Serious Gaming

Lernen in virtuellen 3D-Umgebungen

Eine Projektarbeit im Fach Multimedia Engineering

HS Karlsruhe

Kommunikation und Medienmanagement, Master

Sommersemester 2013

Betreuender Dozent: Prof. Martin Schober

Maximilian Borgartz (42508, boma1026@hs-karlsruhe.de)

Tobias Noeske (42283, noto1012@hs-karlsruhe.de)

Inhaltsverzeichnis

1. Einleitung/Vorwort

> *Sage es mir, und ich werde es vergessen. Zeige es mir, und ich werde es vielleicht behalten. Lass es mich tun, und ich werde es können.“* (Konfuzius)

In einer Informationsgesellschaft, die zum einen geprägt ist von dem Bedarf nach Wissensvermittlung bzw. -aneignung und in der zum anderen der Umgang mit digitalen Medien für immer mehr Menschen selbstverständlich wird, muss ein digitales Instrument, dass das Lernen gezielt unterstützt, von größtem Interesse sein.

Bisher galten digitale Lernspiele, sogenannte „Serious Games“ noch als Nischensegment in dem 1,5 Mrd € schweren Markt für Computerspiele. Seit 2009 jedoch wächst die Bedeutung dieser elektronischen Spiele mit edukativem Charakter kontinuierlich. Der Begriff Serious Game bezieht sich auf Spiele, die in einer hard- und softwarebasierten virtuellen Umgebung stattfinden und die erwünschtes Lernen anregen wollen. Digitale Lernspiele werden typischerweise im Bildungs- und Ausbildungssystem eingesetzt. Der Spaß am Spiel stellt aufgrund seiner motivierenden Funktion zwar ebenso wie bei unterhaltungsorientierten Computerspielen eine zentrale Komponente des Serious Games dar, ihr primäres Ziel ist aber der Erwerb von Wissen und Kompetenzen. Serious Games werden, so glauben Experten, unsere Lernkultur nachhaltig verändern, nicht nur in der betrieblichen Aus- und Weiterbildung, sondern auch an Schulen und Universitäten (1).

Serious Games vereinen in sich die Elemente des Spiels mit modernen Lernmethoden, die auf den Erkenntnissen der Kognitionspsychologie beruhen. Hierbei steht die Frage im Vordergrund, wie Menschen Informationen erwerben, verarbeiten und speichern: Untersuchungen zeigen, dass Menschen durch die aktive Anwendung von Wissen lang anhaltender und leichter lernen. Durch das Zusammenführen von verschiedenen Themengebieten kann ein Spieler umfassende, themenübergreifende Erkenntnisse sammeln. Unterstützt wird dieser Lerneffekt durch den Spaß am Spiel. Der Spieler ist sich der Wissensvermittlung nicht direkt bewusst, lernt also passiv durch aktives Handeln. Der didaktische Aspekt wird für den Spieler nicht in den Vordergrund gerückt, stattdessen begleitet der Lernprozess das unterhaltsame Spielerlebnis eher unterschwellig. Die Lernsimulation greift beispielsweise lebensnahe Situationen aus dem Alltag des Spielers auf, in denen es Konflikte oder Fragestellungen zu lösen gilt. Oder es ermöglicht dem Rezipienten, sich mit komplexen Vorgängen und Verfahren aus der Berufswelt vertraut zu machen. So lernt der Spieler Zusammenhänge und Prozesse unmittelbar zu verstehen, indem er sie nicht nur passiv aufnimmt, sondern im Spiel selbst aktiv mit ihnen arbeitet.

2. Theoretische Grundlagen

2.1. Didaktik

Anders als klassische E-Learning-Ansätze wie z.b. Computer Based Trainings (Präsentationssysteme bzw. tutorielle Systeme) beziehen Serious Games didaktische Erkenntnisse aus der Entwicklung von Computerspielen in ihre Methodik mit ein, um Lerninhalte unterhaltsam, effizient und anwenderfreundlich zu vermitteln. Der Spieltrieb des Anwenders und sein Verlangen, sich mit anderen im Wettbewerb zu messen, werden dabei bewusst genutzt, um den Lernenden intrinsisch zu motivieren und nachhaltig für den Lernstoff zu begeistern. Zentrale Herausforderung für den Entwickler eines Serious Games ist es hierbei, einen wohldosierten Schwierigkeitsgrad zu gewährleisten, sodass sich der Lernende weder unter- noch überfordert fühlt und so dank regelmäßiger Erfolgserlebnisse das Verlangen verspürt, die Lernumgebung immer weiter zu erkunden. Indem er sich Wissen spielerisch aneignet und dieses aktiv beim Lösen von Aufgaben einsetzt, taucht der Anwender unversehens immer tiefer in die Spielewelt ein und erlebt das, was die Lernpsychologie als „Flow“ beschreibt (2).

2.1.1. Flow-Erleben

Der ungarische Psychologe Mihály Csíkszentmihályi unterteilt dieses Flow-Erleben in 6 Komponenten, die für die Konzeption des Schwierigkeitsgrads eines Serious Games von zentraler Bedeutung sind:

1. Passung zwischen Fähigkeit und Anforderung. Man fühlt sich optimal beansprucht und hat trotz hoher Anforderung das sichere Gefühl, das Geschehen noch unter Kontrolle zu haben.
2. Handlungsanforderungen und Rückmeldungen werden als klar und interpretationsfrei erlebt, sodass man jederzeit und ohne nachzudenken weiß, was jetzt als richtig zu tun ist.
3. Der Handlungsablauf wird als glatt erlebt. Ein Schritt geht flüssig in den nächsten über, als liefe das Geschehen gleitend wie aus einer inneren Logik
4. Man muss sich nicht willentlich konzentrieren, vielmehr kommt die Konzentration wie von selbst, ganz so wie die Atmung. Es kommt zur Ausblendung aller Kognition, die nicht unmittelbar auf die jetzige Ausführungsregulation gerichtet sind.
5. Das Zeiterleben ist stark beeinträchtigt; man vergisst die Zeit und weiß nicht, wie lange man schon dabei ist. Stunden vergehen wie Minuten.
6. Man erlebt sich selbst nicht mehr abgehoben von der Tätigkeit, man geht vielmehr gänzlich in der eigenen Aktivität auf (sog. „Verschmelzen“ von Selbst und Tätigkeit). Es kommt zum Verlust von (Selbst-)Reflexivität und Selbstbewusstsein.

Idealerweise beziehen Serious Games alle Sinne in den Lernprozess mit ein und schaffen für den Spieler eine angenehme Lernatmosphäre, in der er sich wohlfühlt. Es sollte dem Spieler möglich sein mit allen Objekten der Spielwelt zu interagieren, er sollte ausprobieren und mit verschiedenen Lösungsansätzen herumexperimentieren können. Des Weiteren sollte die Spielwelt auch über eine gewisse Fehlertoleranz bezüglich der Anwenderentscheidungen verfügen. Auf diese Weise werden Zusammenhänge leichter verstanden und das erworbene Wissen lässt sich anhand der Aufgabenstellungen des Spiels sofort praktisch anwenden und einüben.

2.1.2. Immersion

„…intense focus, loss of self, distorted time sense, effortless action…“ (Allen Varney: Immersion Unexplained. 2006)

Einen weiteren zentralen Aspekt des Lernens in virtuellen Umgebungen, der auch in Csíkszentmihályi's Modell des Flow-Erlebens beschrieben wird (vgl. Komponenten 5 und 6 des Flow-Erleben Modells) stellt der Begriff der Immersion dar: Taucht der Lernende völlig in die Realität des Spiels ein, bezeichnet man diesen Zustand als Immersion. Angesichts des packenden Spielerlebnisses tritt die Wahrnehmung der eigenen Person und des Lernprozesses in den Hintergrund. Der Anwender lernt, indem er sich ganz auf die Spielwelt einlässt. Der Grad der Immersion ist hierbei jedoch nicht allein von der Qualität der Anwendung abhängig, sondern kann auch über externe Faktoren, wie die Auswahl der Bedienelemente beeinflusst werden. Man unterscheidet hierbei zwischen:

- Vollkommener Immersion: totale Ausblendung der Realität (head-mounted-displays [HMD], Cave Automativ Virtual Environment [CAVE])
- Semi-Immersion: teilweise Ausblendung der Realität (Workbench, Virtual Reality)
- Perzeptuelle und inhaltliche Immersion (Lineare Medien, interaktive Medien)

2.1.3. Incentivierung

Obwohl auch für die Entwicklung klassischer Computerspiele relevant, gewinnt ist die Schaffung von Anreizen, die sogenannte Incentivierung, im Kontext der Serious Games durch die Bestrebung der Simulation, dem Spieler (komplexes) Wissen zu vermitteln und ihn zu kognitiven Leistungen anzuregen nochmals an Bedeutung:

Der Anwender wird belohnt, um Ehrgeiz und Einsatzbereitschaft weiter anzufachen – im Spiel etwa durch das Freischalten besonderer Extras, die den eigenen Aktionsradius erweitern oder den eigenen Erfolg nach außen hin für alle sichtbar machen. Incentives motivieren im Wettbewerb, indem sie den Spieler dazu animieren, besser als die anderen sein zu wollen. Er wird angestachelt, die gestellten Aufgaben möglichst gut zu lösen, um am Ende die Belohnung zu erhalten. Dabei kann es sich um begehrte Ausrüstung für den eigenen Avatar, nützliche Fertigkeitspunkte, Minispiele oder sonstige Boni handeln (3).

2.2. Spiele-Entwicklung

Ein von professionellen Spieleentwicklern hergestelltes Computerspiel ist heutzutage ein hochgradig komplexes, interaktives Mediensystem. Mit ca. 50-100 beteiligten Entwicklern (Producer, Projektleiter, Game Designer, Programmierer, Grafiker, Musiker, Leveldesigner, Tester etc.), 18-24 Monaten Entwicklungszeit und 10-20 Millionen € Entwicklungskosten stellen Vollpreistitel ein hohes unternehmerisches Risiko für das Entwicklerstudio selbst und eventuelle Publisher dar. Besondere Schwierigkeiten bei der Entwicklung von Computerspielen sind unter anderem:

- Hohe Anforderungen an dynamischer, immersiver Interaktion in Echtzeit
- Notwendigkeit, die neueste Technologie zu implementieren
- PC-Spiele müssen auf diversen Plattformen in verschiedensten Hardware-Kombinationen lauffähig sein
- Medienintegration, stark interdisziplinäre Arbeit
- Hohe Projektkomplexität
- Schlechte Standards im Software-Engineering Prozess

Die nachfolgende Abbildung soll zudem die Spiele-Entwicklungsphasen von der Entstehung der Spielidee bis zur abschließenden Bilanz verdeutlichen:

Abbildung 1: Spiele-Entwicklungsphasen

2.2.1. Die Entwicklung von Serious Games

Auch wenn das Budget ebenso wie der Inhalt der meisten Serious Games wesentlich weniger umfangreich ist als bei aktuellen unterhaltungsorientierten Vollpreistiteln sehen sich Entwickler von Serious Games diversen Herausforderungen gegenüber: Das pädagogische Ziel der Anwendung erfordert die Kooperation mit externen Experten des betreffenden Arbeits-/Wissensgebiets, sowie einen langen Entscheidungsprozess im Vorfeld der Entwicklung der Spielidee. Zentraler Bestandteil ist eine gute Kommunikation zwischen Entwicklern (setzen das Lernziel technologisch um, achten auf Gameplay und Spielspaß) und Lehrenden (liefern Ideen für neue Szenarien und Anwendungsgebiete). Anschließend muss eine kosteneffiziente Lösung gefunden und die Lernziele evaluiert werden. Zudem werden Serious Games aufgrund ihrer speziellen Ausrichtung fast ausschließlich über den Auftraggeber, nicht über den Markt finanziert und müssen ggf. auch für Nicht-Spieler zugänglich sein.

