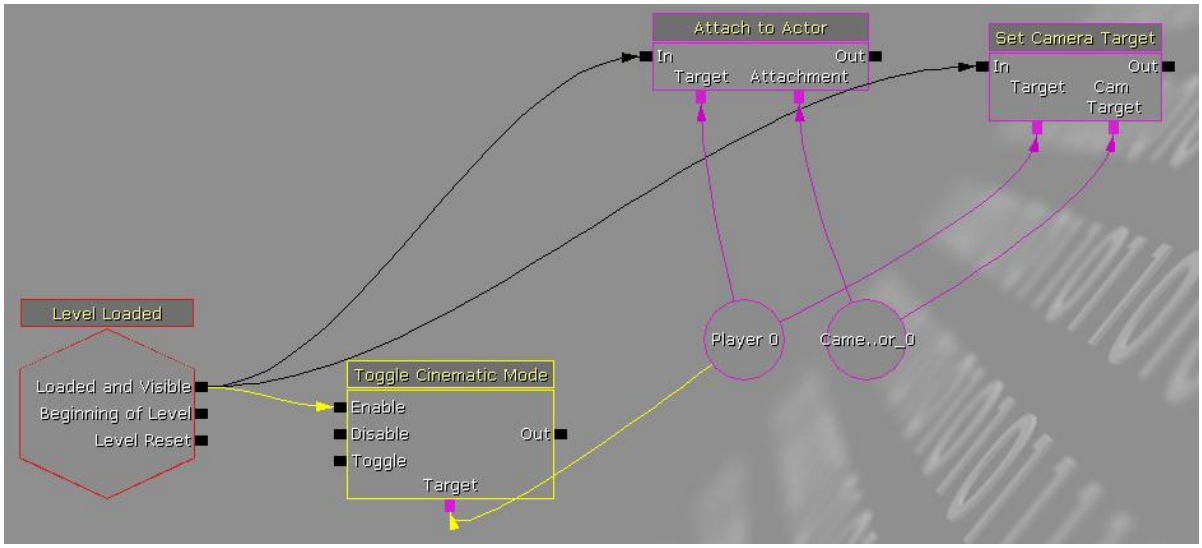


Abbildung 15: UnrealKismet

Ein kleiner Test zeigte, dass die besagte Perspektive erstellt wurde. Ein Beispiel ist in Abbildung 16 zu sehen. Einen kleinen Makel gibt es mit dieser Methode trotzdem. Das freie Umschauen in der vertikalen Richtung ist leider nicht möglich. Was aber in diesem Projekt erst einmal vernachlässigt wurde.

Als kleinen Zusatz kam dann noch die Aktion Toogle Cinematic Mode hinzu. Als Target wurde die Player Variable gewählt und das Enable (Aktivieren) mit dem Level Loaded (Loaded and Visible) Event verbunden. Mit dieser Aktion ist es möglich das Standard HUD[22] des UDK auszublenden. Bei diesen Prototypen ist es uninteressant wie viel Gesundheit und Munition der Spieler noch besitzt. Es hat inhaltlich nicht dazu gepasst.

3.2.3.5 Zusammenfügen der Einzelteile

Es musste ein neuer Level im UDK erstellt werden. Beim Erstellen einer neuen Map besitzt das UDK vier vorgefertigte Licht Templates wo man sich die Tageszeit wählen kann (Afternoon Lighting, Midday Lighting, Morning Lighting, Night Lighting). Oder man wählt einfach eine „Blank Map“ aus, welche komplett leer ist. Die Wahl

[22] Head up Display - Wortwörtlich übersetzt „Kopf oben Anzeige“ – ist eine feste, grafische Komponente die Informationen für den Spieler beinhaltet

bei diesem Prototyp fiel auf das Midday Lighting Template, da sich die Story des Spiels ungefähr um die Mittagszeit abspielt. Unter den Einstellungen View → World Properties ist im Abteil Game Type der Spieletyp zu ändern. In diesem Fall wurde unter den Eigenschaften Default Game Type und Game Type for PIE das selbst erstellte Script MittweidaGameInfo ausgewählt, um die selbsterstellte Spielfigur einzubinden.