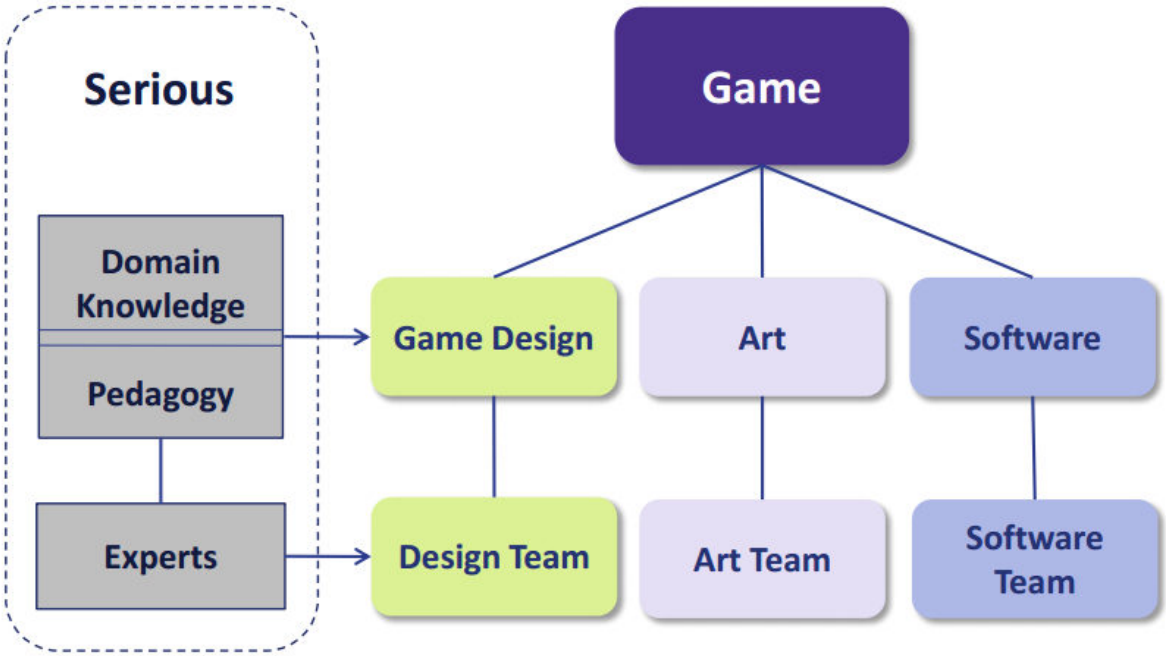


Abbildung 2: Spiele-Entwicklung von (Serious) Games

2.3. Anwendungsgebiete von Serious Games

Generell lassen sich Serious Games zunächst in 2 Bereiche einteilen: zum Einen Spiele, die beim Spieler zur Ausbildung einer bestimmten Handlungskompetenz beitragen wollen, zum Anderen Spiele, die dem Anwender theoretisches Wissen und Bildung vermitteln wollen.

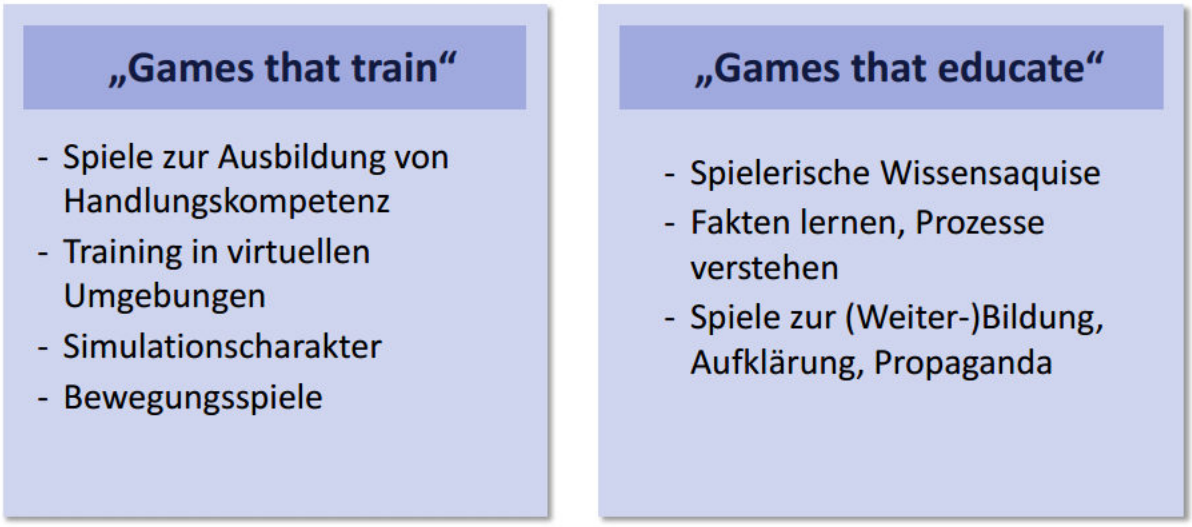


Abbildung 3: Serious Games

Die Anwendungsgebiete, für die Serious Games letztendlich entwickelt werden sind jedoch weitaus vielfältiger (4): Militärsimulationen wie z.b. „America's Army“, „Close Combat Marines“ oder „Full Spectrum Warrior“ sollen Soldaten in virtuellen Konfliktsituationen auf den Ernstfall vorbereiten bzw. neue Rekruten für die Armee werben. Politische Simulationen präsentieren einen überzeichneten Standpunkt oder bedienen sich Propaganda und Polemik um die politische Meinung des Rezipienten zu beeinflussen. Beispiele hierfür wären Serious Games wie „September 12th“, „Food Force“, oder auch das „Mc Donald's Video Game“ der gleichnamigen Fast-Food-Kette. Dem gegenüber stehen Fitness-Spiele, die eine körperliche Betätigung erfordern, die mittels Sensoren erfasst wird. Beispiele hierfür wären „Wii Fit“, „Dance Dance Revolution“ oder „Microsoft Kinect“. Einen Großteil des Serious Gaming Marktes machen zudem Spiele aus, die sich mit Bildung und Weiterbildung beschäftigen. Spiele wie „EnterTech“, „Brain Age“ oder „Virtual U“ sollen explizit Wissen oder mentale Fertigkeiten lehren, oder mittels spielerischer Simulation zum Verständnis komplexer Prozesse beitragen. Digitale Lernspiele können aufgrund der angesprochenen Zielgruppen zudem in weitere Untergruppen unterteilt werden, da sich Lernspiele für Kinder aufgrund ihres Anspruchs, ihrer Thematik und Interaktionsmöglichkeiten teilweise gravierend von Lernspielen zur Erwachsenenbildung unterscheiden. Da sich die vorliegende Semesterarbeit vorrangig mit dem letztgenannten Themengebiet beschäftigt sollen im Folgenden die Einsatzgebiete interaktiver Lernanwendungen zur Erwachsenenbildung fokussiert werden.

Abbildung 4: Anwendungsgebiete Serious Games

2.4. Projektmotivation

Gerade im Umgang mit Gefahrenstoffen, hochwertigen Materialien und/oder komplexen Maschinen ist es für Firmen von äußerster Wichtigkeit, über qualifiziertes Personal zu verfügen, dass mit den Verfahrensabläufen vertraut ist und auch im Problemfall schnell und sicher handeln kann. Aus diesem Grund wurde im Zuge der Semesterarbeit eine interaktive Lernanwendung konzipiert und programmiert, in welcher der Rezipient die Bedienung einer fiktiven Verpackungsmaschine erlernen soll. Der „Paketomat 3000“ soll den Spieler (in diesem Fall einen potentiellen neuen Fabrikmitarbeiter) mit der Bedienung der Maschine vertraut machen. Natürlich ist es möglich den Umgang mit einer solchen Maschine auch mit einem klassischen Handbuch zu erlernen, der große Vorteil eines Serious Games ist allerdings, dass auch komplexe Vorgänge oder Notfallszenarien trainiert werden können, ohne Material- oder Personenschäden befürchten zu müssen. Die spielerische Umgebung wirkt zudem motivierend und senkt die Einstiegshürden (5). Der ungeschulte Arbeiter kann innerhalb der Simulation frei mit allen Objekten interagieren (vgl Kapitel 2.1.1), ohne befürchten zu müssen, durch einen Druck auf den falschen Knopf an der (realen) Maschine den Produktionsablauf zu gefährden. Dabei gibt das Spiel auf unterschiedliche Weise kontinuierliches Feedback. Dies können Informationen aus der Spielwelt (z.b. die Veränderung des Maschinenzustands) oder ein Bericht über die Leistung des Spielers sein (z.b. „Mission erfolgreich!“) sein. Das Feedback ermöglicht es dem Spieler, die beste Strategie herauszufinden und seinen Punktestand zu verbessern. (6)

Des Weiteren belegen wissenschaftliche Studien, dass Menschen Informationen besser behalten, wenn sie möglichst viele Sinne nutzen. Sie sollten lesen, hören, sehen und

gleichzeitig emotional und interaktiv an dem Lernexperiment teilnehmen. Informationen werden im Vergleich zu einem traditionellen Präsenztraining im Schnitt 25-50% länger behalten, der Kontext um 40-70% schneller verstanden (7). Hinzu kommt, dass die Lernsimulation zu jeder Zeit und an einem beliebigen Ort, der über die entsprechenden technischen Voraussetzungen verfügt eingesetzt werden kann, ohne dass die Maschine selbst verfügbar sein muss. So ist es möglich, auch große Gruppen von Trainees in intensiven und individuellen Lerneinheiten zu schulen.

Abbildung 5: Schulungssituation Paketomat 3000

Ziel der interaktiven Lernanwendung soll es letztendlich sein umfassende Handlungskompetenz in den Bereichen der Kommunikation, Beobachtung und Entscheidung zu vermitteln. Im konkreten Fall des Paketomat 3000 bedeutet dies: der Anwender soll nach mehreren Spielen in der Lage sein die Maschine sicher zu bedienen, er soll aber auch in der Lage sein eventuelle Probleme im Produktionsablauf frühzeitig zu erkennen und zu beheben. Des Weiteren soll er in der Lage sein schnell und sicher auf etwaige Notfallsituationen zu reagieren.

2.5. Geplantes Vorgehen

Zu Beginn des Projekts machten wir uns Gedanken darüber, in welchen Schritten wir das Projekt angehen wollen und kamen zu folgendem Vorgehen:

1. Ausarbeiten einer Idee und Lernzielen/ Aufgaben
2. Auswahl einer geeigneten Entwicklungs- und Toolumgebung
3. Einarbeiten in die Toolumgebung, insbesondere die Entwicklungsumgebung
4. 3D-Umgebung erstellen
5. Programmierung der Aufgaben und schaffen von Interaktivität mittels Scripts
6. Testen und Optimieren
7. Projektdokumentation verfassen

3. Praktische Umsetzung

3.1. Auswahl einer Entwicklungsumgebung

Wie alle Computerprogramme benötigen auch Spiele eine Entwicklungsumgebung. Auch wenn einzelne Bestandteile des Spiels in unterschiedlichen Programmen erstellt werden, so kommt letztlich doch alles in einem Programm zusammen. Meist bietet diese sogenannte Game-Engine neben der eigentlichen Entwicklungsumgebung auch eine Reihe nützlicher Funktionen, die die Erstellung des Spiels deutlich erleichtern.

Von den Möglichkeiten der Engine hängen weitere Faktoren bei der Erstellung eines Spiels ab. Je nach Programmiersprache, Schnittstellen etc. können bzw. müssen andere Programme zur Bereitstellung von Inhalten verwendet werden.

Im Folgenden wir beschrieben, welche Tools wir für die Erstellung des Spiels einsetzen und warum.

3.1.1. Game Engine

Als zentraler Bestandteil und Ausgangsbasis für alle Aktivitäten war die Game-Engine das wichtigste Tool. Wie bereits beschrieben hängen von der Engine viele weitere Faktoren ab. Es gibt eine Vielzahl von Engines auf dem Markt und es galt für uns, die passendste auszuwählen.

Zunächst fokussierten wir uns auf die gängigen und bekannten Engines wie die Cryengine der Firma Crytek und die Unreal Engine von Epic Games, stießen bei unserer Recherche jedoch schnell auf einen hilfreichen Wikipedia-Artikel. Dieser war eine entscheidende Hilfe bei der Auswahl einer Engine, da dieser einen schnellen Überblick über die verfügbaren Engines und deren grobe technische Daten gab. Der Artikel ist unter http://de.wikipedia.org/wiki/Liste_von_Spiel-Engines zu erreichen.