Der erste kleine Test vor dem Zusammensetzen aller Komponenten beinhaltet das Zusammenbringen der Third-Person-Perspektive und dem ersten Gebäude die Mensa. Um zu schauen ob auch alles so funktioniert wie erwartet. (siehe Abbildung 16). Das HUD ist ebenfalls bei diesem Test ausgeblendet.

Abbildung 16: Annie und Mensa

Ab diesem Zeitpunkt konnten die ganzen Puzzleteile in einem Level in mühseliger Kleinstarbeit zusammengesetzt werden.

Nachdem die zwei Hauptgebäude, Straßen, Wegweiser, Mülleimer, Bäume, Felsen, Platzhalter etc. platziert wurden, war der nächste Schritt die Programmierung des Story-Ausschnitts. Der PlayerStart wurde direkt vor dem Ausgang der Mensa

gesetzt, mit dem Blick in Richtung „Bibliothek + Mensa“ Schriftzug. Wenn der Spieler sich in Richtung Wegweiser bewegt soll eine Kamerafahrt-Animation ausgelöst werden. Hierfür musste ein sogenanntes TriggerVolume[23] erstellt werden. Dieser Körper ist ein Zylinder wo der Mittelpunkt in der Nähe des Wegweisers sitzt. Betritt der Spieler diesen Radius wird dieser Schalter ausgelöst. Als zweites wurde ein CameraActor platziert. Nachdem das TriggerVolume markiert wurde kam wieder das UnrealKismet zum Einsatz. Im UnrealKismet wählte man nun New Event Using TriggerVolume_[Indexnummer] um den Schalter zu Programmieren. Nun wurde ein New Matinee gesetzt. Mit der UnrealMatinee kann man Animationen erstellen. UnrealMatinee wurde geöffnet und eine neue Kamera Gruppe erstellt via Add New Camera Group. Diese Gruppe beinhaltet die Movement und FOVAngle Eigenschaften. Im Movement Zweig werden die Bewegungen der Kamera mittels Hotkeys gespeichert. Der CameraActor wird hierfür immer so positioniert wie gewünscht und ein Hotkey im Movement Zweig mit der Tastatur Eingabe „Enter“ erstellt. Nun die gewünschte Länge wählen und die nächste Kameraposition einnehmen und mit einen weiteren Hotkey bestätigen. Dies wurde so oft wiederholt bis die Endposition der Kamera vorhanden war. Die FOVAngle Eigenschaft ist lediglich für den Zoom der Kamera verantwortlich. Das die erstellte Kamerafahrt auch richtig funktionierte, musste ein Wechsel von der Spielfigur-Kamera zur Animations-Kamera erstellt werden. Dies ging über das Hinzufügen eines Direktors via Add New Director Group. Dieser Director Track musste wieder mit „Enter“ bestätigt werden und hier wurde dann die gewünschte Kamera gewählt zu der geschalten werden soll.

Im UnrealKismet sind jetzt der CameraActor und die erstellte Animations-Sequenz der Matinee zugewiesen. Das diese Animation ausgelöst wird beim betreten des Schalter-Körper mussten der TriggerVolume (Touched) mit der Matinee (Play) verbunden werden. Um ein sinnloses herumlaufen während der Animationen zu verhindern und um Fehler vorzubeugen, wurde die Tastatur- und Mauseingabe deaktiviert mithilfe der Toogle Input Aktion. Betritt der Spieler diesen Schalter wird die Eingabe deaktiviert und kurz danach die Animation ausgeführt. Ist diese dann zu