Ein wichtiges Auswahlkriterium für uns war die kostenlose bzw. kostengünstige Verfügbarkeit der Engine, wobei der Anteil der kostenlosen Engines erstaunlich hoch ist. Wir waren durchaus überrascht, dass selbst bekannte, qualitativ hochwertige Engines wie die Cryengine und die Unreal Engine zumindest für den nicht kommerziellen Gebrauch kostenlos verfügbar sind.

Letztendlich entschieden wir uns jedoch für DX Studio als unsere Game-Engine. Dies hatte vor allem folgende Gründe:

- Kostenlos auch bei kommerzieller Nutzung
- Scriptsprache JavaScript
- Funktionsumfang

- Explizite Benennung als Engine für Serious Games auf der Homepage[1]
- Importmöglichkeiten von 3D Modellen aus gängigen Programmen wie Blender oder 3Ds Max
- 1 Jahr lang kostenloser Support

Insgesamt schien DX Studio also genau das zu sein, wonach wir suchten. Insbesondere der Funktionsumfang mit Grafik-, Audio- und Physikkomponente boten alles, was wir benötigten. JavaScript als Programmiersprache war ein weiterer Vorteil, da uns diese bereits ansatzweise bekannt war und wir uns so nicht in eine zusätzliche, neue Sprache einarbeiten mussten.

Die Möglichkeit, 3D-Modelle aus gängigen Tools zu importieren empfanden wir ebenfalls als wichtig, da so die Möglichkeit bestand, in einer vertrauten Umgebung 3D-Modelle zu erstellen und dabei den vollen Funktionsumfang der 3d-Tools zu nutzen.

Auch die sonstigen Funktionen und Möglichkeiten, die unter http://www.dxstudio.com/features.aspx aufgezählt wurden, überzeugten uns von DX Studio, auch im Hinblick auf mögliche, zukünftige Weiterentwicklungen.

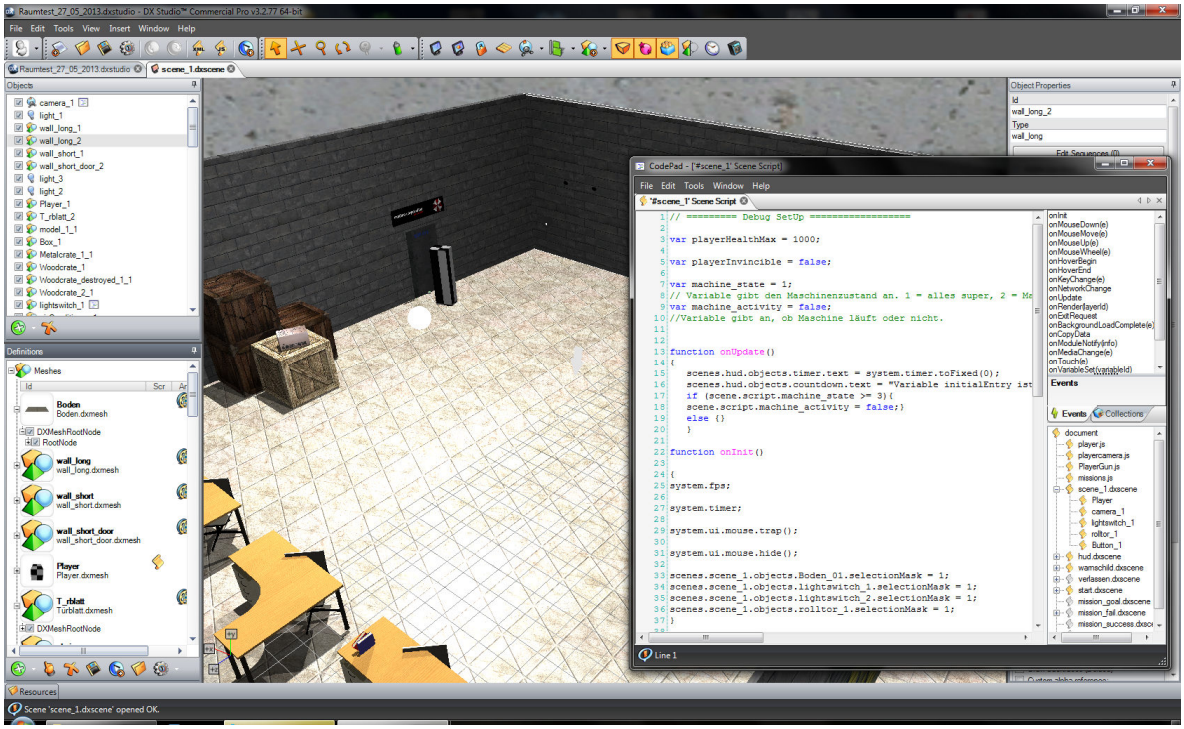


Abbildung 6: DX Studio Entwicklungsumgebung mit Objektbibliothek, Werkzeugleiste und Codepad, durch das Codepad verdeckt sind die Objekteigenschaften, zentral das Vorschaufenster

[1] http://www.dxstudio.com/index.aspx im Seitentitel

3.1.2. 3D Modeling

Wie bereits beschrieben, ist es möglich, in DX Studio 3D Daten aus allen gängigen Tools zu importieren, solange diese 3D-Modeel in bestimmten Standardformaten vorliegen. Zwar hat DX Studio auch einen eigenen 3D Editor, dieser ist jedoch lediglich in der Lage einfache Grundformen zu erstellen, wenngleich die Möglichkeiten zum Zuweisen von Materialien, Farben und Texturen erstaunlich ausgereift ist.

Letztendlich verwendeten wir für die Modellierung hauptsächlich 3Ds Max von Autodesk. Dieses Tool wird auch von vielen Professionellen Spieleentwicklern eingesetzt, um3D-Modelle zu erstellen. Für Studenten steht 3Ds Max im Rahmen des Studentenprogramms von Autodesk kostenlos zur Verfügung.

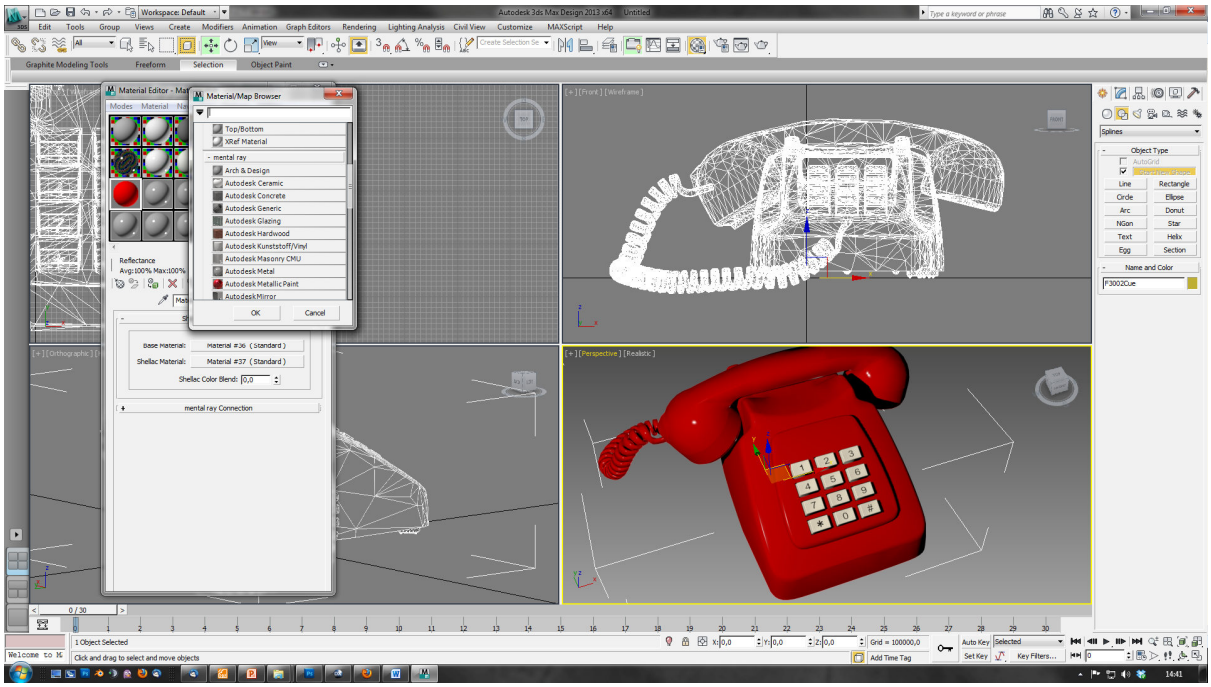


Abbildung 7: Modellerstellung in 3ds Max

Die grundsätzlich kostenlose Alternative stellt Blender dar. Jedoch mussten wir im Laufe des Projekts feststellen, dass Daten aus Blender zwar in DX Studio importiert werden konnten, deren Darstellung jedoch derart verfälscht war, dass eine Verwendung von 3d-Daten aus Blender nicht möglich war. Daher entschieden wir uns, sämtliche 3D-Modelle in 3ds Max zu erstellen.

Zusätzlich kamen einige, wenige Modelle aus der Bibliothek von DX Studio zum Einsatz. In dieser Bibliothek legen DX-Studio-Nutzer von ihnen angefertigte Modelle usw. ab, so dass diese von anderen Nutzern verwendet werden können. Da die Anzahl der Nutzer begrenzt ist, beschränkt sich das Angebot an Modellen jedoch momentan auf wenige Themengebiete.

3.1.3. Weitere Tools

Neben den genannten Tools kamen auch weitere Programme zum Einsatz. Hierzu zählen insbesondere Bildbearbeitungsprogramme für sämtliche 2D-Elemente des Spiels, einschließlich von Texturen für 3D-Modelle sowie Konvertierungsprogramme für Mediendateien. Für die Bildbearbeitung verwendeten wir hauptsächlich Adobe Photoshop,

das quasi das Standardtool zur professionellen Bildbearbeitung darstellt und in der Lage ist, in unterschiedlichste Zielformate zu exportieren. Insbesondere die Möglichkeit, über PlugIns .dds Dateien zu erzeugen und zu öffnen macht es auch in der Spielentwicklung zu einem nützlichen Tool. Bei .dds Dateien handelt es sich um spezielle Dateien, die die Texturen von Objekten darstellen können. Diese Dateien sind für die Darstellung mit dem von Microsoft entwickelten Standard Direct X vorgesehen. Dieser Microsoft Standard kommt bei den meisten Spielen zum Einsatz.

Zusätzlich zur Bildbearbeitung und Medienkonvertierung waren auch Video- und Audiobearbeitungsprogramme vorgesehen, die jedoch wenig bis gar nicht zum Einsatz kamen, da eine Nachbearbeitung insbesondere von Audiodateien nicht nötig war.

3.1.4. Übersicht der verwendeten Tools

Im Folgenden nochmals ein kurzer Überblick über die Verwendeten Programme im Einzelnen:

- DX Studio, Game Engine, Entwicklunsumgebung
- 3ds Max, 3D Modellierung
- Blender, 3D Modellierung, Daten nicht verwendbar
- Adobe Photoshop, 2D-Elemente, GUI, Texturen
- XMedia Recode, Konvertierung von Mediendateien in verwendbare Formate
- Windows Live Movie Maker, Erstellen von Platzhaltervideos
- Audacity, Audio Bearbeitung, wurde aber quasi nicht benötigt

3.2. Grobkonzept Storyboard

Bevor wir mit der eigentlichen Umsetzung begannen, überlegten wir uns, wie das Spiel grob ablaufen sollte. Hierbei kam schnell der Gedanke auf, Schulung und praktische Umsetzung in der virtuellen Umgebung zu kombinieren. Die Idee dahinter war, den Aufwand für Schulungen auf ein Minimum zu reduzieren und möglichst alle wichtigen Bestandteile der Schulung in einem Paket zu haben.