[23] Schalter Körper

Ende muss die Tastatur- und Mauseingabe wieder freigegeben werden mit Hilfe eines weiteren Toogle Input. TriggerVolume (Touched) mit dem Toogle Input (Turn Off) verbinden und den Toogle Input (Out) mit dem Matinee (Play) verbinden. Nun musste noch die Matinee (Completed) mit dem Toogle Input (Turn On) verbunden werden. Den beiden Toogle Input musste noch die Player Variable als Target zugefügt werden, dass auch die gewünschte Spielfigur deaktiviert wird. Eine Abbildung dieser kompletten Syntax ist in der mitgelieferten CD zu finden (/Abbildungen/Kismet_all.jpg) da diese Abbildung in dieser Arbeit kaum lesbar wäre.

4 Testphasen und Änderungen

Die Ausgabe und Installation des Prototyps auf einem Mobilen Endgerät wurde geplant, ist aber leider nicht zu realisieren gewesen, aus Mangel der benötigten Hardware. Deswegen wurde diese kleine Applikation auf einem Personal Computer getestet und ausgewertet.

Die Testphase wurde in zwei unterschiedlichen Varianten durchgeführt. Die ersten Testpersonen haben diesen Prototyp direkt am Ersteller-PC getestet. Das Spiel wurde direkt aus dem Unreal Development Kit gestartet. Der zweite große Test war das Erstellen der Installationsdatei des Prototyps, um diesen auf einen externen PC zu installieren und zu testen.

4.1 Erster Test

Die erste Testphase fiel recht positiv aus. Die Testpersonen waren zwar keine Studenten, aber die Schwierigkeit der Orientierung in einer fremden und ungewohnten Umgebung war ihnen bekannt. Wie zum Beispiel das Umziehen in einem anderen Ort oder der Wechsel einer neuen Arbeitsstelle. Die erste Kamerafahrt im Spiel, die in Richtung Wegweiser verläuft und auf das Haus 1 verweist, wurde von den Testern auch so interpretiert den Weg dorthin zu suchen. Was den Testern als erstes Aufgefallen ist und gestört hat, war das fehlende freie Umschauen mit der Maus. Dies wurde im Nachhinein im Unreal Development Kit geändert. Die erstellte Third-Person-Perspektive mithilfe des UnrealKismet wurde von Grund auf neu erstellt. Der benötigte CameraActor wurde gelöscht. Entfernt wurden auch die Attach to Actor, Set Camera Target Aktionen und das Camera Objekt. Das Level Loaded Event wurde mit dem PlayerSpawned Event ausgetauscht. Der Unterschied hierbei ist, dass die gewünschten Aktionen jedes Mal ausgeführt werden, wenn die Spielfigur neu ins Spiel gesetzt wird und nicht nur wenn der Level geladen wurde. Mit Hilfe eines einzigen Consolen[24]-Befehls konnte man die Third-Person-Perspektive einigermaßen Erzeugen. Zum öffnen dieser

[24] Console – ist eine Zeichenketten basierende Eingabeaufforderung im UDK.

Console muss während des Spielens die Tabulatoren Taste gerückt werden. Mit dem Befehl “BehindView“ wurde die Spielfigur nun von hinten gezeigt. Um diesen Befehl nicht jedes Mal beim Start eingeben zu müssen, musste eine neue Aktion im UnrealKismet hinzugefügt werden. Diese Aktion nennt sich Console Command. Unter diesem Console Command wurde unter Seq Act Console Command→Commands→[0] der Befehl “BehindView” eingetragen. Eine Verbindungen des Event PlayerSpawned (Out) mit der Aktion Console Command (In) musste erstellt werden und das Objekt Player musste wiederum dem Console Command (Target) zugewiesen werden. Nur wird dieser Befehl jedes Mal ausgeführt, wenn die Spielfigur den Level betritt. Eine komplette UnrealKismet-Abbildung ist auf der beigefügten CD zu finden unter /Abbildungen/Kismet_all.jpg.