Daraus ergibt sich auch die logische Reihenfolge der Inhalte: Zuerst sollten die Lernenden eine virtuelle Schulung durchlaufen und anschließend das erworbene Wissen in der Praxis erproben.

Insgesamt gliedert sich das Spiel in 4 Phasen:

1. Spielstart
2. Schulung
3. Praktische Durchführung
4. Spielende

Zusätzlich gibt es noch einen Hilfebildschirm, der grundlegende Informationen zum Spiel gibt. Dieser ist über das Menü in allen Spielsituationen erreichbar.

3.2.1. Spielstart

Diese Phase Spielstart fällt relativ kurz aus und hat in erster Linie den Zweck, den Anwender im Spiel willkommen zu heißen und ihm die Möglichkeit zu geben, das eigentliche Spiel zu einem Zeitpunkt seiner Wahl zu starten. Beim Start des Spiels wird dem Anwender ein Auswahlmenü präsentiert, in dem er die Möglichkeit hat, die Darstellungsgröße des Fensters zu wählen, die Anwendung zu schließen und sich eine kurze Information zu Ziel und Bedienung des Spiels anzusehen. Wichtigstes Bedienelement ist jedoch der Button zum Starten des Spiels.

3.2.2. Schulung

Nachdem der Anwender sich dazu entschieden hat das Spiel zu starten, verlässt er das Spielstart-Menü und findet sich in der virtuellen 3D Welt wieder. Dort wird der Anwender in folgende Situation gestellt: Es sind Tische und Stühle aufgestellt, an der Stirnseite des Raums befindet sich eine Leinwand, auf der zu erkennen ist, dass er sich in einer Schulung befindet.

Grundsätzlich kann sich der Anwender frei im Raum bewegen, hat aber zunächst keinen Zugang zur eigentlichen Maschine. Dieser wird erst nach erfolgreicher Schulung oder auf ausdrücklichen Wunsch des Benutzers gewährt. Die beiden Bereiche sind durch ein Rolltor abgetrennt, dieses ist zu Beginn des Spiels noch verschlossen.

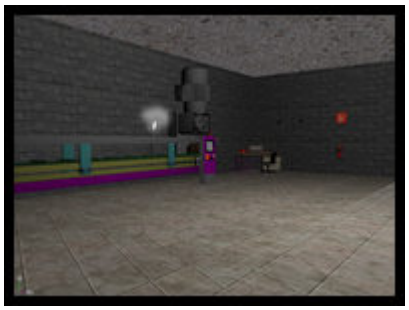
Die eigentliche Schulung ist ein Video, das ausgelöst wird, wenn der Anwender mit einem Stuhl interagiert. Hiermit soll der Vorgang des Hinsetzens angedeutet werden.

Nach Abschluss des Schulungsvideos wird das Rolltor freigegeben und der Anwender kann dieses passieren, um zur Maschinenhalle zu gelangen.

Weitere Interaktionsmöglichkeiten mit Objekten in der Umwelt sind rein fakultativ und haben keine weitere Bedeutung für den eigentlichen Spielverlauf. Sie tragen lediglich zu einem realistischen Umfeld bei.

Will der Anwender die Schulung überspringen, kann er dies tun, indem er direkt mit dem Rolltor interagiert und bestätigt, dass er die Schulung überspringen will.

3.2.3. Praktische Durchführung



Nachdem der Anwender das Rolltor passiert hat, erhält er direkt seine erste Aufgabe. Insgesamt gibt es derzeit drei aufgaben mit ansteigendem Schwierigkeitsgrad. Je nach Aufgabe sind unterschiedliche Interaktionen innerhalb der Maschinenhalle nötig, um die Aufgabe erfolgreich abzuschließen.

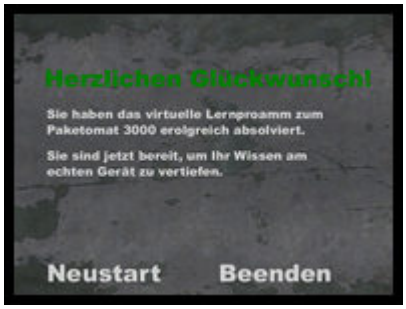
Folgende Aufgaben sind aktuell vorhanden:

1. Maschine einschalten: Der Anwender soll den Einschaltknopf der Maschine betätigen, damit diese den Betrieb aufnimmt

2. Maschine überwachen: Der Anwender soll die Maschine im Auge behalten und bei auffälligem Verhalten die Maschine ausschalten und den Schichtleiter anrufen. Für das Ausschalten ist eine erneute Interaktion mit dem Einschaltknopf der Maschine notwendig, für den Anruf genügt es in diesem Fall, mit dem Telefon zu interagieren.

3. Maschine brennt: Durch einen technischen Defekt fängt die Maschine Feuer. Der Anwender muss innerhalb einer festgelegten zeit die Maschine löschen und die Feuerwehr verständigen. Gelingt ihm dies nicht innerhalb der vorgegebenen Zeit, explodiert die Maschine und die Aufgabe wurde nicht erfüllt. Ein erneutes Durchlaufen des Trainingsprozesses ist dann erforderlich. Wie in der vorherigen Aufgabe reicht eine einfach Interaktion mit dem Telefon, um die Feuerwehr zu verständigen. Das Feuer wird gelöscht, indem der Anwender mit dem Feuerlöscher an der Wand interagiert wodurch der Feuerlöscher vom Anwender aufgenommen wird. Mit dem Feuerlöscher muss sich der Anwender nun zur Maschine begeben und mit dieser interagieren.

3.2.4. Spielende



Hat der Anwender alle Aufgaben erfolgreich bewältigt, verlässt er die 3D Welt wieder und eine Nachricht wird angezeigt, die den Anwender über den erfolgreichen Abschluss des Lernprogramms informiert. Damit ist das Programm theoretisch abgeschlossen. Der Benutzer hat jedoch neben der Möglichkeit, das Spiel zu verlassen auch die Möglichkeit, das Spiel erneut zu starten, z.B. um seine Leistung zu verbessern oder sein Wissen zu festigen.

3.2.5. Hilfemenü

Das Hilfemenü soll dem Anwender eine Hilfestellung geben, wenn dieser mit dem Spiel nicht zurechtkommt. Das Menü enthält Informationen zur Steuerung des Spiels und zum Gesamtziel des Spiels.

Das Hilfemenü ist von unterschiedlichen Stellen des Spiels aus erreichbar. In jedem relevanten Menü gibt es einen Button, der das Hilfemenü einblendet. Während des Spiels kann das Menü über die Taste Escape aufgerufen werden. Als zusätzlicher Schnellzugriff ist

die Taste „H“ definiert, die das Hilfemenü direkt einblendet. Dies ist im Hilfemenü nicht erwähnt und stellt daher eher eine Zugriffsmethode für Experten dar.

3.3. 3D-Umgebung

Die dreidimensionale Spielumgebung muss zunächst geschaffen werden, da es seitens der der Engine in dieser Hinsicht grundsätzlich keine Vorgaben oder Beschränkungen vorliegen. Das bedeutet aber auch, dass wir als Entwickler eine vollständige 3D Umgebung schaffen mussten, die unserem Spiel den Rahmen gab. Wir entschieden uns für eine Fabrikhalle mit angrenzendem Schulungsraum, ohne nutzbare Möglichkeiten, dieses Umfeld verlassen zu können. Diese relativ kleine Welt erleichterte es uns, diese auszugestalten und alle relevanten Spielfunktionen auch räumlich dicht beieinander zu behalten.

Generell gingen wir bei der Ausgestaltung der Umgebung ähnlich wie bei einem Hausbau vor: Zunächst stellten wir die Grundmauern auf und fügten anschließend Stück für Stück Einrichtungsgegenstände, Lichtquellen und andere Objekte hinzu.

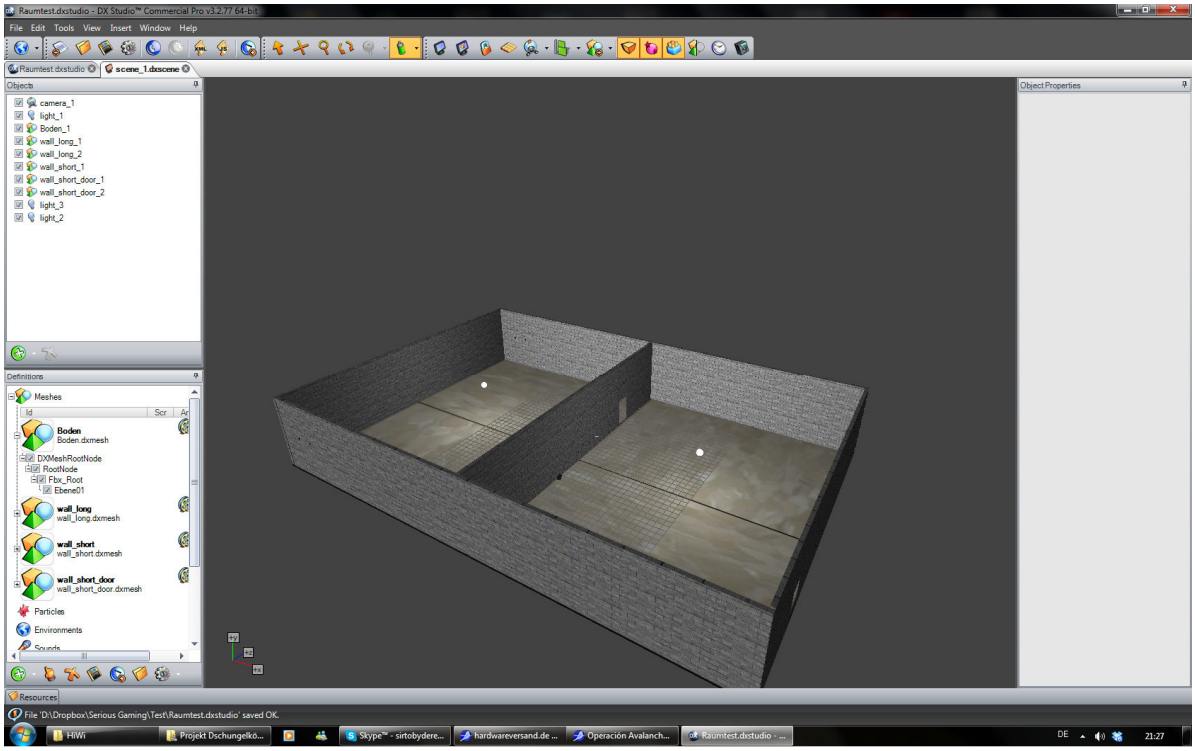


Abbildung 8: 3D-Umgeung relativ zu Beginn der Entwicklung, die Grundmauern stehen

Parallel zur Erstellung von neuen 3D-Objekten und deren Positionierung in der 3D-Umgebung begannen wir, einigen Objekten erste Funktionen zuzuweisen. Dies geschah über Scripte, die den jeweiligen Objekten zugeordnet waren. Im folgenden Kapitel wird dieses Vorgehen noch näher erläutert.

Beim Einfügen von 3D-Objekten fiel uns jedoch eine Eigenheit von DX Studio auf: 3D-Modelle wurden in der Voransicht häufig fehlerhaft dargestellt. Insbesondere dann, wenn diese aus mehreren Baugruppen bestanden, wurden die Baugruppen häufig verschoben eingefügt. Sobald man das Spiel jedoch startetet, waren alle Baugruppen wieder an ihrem

vorgesehenen Platz. Das machte zwar das Positionieren von Objekten etwas mühsam, jedoch nicht gänzlich unmöglich.

Bei 3D-Modellen, die im Programm Blender erstellt wurden, war dieser Effekt jedoch so sehr ausgeprägt, dass hier eine Positionierung nahezu unmöglich wurde. Daher entschlossen wir uns, ausschließlich 3Ds Max zu verwenden, da der Effekt hier deutlich weniger stark auftrat.

Ein weiteres Problem war die Skalierung von Objekten. Anscheinend verwendet DX Studio andere Maßstäbe als 3ds Max oder es war nötig Objekte in 3Ds Max in einem andern Maßstab zu erstellen, um detailgetreu und mit handhabbaren Objektgrößen gerecht zu werden. Leider bietet DX Studio keine Möglichkeit, Objekte ohne weiteres über numerische Werte zu skalieren. So war es stets eine frage des Augenmaßes, ein Objekt in der Welt zu platzieren und auf die richtige Größe zu bringen.