Die fehlende Laufanimation der Spielfigur störten die Tester. Auch die Standard Sounds des UDK waren noch vorhanden. Auf diese kleinen optischen und akustischen Makel wurde bei der Entwicklung des Prototyps nicht näher eingegangen. Auch grafisch ist dieser Prototyp nicht zu vergleichen mit aktuellen Computerspielen. Es gibt diverse Grafikfehler durch die Verwendung des Glas-Materials bei den beiden Hauptgebäuden. Es sollte lediglich gezeigt werden wie solch eine Applikation aussehen könnte. Auch die Gebäude-Platzhalter wurden von den ein oder anderen Tester verwirrend aufgenommen. Im Prototyp sind das die großen blau-weißen Quadrate. Eine Demonstration dieses Tests ist auf der mitgelieferten CD als Video zu finden (UT_Mittweida_Test1_Video.mp4).

4.2 Zweiter Test

Der zweite Test umfasste das Zusammenbringen der gesammelten Packages und der erzeugten Karte. Sodass eine eigenständige Installationsdatei des Spiels entstehen konnte. Dieses wurde mit den UDK internen Unreal Frontend realisiert. Mit diesem Tool ist das erstellen eines Installationsprogramm für ein PC-Spiel oder einer Apple iPhone-Applikation möglich. Hierfür musste die erstellte Karte mit dem Namen „Mittweida_Game.udk“ in dem Unterverzeichnis /UDK/UDKGame/Content/Maps/Bachelor_Mittweida/ gespeichert werden. Auch die benötigten Packages worin die importieren Objekte gespeichert wurden, müssen in

dem /UDK/UDKGame/Content/ Ordner liegen. Nun musste im Unreal Frontend ein Profil ausgewählt werden. Dieses Profil wurde geklont und umbenannt. Nun musste die gewünschte Map ausgewählt werden die im Spiel enthalten sein sollte. Über Script → Full recompile wurden die sämtlichen Scripts noch einmal kompiliert. Nachdem dies fertig war musste das Paket „gekocht“ werden über Cook → Clean and Full Recook. Als das beendet war konnte das Spiel über Package Game → Package Game zusammengefasst werden. Die erzeugte Installationsdatei ist im Hauptverzeichnis des UDK zu finden. Diese Datei wurde auf einem zweiten Computer der nichts mit der Erstellung dieses Prototyps zutun hatte installiert und ausgeführt. Die Installation verlief problemlos und ohne Komplikationen.

Das Menü des Prototyps ist das Standard-Menü des Unreal Development Kits, welches visuell nicht wirklich passend für den Prototyp ist. Es konnten verschiedene Spielmodi ausgewählt werden, wie zum Beispiel ein Multiplayerspiel, ein Deathmatch Game oder ein Team-Deathmatch. Diese Wahl ist bei diesem Prototyp nicht relevant und wurde entfernt. Es wurden lediglich die „Verknüpfungen“ dieser Auswahl gelöscht. Um dies zu ändern musste die Datei UDKUI.ini im Ordner \UDK\UDKGame\Config\ verändert werden. Es wurden die überflüssigen ListOptions entfernt (In Listing 2 sind diese rot markiert). Die ListOptions sind Verweise auf die unterschiedlichen Spielmodi. Diese Modi sind noch vorhanden, nur werden diese im Hauptmenü des Spiels nicht mehr angezeigt.

Listing 2: Alter Dateiauszug UDKUI.ini

```
…

[UTGame.GFxUDKFrontEnd_MainMenu]
ViewTitle="MAIN MENU"
ListOptions=(OptionName="InstantAction",OptionLabel="INSTANT ACTION",OptionDesc="Jump right into the action with some bots.")
ListOptions=(OptionName="Multiplayer",OptionLabel="MULTIPLAYER",OptionDesc="Host or join a multiplayer game.")
ListOptions=(OptionName="Exit",OptionLabel="EXIT",OptionDesc="Exit to the desktop.")

[UTGame.GFxUDKFrontEnd_InstantAction]
ViewTitle="INSTANT ACTION"
```

```
ListOptions=(OptionName="GameMode",OptionLabel="GAME MODE",OptionDesc="Change the game type.")
ListOptions=(OptionName="MapSelect",OptionLabel="MAP",OptionDesc="Change the field of battle.")
ListOptions=(OptionName="Settings",OptionLabel="SETTINGS",OptionDesc="Modify the game settings.")
ListOptions=(OptionName="Mutators",OptionLabel="MUTATORS",OptionDesc="Configure the mutators for this match.")
ListOptions=(OptionName="StartGame",OptionLabel="START GAME",OptionDesc="Launch the match.")

…
```

Bei den übrigen ListOptions wurden die OptionLabel und OptionDesc dem Prototyp angepasst. (Die Änderungen sind in Listing 3 rot markiert)

Listing 3: Neuer Dateiauszug UDKUI.ini

```
...

[UTGame.GFxUDKFrontEnd_MainMenu]
ViewTitle=MAIN MENU
ListOptions=(OptionName="InstantAction",OptionLabel="Mittweida Spiel",OptionDesc="Springe ins Spiel")
ListOptions=(OptionName="Exit",OptionLabel="Beenden",OptionDesc="Zurück zum Desktop.")

[UTGame.GFxUDKFrontEnd_InstantAction]
ViewTitle=INSTANT ACTION
ListOptions=(OptionName="StartGame",OptionLabel="Starten",OptionDesc="Beginne das Spiel.")

…
```

Nach dem dieses Spiel erneut kompiliert und „gekocht“ wurde, musste eine neue Installationsdatei erstellt werden. Danach wurde dieses Spiel wieder auf dem zweiten PC installiert und ausgeführt. Die Karte wurde geladen und geöffnet. Der “BehindView” Befehl wurde richtig ausgeführt und man hatte die gewünschte Third-Person-Perspektive. Das größte Problem was auftrat war der Tausch der Spielfigur. In dieser Testphase wurde die Spielfigur Annie nicht dargestellt. Stattdessen wird ein Roboter mit einer Waffe in der Hand dargestellt. Dies ist zwar visuell für diesen

Prototyp sehr unpassend, aber es ist trotzdem möglich die Umgebung der Karte zu Erkunden. Eine Lösung dieses Problem konnte leider nicht gefunden werden. Die Kamerafahrt-Animationen wurden wie gewünscht dargestellt. Auch diverse Licht/Schatten Fehler waren zu erkennen. Im Startmenü sind noch einige Einträge die verwirrend wirken können, da es das Standard-Menü des UDK ist. Manche Einträge wie z.B. „Deathmatch“ waren ohne weiteres nicht so einfach zu beheben. Die Erstellung eines eigenen individuellen Menüs ist möglich. Welches aber ein weiteres Drittanbieterprogramm (Flash-Programmierung) benötigt zur Erzeugung. Aus zeitlichen Gründen wurde das Standard-Menü des UDK beibehalten.

Im Großen und Ganzen fielen die Testphasen positiv aus.

5 Zusammenfassung

5.1 Ausblick

Es besteht Potenzial dieses Projekt weiter zu entwickeln. Man müsste die Anzahl der Mitarbeiter um das vielfache erweitern. Auf Grund der begrenzten Zeit konnte das Projekt nur in einem relativ kleinen Umfang durchgeführt werden. Als Studentenprojekt könnte man dieses ausarbeiten und erweitern. Separate Arbeitsgruppen müssten erstellt werden. Ein Team welches nur für das Modeln der Gebäude zuständig wäre. Eine Gruppe die sich um das UnrealScripting kümmert und so weiter.

Eine kleine Marktanalyse könnte erstellt werden, um zu schauen ob solch ein Orientierungsspiel überhaupt beim Endverbraucher Anklang findet. Danach könnte eine Fertigstellung und offizieller Vermarktung im deutschen Raum dieses Spiels Erfolgen. Dies wäre für die Hochschule Mittweida ein gutes Marketingkonzept und würde ihr positives Image unterstreichen.