Trotz aller Schwierigkeiten, hatten wir schlussendlich eine 3d-Umgebung geschaffen, die unseren Erwartungen und Anforderungen gerecht wurde.

Abbildung 9: Draufsicht auf die 3D-Umgebung im finalen Zustand, die Decke des Raums ist ausgeblendet, links der Maschinenraum, rechts der Schulungsraum

Das eigentliche Erstellen der 3D-Modelle war zwar für das Projekt nötig, jedoch nicht Fokus dieser Arbeit, daher wird auf eine Beschreibung der Erstellung von 3D-Modellen an dieser Stelle verzichtet.

3.4. Programmierung

Die Grundlage für jegliche Interaktivität im Spiel bilden Scripte, die unterschiedlichen Objekten Funktionen zuweisen. DX Studio unterscheidet zwischen vier unterschiedlichen Arten von Scripten: Dokument- Szenen-, Objekt- und externe Scripte. Alle Scripttypen basieren auf JavaScript und folgen den entsprechenden Syntaxregeln. Der einzige Unterschied besteht darin, auf welche Elemente des Spiels sie sich beziehen. Aus den Namen wird ersichtlich, dass sich Dokumentscripte auf Dokumentebene befinden, Szenenscripte auf Szenenebene und Objektscripte auf Objektebene. Externe Scripte befinden sich außerhalb der Ebenen und werden auf der gleichen Hierarchiestufe wie Dokumentscripte verwaltet, können aber innerhalb jeder anderen Scriptart eingebunden werden.

Diese Trennung der Scripte führte zuweilen zu leichten Verwirrungen. Dies war dadurch bedingt, dass bestimmte Funktionen nur auf bestimmten Ebenen angesprochen werden konnten, insbesondere Timer, die jedoch für zeitlich gesteuerte Aktionen unerlässlich waren.

Andererseits jedoch bot DX Studio einige gute Hilfestellungen beim Programmieren. So wurden Scripte vor dem Speichern vor dem Speichern automatisch auf Validität geprüft. Diese Funktion konnte auch manuell gestartet werden, so dass Syntaxfehler und vergessene Steuerzeichen kein großes Problem darstellten. Auch wenn diese Prüfung nicht in der Lage war, zu prüfen, ob Objektpfade und Variablen korrekt waren.

Dafür gab DX Studio bei der Adressierung von Objekten und der Auswahl von Befehlen und Funktionen Unterstützung und bot in Dropdown-Menüs alle vorhandenen Auswahlmöglichkeiten an, was eine erhebliche Hilfe darstellte und Schreibfehler in Befehlen und Pfaden nahezu vollständig verhindern konnte.

3.4.1. Einstieg in die Programmierung

Auch wenn DX Studio eine auf JavaScript basierende Scriptsprache zur Programmierung verwendet, haben diese Sprache und ihre Verwendung innerhalb von DX Studio doch einige Eigenheiten. Um einen Einstieg in die Programmierung mit DX Studio zu erleichtern, gibt es unter http://www.dxstudio.com/guide.aspx eine Vielzahl an Tutorials zu den unterschiedlichen Teilbereichen von DX Studio.

Insbesondere das „First Person Shooter Frame Work Tutorial“[2] des Nutzers aRa NightHawk erwies sich für uns als besonders hilfreich. Weniger wegen der Shooter-Komponenten, als wegen der Grundlagen, die hierin für die Kameraeinstellungen und Spielersteuerung gelegt werden.

Aus diesem Grund bilden die Ergebnisse aus diesem Tutorial auch die Basis für unser Spiel, wobei die Grundlagen an unsere Bedürfnisse angepasst und erweitert wurden, während nicht notwendige Funktionen von uns entfernt wurden.

[2] http://www.dxstudio.com/guide.aspx?id=f1662876-a199-454f-9ff9-2b5cd77dc3b9

3.4.2. Codebeispiel

Anhand des folgenden Beispiels soll gezeigt werden, wie Scripte in DX Studio grundsätzlich aufgebaut sind. In diesem Fall handelt es sich um das Objektscript des Rolltors.

```
function onPhysicsContact(Player)
{
	if (system.script.training == true) {
	objects.rolltor_1.sequencePlay ("move_up") ; // activate the
sequence.
	sounds.TankTurret.play();
	object.setTimer("Tor", 5);
	}
	else {
		layers.Message_1.visible = true;
		system.ui.mouse.show();
		system.ui.mouse.untrap();
	}
}

function onTimer(Tor)
{
	objects.rolltor_1.sequencePlay("move_down"); // activate the
sequence.
	sounds.TankTurret.play();
	}
```

Auf Dokumentebene wurde bereits eine Variable definiert, die enthält, ob der Anwender das Training absolviert bzw. übersprungen hat oder noch nicht. Der Standardwert ist, dass er dies noch nicht getan hat.

Wenn jetzt der Anwender das Tor berührt wird die Funktion mit dem Event-Listener „onPhysicsContact“ ausgelöst. In dieser Funktion wird abgefragt, ob die Variable „Training“ wahr oder falsch ist. Da die Variable auf Systemebene befindet, muss diese erst noch adressiert werden, bevor die Engine ihren wert korrekt ermitteln kann.

Wenn die Variable den Wert „wahr“ hat, dann wird das Objekt „Rolltor“ adressiert und eine Sequenz ausgelöst, die das Tor nach oben fahren lässt. Zeitlich wird eine Audiodatei abgespielt, diese soll das Geräusch des hochfahrenden Tors darstellen. Außerdem wird ein Objekt-Timer mit dem Namen „Tor“ und dem Zeitwert 5 erstellt. Das bedeutet, dass nach 5 Sekunden die zugeordnete Funktion mit dem Event-Listener „onTimer“ ausgelöst wird.

Ist die Variable „Training“ nicht wahr, wird die Ebene „Message_1“ sichtbar geschaltet und der Mauszeiger sichtbar. Außerdem kann der Mauszeiger dann auch außerhalb des Anwendungsfensters verwendet werden. Die nun angezeigte Ebene enthält eine Warnung, dass das Training noch nicht durchlaufen wurde und bietet über eigene Scripte die Möglichkeit, die Variable „Training“ auf wahr zu setzen und das Tor zu öffnen.

Der letzte Teil des Codes enthält eine Funktion die durch den Timer „Tor“ ausgelöst wird. Wenn der Timer abgelaufen ist, wird eine weitere Sequenz abgespielt, die das Tor wieder nach unten fahren lässt. Dabei wird wiederum eine Audiodatei abgespielt.

In diesem Stil sind alle Scripte aufgebaut. Funktionen werden über Event-Listener aufgerufen und je nachdem welche Aktion ausgeführt wird und welche Variablen welche Werte haben, werden unterschiedliche Befehle ausgeführt.

3.4.3. Probleme beim Programmieren

Trotz aller Hilfsmittel beim Programmieren ergaben sich im Projektverlauf doch auch einige Probleme.

Viele Aktionen im Spiel werden durch Timer ausgelöst, insbesondere im zweiten Teil, in dem der Anwender die Aufgaben an der Maschine erfüllen muss. Das Problem hierbei war nun, dass einem Objekt immer nur ein Timer zugewiesen werden konnte. Außerdem konnten Timerfunktionen nur innerhalb des Objekts aufgerufen werden, für das der Timer definiert war und konnten nicht von außen aufgerufen werden. Eine Definition von Timern von Timern von außen hingegen war problemlos möglich. Leider waren diese Einschränkungen in der Dokumentation der Engine nirgendwo erwähnt, so dass es viele Experimente brauchte, bis alles einigermaßen lief, wie es sollte.

Ein weiteres Problem ergab sich beim Einbinden des Schulungsvideos. Prinzipiell verfügt DX Studio über die Möglichkeit, Videodateien einzubinden und wiederzugeben. Es gelang uns jedoch nicht, das Schulungsvideo erfolgreich einzubinden. Auch das Aufrufen einer externen Datei gelang uns nicht. Auch die DX Studio Community konnte uns in diesem Fall nicht weiterhelfen, da die Methode, die wir zum Aufrufen verwendeten genau dem entsprach, was uns auch die Community empfohlen hatte.

So mussten wir letztendlich den Videoaufruf im Spiel auslassen. In einem realen Projekt hätten wir eine alternative Lösung in Betracht gezogen, die wir jedoch aus Zeitgründen in diesem Projekt nicht mehr realisieren konnten. Die Alternative wäre gewesen, die Kameraeinstellungen per Script so zu steuern, dass diese nacheinander bestimmte wichtige Punkte innerhalb der Spielwelt ansteuern. Zeitgleich würde eine Tonspur im Hintergrund abgespielt werden, die die gezeigten Elemente und die Bedienung erklärt.

3.5. 2D-Elemente

Auch wenn sich der Anwender die meiste Zeit innerhalb eines dreidimensionalen Raums mit zahllosen 3D-Objekten bewegt, so sind auch die zweidimensionalen Elemente des Spiels von entscheidender Bedeutung. Im Folgenden sollen die wichtigsten 2D-Elemente kurz vorgestellt werden.

3.5.1. HUD

Die Abkürzung HUD steht für Head-Up-Display und bezeichnet im Spielebereich den Bereich des Spielbildschirms, in dem dem Spieler wichtige Informationen angezeigt werden, z.B. Himmelsrichtungen, Geschwindigkeit, verbleibende Zeit o.ä.

In unserem Fall waren etwas andere Informationen von Nöten. Unser HUD zeigt dem Anwender an, ob die Maschine läuft, in welchem Zustand sich diese befindet, wie viel zeit insgesamt abgelaufen ist und wie viel Zeit dem Anwender bis zum Eintreten bestimmter

Events bleiben. Außerdem wird in die Bildschirmmitte und ein eventuell verfügbarer Ausrüstungsgegenstand im Zentrum des Bildschirms durch ein Symbol gekennzeichnet.

Abbildung 10: HUD-Elemente, oben Zeitanzeigen, unten links Maschinenanzeigen, Mittenindikator

Die Maschinenanzeigen im unteren linken Bereich verändern Farbe und Text je nachdem, ob die Maschine läuft und in welchem Zustand sich diese befinden.

Abbildung 11: Mittenindikatoren mit unterschiedlichen Gegenständen

Der Mittenindikator hat eine Doppelfunktion. Einerseits zeigt er die Bildschirmmitte an und erleichtert so die Interaktion mit Objekten, andererseits ändert er sein Erscheinungsbild, wenn bestimmt Gegenstände ausgerüstet werden. In unserem Fall ändert sich die leere Hand zum Feuerlöscher, wenn der Anwender mit dem 3D-Objekt Feuerlöscher interagiert, um ein Feuer zu löschen. Der Mittenindikator zeigt so also auch an, mit welchem Gegenstand der Anwender also aktuell ausgerüstet ist. Entsprechend andere Interaktionsmöglichkeiten gibt es in diesem Fall.

Die Zeitanzeigen sollen dem Anwender in erster Linie eine Möglichkeit zur Selbstkontrolle und einen Überblick über die bisher benötigte Zeit geben. Der Countdown ist bisher noch ohne Funktion, kann aber verwendet werden, um auf nahende Ereignisse hinzuweisen und Zeitgrenzen für bestimmte Tätigkeiten vorgeben, z.B. wenn die Maschine innerhalb von X

Sekunden nach Eintreten eines bestimmten Falls ausgeschaltet sein muss, um Schäden zu vermeiden.