5.2 Resümee und kritische Betrachtung

Das Ziel dieser Arbeit war es eine Konzeption und prototypischen Umsetzung eines Orientierungsspiel mithilfe des Unreal Development Kit zu entwickeln und zu erstellen. Es wurde ein kleiner Ausschnitt des Konzepts erarbeitet. Es musste eine Wahl der benötigten Programme getroffen werden. Auch eine grobe Reihenfolge der zu erstellenden Einzelprojekte für Gebäude etc. musste für die prototypische Umsetzung geplant werden. In diesem Konzept ist eine Geschichte samt Dialoge enthalten. Diese Story wäre für eine Vollversion des Orientierungsspiels ein geeignetes Startkapitel.

Bei diesem Projekt wurde nur ein kleiner Ausschnitt dieses Spiels realisiert und in einem Prototyp verwirklicht. Der enorme Umfang einer Videospielproduktion ließ dies leider nicht zu es weiter auszubauen. Trotz alledem ist es gelungen einen kleinen Ausschnitt des Campus in Mittweida nachzubilden und virtuell darzustellen.

Einige diverse Fehler und Probleme die entstanden sind, konnten aber dank der Testphasen behoben werden wie z.B. die Third-Person-Perspektive oder das Fehlen der Collision-Box. Diverse andere Fehler konnten stattdessen nicht behoben werden wie z.B. in Testphase 2 die falsche Anzeige der Spielfigur oder einige Lichtprobleme. Aber für eine Demonstration eines Orientierungsspiels in dieser Art, konnten diese Fehler erst einmal hingenommen werden.

Ob viele Menschen eine solche Applikation überhaupt verwenden würden steht noch im Raum. Es gibt einige Personen die sich lieber Vorort einen Eindruck ihrer neuen Umgebung machen wollen. Die meisten Endbenutzer solcher Orientierungsspiele wären höchstwahrscheinlich auch Menschen die in ihrer Freizeit Videospiele spielen.

Diese Arbeit weist außerdem auf eine noch kaum erforschte und ausbaufähige Methode hin sich in einer neuen Umgebung auf spielerischer Art und Weise zu orientieren. Mit einer Kombination der Oculus Rift und solch ein Orientierungsspiel würde diese Methode für den Markt sehr interessant sein.

LITERATURVERZEICHNIS

[1] Blow, Jonathan: Game Development: Harder Than You Think. Band 1 Ausgabe 10, Februar 2004

[2] Cordone, Rachel: Unreal Development Kit Game Programming with UnrealScript: Beginner's Guide. Verlag Packt Publishing, Dezember 2011

[3] Lewalter, Udo: Schöpfer David Cage setzt erzählerisch mit "Heavy Rain" zwar Maßstäbe – scheitert aber an seiner eigenen Ansprüchen. 17.02.2010, http://www.computerbild.de/artikel/cbs-Tests-PS3-Heavy-Rain-Review-5081188.html (gelesen am 24.02.2015)

[4] Lorber, Martin. Electronic Arts PR Director GSA und Jugendschutz: EA Magazin: Für Digitale Spielkultur. Ausgabe 03_08

[5] Lorber, Martin: Onlinebericht: Ein Computerspiel entsteht – Von der Idee bis zum Spielvergnügen. 25.10.2011, http://spielkultur.ea.de/themen/gesellschaft-und-kultur/ein-computerspiel-entsteht-von-der-idee-bis-zum-spielvergnugen (gelesen am 11.02.2015)

[6] Mittler, Patrick: Die wichtigsten Spiele-Engines – Hinter den Hits. 26.12.2013, http://www.gamestar.de/specials/spiele/3031120/die_wichtigsten_spiele_engines.html (gelesen am 09.02.2015).