3.5.2. Menüs

Für die Bedienung des Spiels außerhalb des eigentlichen Spiels sind Menüs von entscheidender Bedeutung. Sie erlauben es, Einstellungen an der Fenstergröße vorzunehmen, das Spiel zu beenden oder neuzustarten etc. Außerdem informieren Informationsfenster über aktuelle Aufgaben und helfen dem Anwender so, seinen Weg durch das Spiel zu finden. Die Menüs in unserem Spiel setzen sich aus Grafiken und Textelementen zusammen. Die Grafiken wurden in externen Programmen, v.a. Photoshop erstellt bzw. bearbeitet und sind in erster Linie Hintergrundgrafiken. Die Texte und Bedienelemente wiederum wurden in der 2D Szene von DX Studio erstellt. Dies erlaubt es, die Texte scriptgesteuert zu beeinflussen, so dass im selben Menü je nach Situation unterschiedliche Texte angezeigt werden können und die Texte unterschiedlich gestaltet werden können, ohne dafür neue Szenen erstellen zu müssen.

3.5.3. Texturen

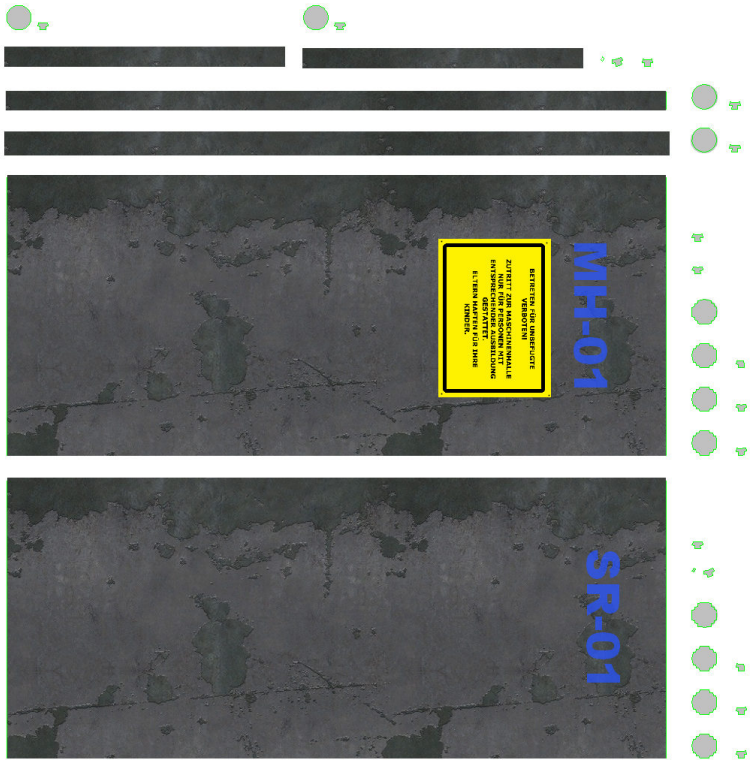


Abbildung 12: Texturvorlage für eine Tür

Texturen sind Grafiken, die auf 3D-Objekte gelegt werden, um diesen ein realistischeres Aussehen zu geben. Damit tragen sie entscheidend zur Atmosphäre des Spiels bei und sorgen für eine hohe Realitätsnähe. Je nach Objekt sind unterschiedlich komplexe Texturen nötig. Die Kisten im Spiel z.B. sind in ihrer Grundform Würfel, denen auf jeder Seite die gleiche quadratische Textur zugewiesen wurde. Anders sieht es bei der Textur für eine Tür aus, wie sie hier zu sehen ist. Die Tür hat zwar grundsätzlich auch eine relativ einfache Grundform, verfügt aber zusätzlich noch über einen Türknauf und soll außerdem von unterschiedlichen Seiten unterschiedlich aussehen. Daher wurden aus 3Ds Max Texturvorlagen exportiert, die dann mit Photoshop bearbeitet werden können. Bei den Vorlagen werden die einzelnen Seiten des dreidimensionalen Objekts in eine zweidimensionale Ebene gebracht und können dann wie

eine normale Grafik bearbeitet werden. Später erkennt 3Ds Max welche Teile der Grafik auf welche Seite des Objekts „geklebt“ werden.

3.6. Sounddesign

Die passende Geräuschkulisse trägt neben dem realistischen Aussehen der Umgebung entscheidend zu einer passenden Atmosphäre bei. Außerdem ist auch akustisches Feedback für manche Aktionen sinnvoll und nötig.

Insbesondere deshalb, weil es eine einfache und leicht zu realisierende Möglichkeit darstellt, Ergebnisse einer Aktion zu verdeutlichen. Im Gegensatz zu Videodateien lassen sich Audiodateien in DX Studio über die integrierten Befehle problemlos steuern.

Bei den Geräuschen galt es jedoch auch, ein gesundes Mittelmaß zu finden, so dass zwar eine passende Geräuschkulisse gegeben ist, aber auch nicht zu viele Geräusche da sind, die verhindern könnten, dass akustisches Feedback nicht wahrgenommen wird.

Insbesondere werden Audiodateien an folgenden Stellen abgespielt:

- Betätigen eines Buttons in bestimmten Menüs
- Öffnen und Schließen des Rolltors
- Interaktion mit dem Telefon während bestimmter Aufgaben
- Paketmaschine im laufenden Betrieb
- Brand der Maschine
- Betätigen des Feuerlöschers
- Erfolgreiches Abschließen eine Aufgabe
- Scheitern einer Aufgabe

3.7. Quellen von Texturen und Sounds

Wie in den vorhergehenden Kapiteln bereits beschrieben tragen hochwertige Texturen und passende Audiodateien erheblich zur Qualität des Spiels und einer passenden Atmosphäre bei. Solche Texturen und Audiodateien anzufertigen ist aufwändig und war daher für uns in der begrenzten Zeit nicht vollständig möglich.

Was wir selbst erstellten, waren die passenden Texturvorlagen für unsere 3D-Modelle. Doch die Basistexturen für diese Vorlagen entstammten anderen Quellen. In unserem Fall bedienten wir uns teilweise bei anderen Spielen. Zwar liegen diese meistens in speziellen, gepackten Formaten vor, so dass man nicht ohne Weiteres an die Rohdaten herankommt doch einige Spiele bieten Tools zum Entpacken von Dateien oder sind sogar gänzlich ungepackt. Für gewöhnlich sind dies Spiele, die es den Spielern leicht machen wollen, das Spiel nach ihren Wünschen zu modifizieren und zu erweitern. Solche sogenannten Mods sind eine übliche Erscheinung in der Spielewelt und werden nicht dafür eingesetzt, sich einen

Vorteil gegenüber andren Spielern zu verschaffen, sondern die Spielerfahrung zu erweitern z.B. durch neue Spielwelten oder Gegenstände.

Neben den Spielen gibt es auch eine große Anzahl an Webseiten, die Texturen zur Verfügung stellen. Auch von solchen Seiten stammen einige Texturen innerhalb des Spiels.

Eine wichtige Quelle für Texturen und Audiodateien war für uns das Spiel Söldner –Secret Wars, das mittlerweile kostenlos im Netz verfügbar ist. Hieraus stammt ein Großteil der Texturen für Wände, Decke und Türen sowie die Audiodateien für Motorengeräusche, das Klickgeräusch der Buttons sowie das Geräusch des Feuers. Weitere Audiodateien stammen aus dem Spiel Emergency 4, insbesondere Sirenengeräusche, Explosionsgeräusche und das Geräusch des Feuerlöschers.

Bei einem kommerziell genutzten Spiel wäre ein solches Verhalten natürlich wenig sinnvoll. Entweder müssten die Dateien selbst erstellt werden oder von entsprechenden Anbietern käuflich erworben werden, was nicht unerheblich zu den Entwicklungskosten eines Spiels beiträgt.

3.8. Export des Spiels

DX Studio bietet unterschiedliche Möglichkeiten, das Spiel zu exportieren. Die einfachste Möglichkeit ist das Abspeichern im .dxstudio-Format. Hierbei handelt es sich um ein Format, das ein gezipptes Paket mit allen nötigen Spieldateien enthält. Es ist sowohl ein Arbeitsformat als auch ein Abspielformat, das bedeutet, dass man .dxstudio-Dateien also sowohl mit dem Editor weiterbearbeiten kann, als auch mit dem Player spielen. Nachteil hierbei ist: Der Anwender muss sich den DX Player herunterladen, um das Spiel spielen zu können. Der Player ist zwar grundsätzlich kostenlos, muss aber heruntergeladen und installiert werden, was Zusatzaufwand bedeutet. Außerdem kann das Paket ohne technischen Aufwand entpackt und manipuliert werden. Dazu muss man lediglich die Dateiendung ändern und schon kann man die Spielinhalte in leicht editierbaren Formaten, vor allem XML, manipulieren.

Die deutlich bessere Exportoption ist daher der Export als ausführbare Datei (.exe). Hierbei kann entweder eine ausführbare Datei exportiert werden, die alle nötigen Komponenten, einschließlich Player beinhaltet und keine Installation braucht. Dies hat den Vorteil, dass die Datei auch ohne Administratorrechte ausgeführt werden kann. Dadurch war es uns sogar möglich, das Spiel auf Hochschulrechnern zu spielen. Es ist jedoch auch möglich eine Version ohne integrierten Player zu exportieren.

Eine weitere Möglichkeit für einen Export als ausführbare Datei ist der Export als Installationspaket. Dabei muss man vor dem Spielstart eine Installationsprozedur durchlaufen werden, wie man sie von vielen anderen Programmen auch kennt.

Theoretisch ist auch ein Export als HTML möglich, dieser funktionierte bei uns jedoch nicht. Weitere Exportformate, die in unserem Fall jedoch nicht sinnvoll waren: Bildsequenz, Video (AVI oder GIF) und Windwos Sidebar Gadget.

4. Ergebnisse

4.1. Das Lernspiel

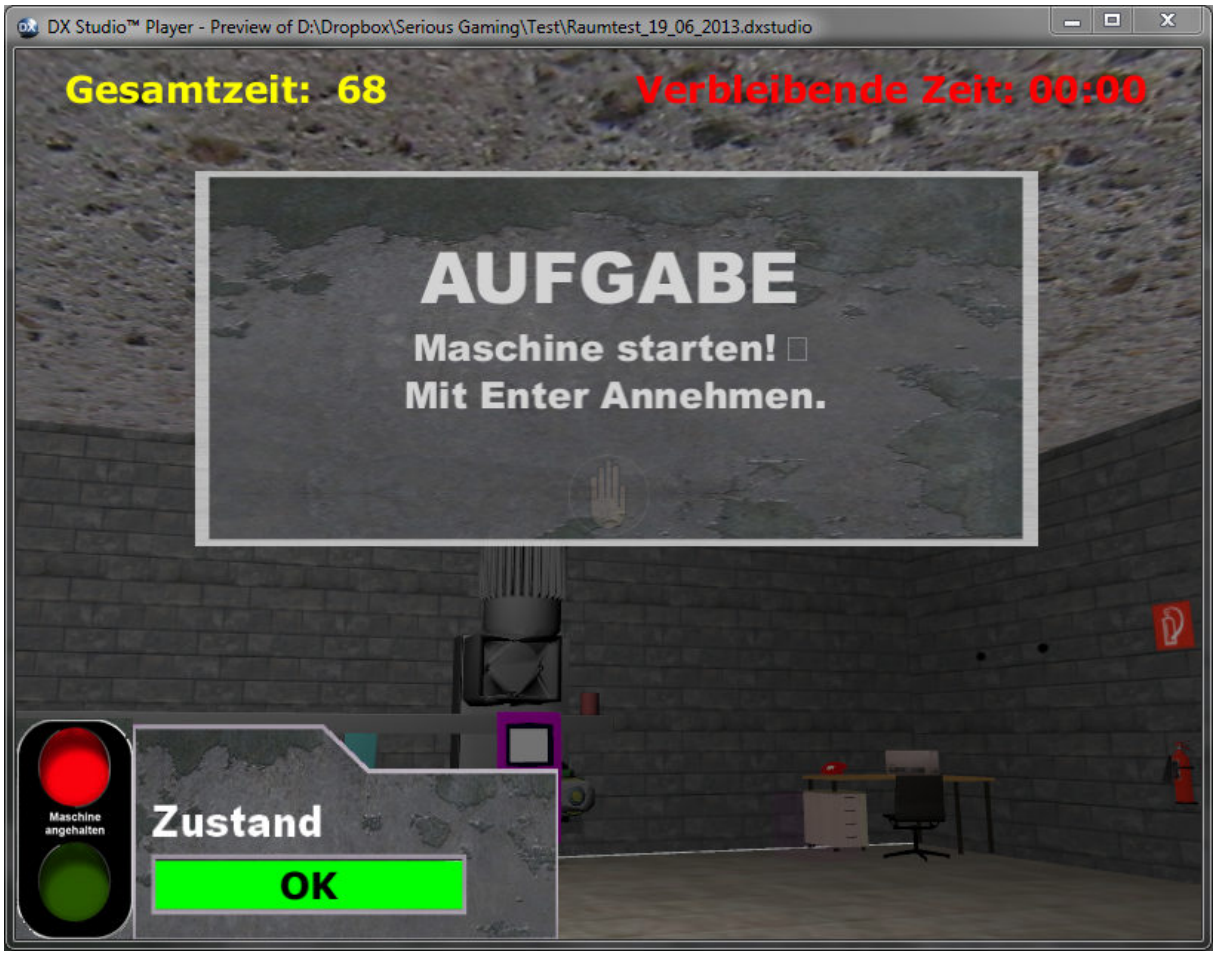


Abbildung 13: Darstellung des Spiels im DX Player

Trotz einiger Schwierigkeiten und Änderungen konnten wir am Ende ein mehr oder weniger komplettes Spiel fertig stellen. Zwar fehlt nach wie vor die Möglichkeit, das Lehrvideo abzuspielen, das eigentliche Spiel jedoch funktioniert jedoch mit gewissen Einschränken wie geplant.

4.1.1. Nicht umgesetzte Funktionen

Folgende ursprünglich vorgesehenen Funktionen gibt es in der aktuellen Version nicht:

- Lehrvideo
- Flüssige Darstellung des Spiels
- Einige Aufgaben an der Maschine
- Unterschiedliche Bedienelemente an der Maschine
- Funktionierender Countdown

Bis auf das fehlende Lehrvideo und die derzeit noch hakelige Darstellung sind die nicht realisierten Funktionen auf Zeitmangel zurückzuführen, da das gesamt Projekt doch mehr Zeit benötigte, als ursprünglich geplant. Mit mehr Zeit wäre es mit überschaubarem Aufwand möglich, diese Funktionen umzusetzen.

Die anderen fehlenden Funktionen sind eher auf Probleme innerhalb der Engine zurückzuführen und bedürfen einer etwas ausgiebigeren Analyse, ließen sich aber mit einigem Aufwand auch noch integrieren.

4.2. Praktischer Einsatz

Da der Paketomat 3000 ein fiktives Produkt ist, wird das Spiel in dieser Form wohl kaum zum Einsatz kommen. Im aktuellen Zustand des Spiels wäre dies auch wenig sinnvoll. Zumindest die Darstellung des Spiels muss auf jeden Fall noch flüssiger werden, bevor es produktiv eingesetzt werden kann.

Grundsätzlich jedoch ist der Einsatz eines Spiels dieser Art denkbar und in unseren Augen auch sinnvoll. Eine virtuelle Welt erlaubt es, Anwender in Situationen zu bringen, die in der Realität schwer umzusetzen sind. In diesen Situationen kann Verhalten durchgespielt werden, das in der Realität nur theoretisch erklärt werden kann.

Als Basis für reale Projekte ist das von uns entwickelte Spiel sicherlich zu gebrauchen, sofern die aktuell bestehenden Probleme mit Performance und der Darstellung von Videos behoben werden. Auch der Einsatz größerer Maschinen stellt kein größeres Problem dar, da die Größe des Raums relativ leicht angepasst werden kann.

4.3. Probleme während der Entwicklung

Natürlich verläuft ein so komplexes Projekt wie ein Spiel nicht gänzlich ohne Probleme. Die größten Probleme, mit denen wir zu kämpfen hatten waren:

- Die bereits erwähnten Probleme bei der Programmierung des Spiels (Timer, Videoeinbindung)
- Fehlerhafter Import von 3D-Daten aus Blender
- Ruckelige Darstellung des Spiels ohne erkennbare Gründe, unabhängig von der Leistungsfähigkeit des Rechners
- Nur bedingt hilfreiche Benutzerdokumentation der Engine, zwar waren alle Scriptelemente und deren Funktion erklärt, aber nicht deren korrekter Einsatz und entsprechende Beispiele
- Darstellung von Sonderzeichen, insbesondere von Umlauten in Texten, die per Script erzeugt wurden

4.4. Erweiterungs- und Entwicklungsmöglichkeiten

Zu Projektbeginn hatten wir eine Vielzahl an Ideen, wie das Spiel im fertigen zustand aussehen könnte. Die Realität jedoch machte uns bald klar, dass wir innerhalb der recht begrenzten Zeit nicht alles verwirklichen konnten.

Insbesondere war zu Beginn der Entwicklung eine größere Vielfalt an Aufgaben an der Maschine geplant. Insbesondere auch Aufgaben, die eher dem Alltagsgeschäft entsprachen. Deren Realisierung erwies sich aber als nicht ganz so trivial, wie ursprünglich angenommen. Zudem wäre es auch möglich, die bestehenden Aufgaben komplexer zu gestalten, indem z.B. bei der Bedienung des Telefons auch eine korrekte Telefonnummer hätte eingegeben werden müssen.

Eine weitere Idee, um die Aufgaben des Spiels interessanter zu gestalten war ein Spielmodus, in dem die Aufgaben in zufälliger Reihenfolge durchgespielt werden. Dies erhöht den Schwierigkeitsgrad für den Anwender, da dieser nicht wissen kann, welche Aufgabe zu welchem Zeitpunkt kommt. Insbesondere in Kombination mit zufälligen Timern kommt so ein relativ realistischer und unberechenbarer Spielverlauf zu Stande.

Ein weiterer Punkt, an dem noch gearbeitet werden kann ist die Performance des Spiels. Momentan wird das Spiel noch sehr ruckelig wiedergegeben, obwohl die Hardware der meisten Test-PCs damit nicht voll ausgelastet war. Die Vermutung liegt daher nahe, dass bestimmte Abschnitte im Script das Spiel ausbremsen. Welche Abschnitte das genau sind, lässt sich jedoch nur schwer sagen.

Wenn die Performance-Probleme behoben sind, könnte man sich auch an einer noch detaillierteren und realistischeren Welt versuchen. Momentan sind die Lichtquellen zwar vorhanden, aber nicht optimal ausgerichtet und einige typische Objekte, wie z.B. Lampen fehlen noch völlig. Die detaillierte Ausgestaltung der Welt bringt dem Lerneffekt des Spiels zwar keinen wirklichen Mehrwert, bringt aber einen ästhetischen Mehrwert ins Spiel.

Derzeit verfügt das Spiel noch über keine theoretische Wissensabfrage. Diese ließe sich aber bei Bedarf mit relativ wenig Aufwand, z.B. in Form von Multiple-Choice-Tests realisieren. Diese Tests könnten direkt nach dem Ansehen des Videos durchgeführt werden oder vor dem Betreten des Maschinenraums. Neben Multiple-Choice-Fragen wären auch Texteingaben denkbar.

Die Ergebnisse des Tests könnten in einem Highscore festgehalten werden, ebenso die Gesamtzeit, die zur Erfüllung aller Aufgaben benötigt wurde. So hat der Anwender auch die Möglichkeit, seine Leistung mit der anderer Anwender zu vergleichen. Der schulende Betrieb wiederum hat die Möglichkeit, mit Hilfe der Highscores die Leistung der Mitarbeiter zu beurteilen und kann nötigenfalls Nachschulungen o.ä. anordnen. Wahlweise ließe sich das Spiel auch zur Beurteilung von Bewerbern eingesetzt werden.

Je nach Anforderungen könnten neben den Testergebnissen und der Gesamtdauer auch andere Daten erfasst werden, z.B. Dauer einzelner Aufgaben, die Anzahl an Neustarts etc.

4.5. Zeitplanung

4.5.1. Geplant

Bei Projektbeginn stellten wir eine grobe Zeitplanung auf, die sowohl den zeitlichen Projektverlauf als auch den geschätzten Stundenaufwand für die entsprechenden Aufgaben

enthielt. Der Stundenaufwand entsprach hierbei dem voraussichtlichen Arbeitsaufwand für zwei Personen.

Folgender Projektverlauf war geplant. Dabei war uns durchaus bewusst, dass einige der Aufgaben nicht unbedingt in dieser Reihenfolge ausgeführt werden würden, sondern vielmehr parallel ablaufen würden. Daher repräsentieren die aufgelisteten Aufgaben stets nur die Entwicklungsschwerpunkte.

Woche	Aufgabe
W1 (25.03. – 31.03.2013)	Expose fertigstellen, Entwicklungsumgebung installieren (8h)
W2 (01.04. – 07.04.2013)	3D-Modelle (20h) und 2D-Texturen (5h) erstellen
W3 (08.04 – 14.04.2013)	3D-Modelle (20h) und 2D-Texturen (5h) erstellen
W4 (15.04 – 21.04.2013)	3D-Modelle (12h) und 2D-Texturen/GUI (15h) erstellen
W5 (22.04 – 28.04.2013)	Animationen erstellen (25h)
W6 (29.04. – 05.05.2013)	Einarbeiten in Scriptsprache (10h) Scripting (20h)
W7 (06.05. – 12.05.2013)	Scripting (20h)
W8 (13.05. – 19.05.2013)	Scripting (20h)
W9 (20.05. – 26.05.2013)	Sounddesign (15h)
W10 (27.05. – 02.06.2013)	Debugging, Betatest (5 – 25h)
W11 (03.06. – 09.06.2013)	Ausarbeitung schreiben (20h)
W12 (10.06. – 16.06.2013)	Ausarbeitung schreiben (20h)
W13 (17.06. – 23.06.2013)	Endkorrektur (10h)
W14 (24.06. – 30.06.2013)	Pufferzeit

Insgesamt entsprach dies dem Umfang von ca. 240 Stunden, was bei gleicher Arbeitsverteilung etwa 120 Stunden pro Person entspricht.

4.5.2. Tatsächlich benötigt

Die folgende Tabelle gibt einen Überblick und einen Vergleich über die tatsächlich durchgeführten Aufgaben und die dafür aufgewendete Zeit.

Woche	Aufgabe (geplant)	Aufgabe (tatsächlich)
W1 (25.03. – 31.03.2013)	Expose fertigstellen, Entwicklungsumgebung installieren (8h)	Expose fertigstellen, Entwicklungsumgebung installieren (8h)
W2 (01.04. – 07.04.2013)	3D-Modelle (20h) und 2D-Texturen (5h) erstellen	Einarbeiten in die Engine (15h)3D Modelle erstellen (10h) 2D Texturen (2h)
W3 (08.04 – 14.04.2013)	3D-Modelle (20h) und 2D-Texturen (5h) erstellen	3D-Modelle (20h) und 2D-Texturen (5h) erstellen
W4 (15.04 – 21.04.2013)	3D-Modelle (12h) und 2D-Texturen/GUI (15h) erstellen	3D-Modelle (12h) und 2D-Texturen/GUI (15h) erstellen
W5 (22.04 – 28.04.2013)	Animationen erstellen (25h)	3D-Modelle (5h) und 2D-Texturen, GUI-Elemente und Programmierung (20h)
W6 (29.04. – 05.05.2013)	Einarbeiten in Scriptsprache (10h) Scripting (20h)	Einarbeiten in Scriptsprache (4h)

		Scripting* (10h), 3D-Modelle erstellen und einfügen (10h)
W7 (06.05. – 12.05.2013)	Scripting (20h)	Scripting* (20h)
W8 (13.05. – 19.05.2013)	Scripting (20h)	Scripting* (20h)
W9 (20.05. – 26.05.2013)	Sounddesign (15h)	Scripting* (15h)
W10 (27.05. – 02.06.2013)	Debugging, Betatest (5 – 25h)	Scripting* (10h)
W11 (03.06. – 09.06.2013)	Ausarbeitung schreiben (20h)	Scripting* (20h)
W12 (10.06. – 16.06.2013)	Ausarbeitung schreiben (20h)	Scripting* (15h) 2D-Elemente für GUI (5h)
W13 (17.06. – 23.06.2013)	Endkorrektur (10)	Scripting* (20h)
W14 (24.06. – 30.06.2013)	Pufferzeit	Präsentation und Projektdokumentaion estellen, inkl. Korrekturen (40h)

* In den Zeiten für Scripting ist auch ein gewisser, nicht genauer zu quantifizierender Anteil an Tests du Debugging enthalten, da wir diese Schritte meist direkt nach dem Umsetzen neuer Funktionen durchführten, um Fehler direkt so gut es ging zu beheben.