[7] Splash Damage, Basic Texture Overview: http://wiki.splashdamage.com/index.php/Basic_Texture_Overview (gelesen am 26.02.2015)

[8] Thorn, Alan (2012): UDK Game Development; veröffentlicht: Boston, Mass. : Course Technology PTR, c2012

Weitere Online-Quellen:

[9] 3D Model: Annie 3d model, heruntergeladen (am 09.02.2015) von

http://tf3dm.com/3d-model/annie-49170.html, eingereicht von 3dregenerator, zur Verfügung gestellt von luxox_18

[10] 3D Model: Trash 3D Model, heruntergeladen (am 18.02.2015) von http://archive3d.net/?a=download&id=b77f9980, eingereicht von Fidel

[11] 3D Model: Guide-board 3D Model, heruntergeladen (am 18.02.2015) von http://archive3d.net/?a=download&id=90fca619, eingereicht von Artem Karapetyan

ANHANG A: Verwendete Software

[1] Unreal Development Kit – August 2014 – heruntergeladen von
https://www.unrealengine.com/previous-versions

[2] Autodesk 3ds Max – Student Edition 2014 – heruntergeladen von
http://www.autodesk.com/education/free-software/3ds-max
AddON: ActorX – heruntergeladen von
http://www.gildor.org/down/39/actorx/ActorX_All.zip

[3] ConTEXT
AddOn: UnrealScript – heruntergeladen von
http://www.contexteditor.org/highlighters/

[4] Adobe Photoshop CS2 – Student Edition
AddOn: NVIDIA Texture Tools for Adobe Photoshop – heruntergeladen von
https://developer.nvidia.com/nvidia-texture-tools-adobe-photoshop

ANHANG B: Inhalt der mitgelieferten CD

[1] die vorliegende Arbeit als PDF-Datei

[2] Quellcode der Klassen *MittweidaGameInfo.uc*
MittweidaPawn.uc
MittweidaPlayerController.uc

[3] Hauptgebäude (3Ds Max) *Haus1_mit_Collision.max*
Mensa_mit_Collision.max

[4] diverse Abbildungen

[5] Spielfigur Annie *Annie_16_Feb_ActorX_X_ModelInfo.log*
Annie_16_Feb_ActorX.psk
Annie_release_3DsMax.max

[6] Testphase 1 Video *UT_Mittweida_Test1_Video.mp4*

[7] installierter Prototyp im Ordner (Prototyp_Mittweida_Campus)

Hinweise:

Zum Ausführen des Prototyps muss die Applikation ***/Prototyp_Mittweida_Campus/Binaries/Win32/UT_Mittweida.exe*** gestartet werden. Um die beste Performanz des Prototyps zu gewährleisten wird Empfohlen das der Ordner ***Prototyp_Mittweida_Camus*** auf der internen Festplatte des Test-PC kopiert wird. Da das Starten direkt von CD aus, durch die Lesegeschwindigkeit einer CD-Rom, gedrosselt wird. Wenn es zu Komplikationen beim Starten des Prototyps kommt, dann bitte Microsoft DirectX 9.0c und Redistributables für Visual Studio 2010 installieren. Dieses Packet ist auf der mitgelieferten CD unter ***/Prototyp_Mittweida_Campus/Binaries/Redist/UE3Redist.exe*** zu finden. Zum Bewegen dieser Spielfigur wird eine Tastatur und Maus benötigt. Mit der Maus kann sich der Spieler umsehen und mit der Tastatur (W,S,A,D oder Pfeiltasten) bewegen.

Da in dieser Arbeit nur teilweise Code-Ausschnitte abgebildet wurden, sollen diese Dateien als Ergänzung der fehlenden Codezeilen dienen. Die beigefügten Dateien (*.uc) beinhalten lediglich den Quellcode der Klassen. Diese sind ohne das UDK nicht ausführbar.

Diverse Abbildungen sind enthalten die die Dokumentation dieser Arbeit bildlich unterstreichen sollen.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Chemnitz, den 14.03.2015

Robert Marass