Weiterhin verteilen sich über den gesamten Projektzeitraum noch etwa 2 Stunden Sounddesign. Durch brauchbare Quellen für passende Audiodateien, die keiner weiteren Bearbeitung bedurften, war die Arbeitszeit hier deutlich geringer als geplant.

Insgesamt benötigten wir also ca. 300 Stunden für das gesamte Projekt, also etwa 30% mehr, als ursprünglich vorgesehen. Die Gründe hierfür wurden in diesem Projektbericht aufgeführt. Der gesamte Arbeitsaufwand verteilte sich in etwa gleichmäßig auf beide Projektmitglieder, wobei hier unterschiedliche Aufgabenschwerpunkte gesetzt wurden, was durch Fähigkeiten und Interessen der Projektmitglieder bedingt war.

4.6. Lessons Learned

Mit jedem durchgeführten Projekt, sammelt man wertvoller Erfahrungen und Erkenntnisse. Auch wir haben während der Projektdurchführung vieles gelernt.

In erster Linie haben wir erfahren, was für ein vielfältiges Wissen und Fähigkeiten die Entwicklung eines Spiels voraussetzt. Um wirklich hochwertige Spiele jeglicher Art benötigt man Expertenwissen in unterschiedlichen Tätigkeitsfeldern. Sei es beim Erstellen von 3D-Modellen, dem Bearbeiten der dazugehörigen, zweidimensionalen Texturen, beim Programmieren oder dem Ausarbeiten eines stimmigen Konzepts.

Hierbei war es für uns hilfreich, das Projekt zu zweit bearbeiten zu können. Jeder von uns beiden hatte individuelle Wissens- und Interessensschwerpunkte, so dass wir uns gut ergänzten. Insbesondere auch bei komplexeren Programmiervorhaben war der regelmäßige Austausch über Ideen und Möglichkeiten hilfreich, da vier Augen auch in der Dokumentation der Engine mehr sehen, als nur zwei. Dies löste zwar nicht alle Probleme, schließlich sind wir beide keine erfahrenen Programmierer, beschleunigte aber viele Problemlösungsvorgänge erheblich.

Dennoch war es nicht immer möglich, Probleme mit Hilfe der Dokumentation der Engine zu beheben. Zwar gibt es grundsätzlich eine recht umfangreiche Dokumentation, die jedoch bei weitem nicht vollständig und einsteigerfreundlich ist. Dadurch, dass DX Studio nur von wenigen Entwicklern eingesetzt wird, ist auch die Möglichkeit, an anderer Stelle Hilfe zu finden recht eingeschränkt. Nahezu jeder Suchvorgang führte uns immer wieder auf die offiziellen DX Studio Homepages zurück, die wir schon zu vor konsultiert hatten.

Bei zukünftigen Entwicklungsvorhaben würden wir daher vorher einen deutlicheren Blick auf die Dokumentation und vor allem auch ihre Inhalte liefern. Eine zuverlässige Quelle für Codebeispiele ist für unerfahrene Programmierer wie uns sicherlich unerlässlich.

Außerdem mussten wir erfahren, dass Murphy's Law auch vor uns und unserem Projekt keinen Halt machte und manch banal scheinender Code zu stundenlanger Fehlersuche führt, nur um schließlich das Problem auf eine andere Weise zu lösen. Durch diese dich recht häufigen Fälle, verzögerte sich auch der Entwicklungsfortschritt immer wieder unerwartet, was letztendlich auch dazu führte, dass wir in der Endfassung des Spiels nicht alle unsere Ideen vollständig umsetzen konnten.

Auch hieraus ziehen wir Lehren und werden bei zukünftigen Projekten versuchen, solche unvorhergesehenen Ereignisse noch besser einzuplanen, so gut dies möglich ist und generell keine zu hohen Ziele zu stecken.

4.7. Fazit

Trotz kleinerer und gelegentlich auch etwas komplexerer Probleme, mit denen wir während der Konzeptionierung und Programmierung des Paketomat 3000 konfrontiert wurden kann man abschließend festhalten dass das Fazit von unserer Seite aus durchweg positiv ausfällt. Die Entwicklung eines Computerspiels ist ein enorm interessantes und vielfältiges Feld, bei dem sowohl spezielles Expertenwissen als auch interdisziplinäres Know-How und eine gute Kommunikation unter den einzelnen Projektmitarbeitern gefragt ist. Es ist aber auch eine aus dem Nichts erschaffene kreative Leistung, bei der man sich nach langem Tüfteln erfreut zurücklehnen kann, wenn die Maschine in Flammen aufgeht oder ein Rolltor korrekt öffnet und schließt.

Es ist anzunehmen, dass Serious Games in den kommenden Jahren in der beruflichen Aus- und Weiterbildung weiterhin an Relevanz gewinnen werden. Sie können auch Mitarbeiter erreichen, die anderen Bildungsmaßnamen weniger aufgeschlossen gegenüberstehen und bieten aufgrund ihrer interaktiven Interaktionsmöglichkeiten eine hervorragende Möglichkeit, den Mitarbeiter im Umgang mit komplexen Verfahrensabläufen auszubilden, ohne Material- oder Personenschäden befürchten zu müssen.

Dabei ist allerdings zu beachten, dass die Entwicklung eines qualitativ hochwertigen Serious Games ebenso wie die Entwicklung eines unterhaltungsorientierten Computerspiels neben einer gewissen Entwicklungszeit und finanziellen Mitteln idealerweise ein Team aus spezialisierten Experten voraussetzt; einerseits im Bereich der Entwicklung (Game Designer,

Programmierer, Grafiker etc.), andererseits im Bereich der Lehre (Pädagogen, Experten in der jeweiligen Wissensdomäne).

Wenngleich das Potential von Serious Games für pädagogische Ziele grundsätzlich als durchaus hoch einzuschätzen ist, sollte jedoch auch auf einige Grenzen hingewiesen werden. So stehen pädagogische Angebote im Spielbereich grundsätzlich vor der Herausforderung, einerseits niedrige Systemanforderungen zu stellen, um auch Nicht-Spieler mit ungünstigen technischen Voraussetzungen zu erreichen, andererseits aber in Bezug auf Spielegrafik und Gameplay mit Entertainment-Titeln konkurrieren zu können. Beide Faktoren sind mit Blick auf potentielle Nutzergruppen von Serious Games sorgfältig abzuwägen. Konkret auf den Paketomat 3000 und vergleichbare interaktive Lernanwendungen zur Erwachsenenbildung bezogen, deren Ziel es langfristig sein sollte Mitarbeiter für die Arbeit an komplexen Maschinen auszubilden bedeutet dies: die Performance des Spiels und ein flüssiger Ablauf der Missionen sind von zentraler Bedeutung um das Spiel für möglichst viele potentielle Arbeiter nutzbar zu machen, allerdings sollte die grafische Qualität des Spiels schon aus Gründen der Immersion keinesfalls vernachlässigt werden. Die Entwicklung des Serious-Games-Marktes steht zumindest in Deutschland noch am Anfang. Es ist jedoch absehbar, dass sich das Angebot noch deutlich erweitern wird. Für eine weitere Diskussion über die Potentiale und Wirkungen dieser Spielegattung wären entsprechende Nutzungs-, Evaluations- und Wirkungsstudien wünschenswert, um ihre Reichweite, ihren Unterhaltungswert, ihr pädagogisches Potential und ihre Akzeptanz vor allem außerhalb der Zielgruppe der „Digital Natives“ empirisch fundiert einschätzen zu können. Denn letztendlich ist vor allem für das Unternehmen, dass ein Serious Game zur Schulung ihrer Mitarbeiter einsetzen möchte entscheidend, inwieweit und unter welchen Bedingungen der Wissenstransfer vom Spiel in die Realität gelingt bzw. gelingen kann. Inhaltliche und gestalterische Aspekte werden hierbei ebenso zu berücksichtigen sein, wie der lebensweltliche und spielbezogene Kontext des Spielers.

5. Anhänge

5.1. Quellenverzeichnis

(1) Hugger, K.-U. & Walber, M. (2010) Digitale Lernwelten: Konzepte, Beispiele und Perspektiven. VS Verlag für Sozialwissenschaften

(2) Csikszentmihalyi, M. (2010). *Das flow-Erlebnis. Jenseits von Angst und Langeweile: im Tun aufgehen.* 10. Auflage. Stuttgart: Klett-Cotta.

(3) Gee, J. (2008) What Video Games Have to Teach Us About Learning and Literacy

(4) Michael, D. & Chen, S. (2005) Serious Games: Games That Educate, Train, and Inform

(5) Prensky, M. (2001) The Digital Game-Based Learning Revolution

(6) Ritterfeld et al. (2009) Serious Games: Mechanisms and Effects

(7) Wenzler, I. (2008). Simulations as learning from the future

5.2. Bilder

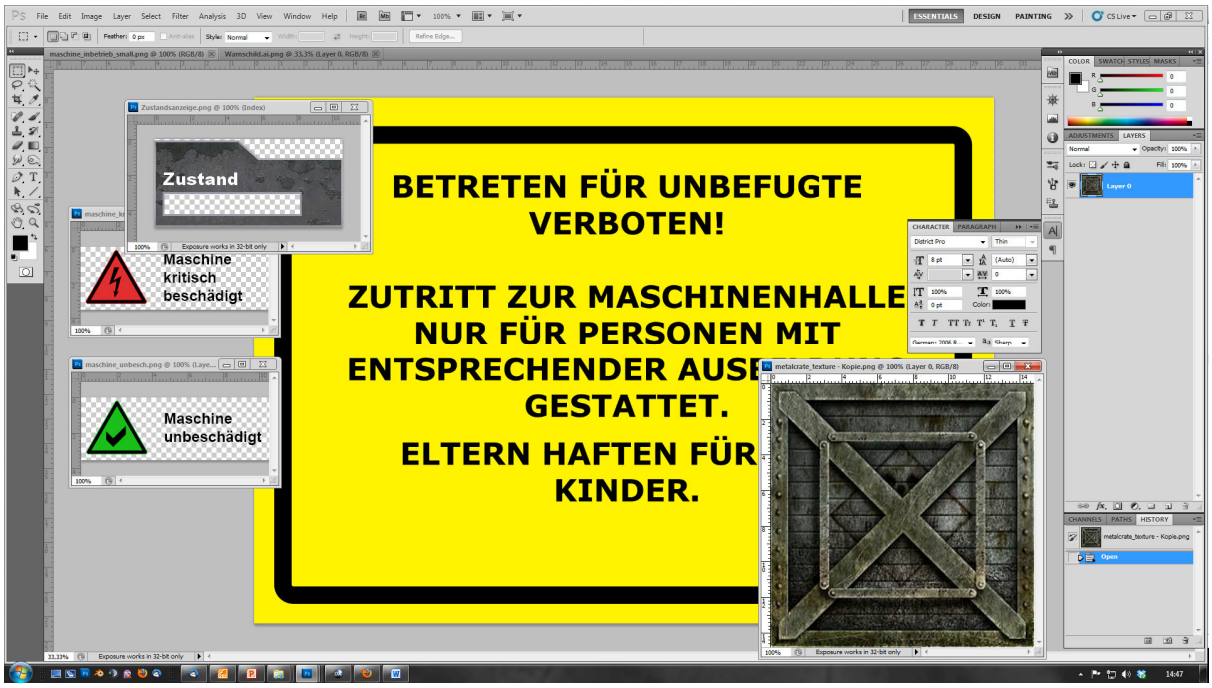


Abbildung 14: Einige verwendete 2D-Elemente in Adobe Photoshop

Abbildung 15: Programmierfehler mit überraschenden Konsequenzen

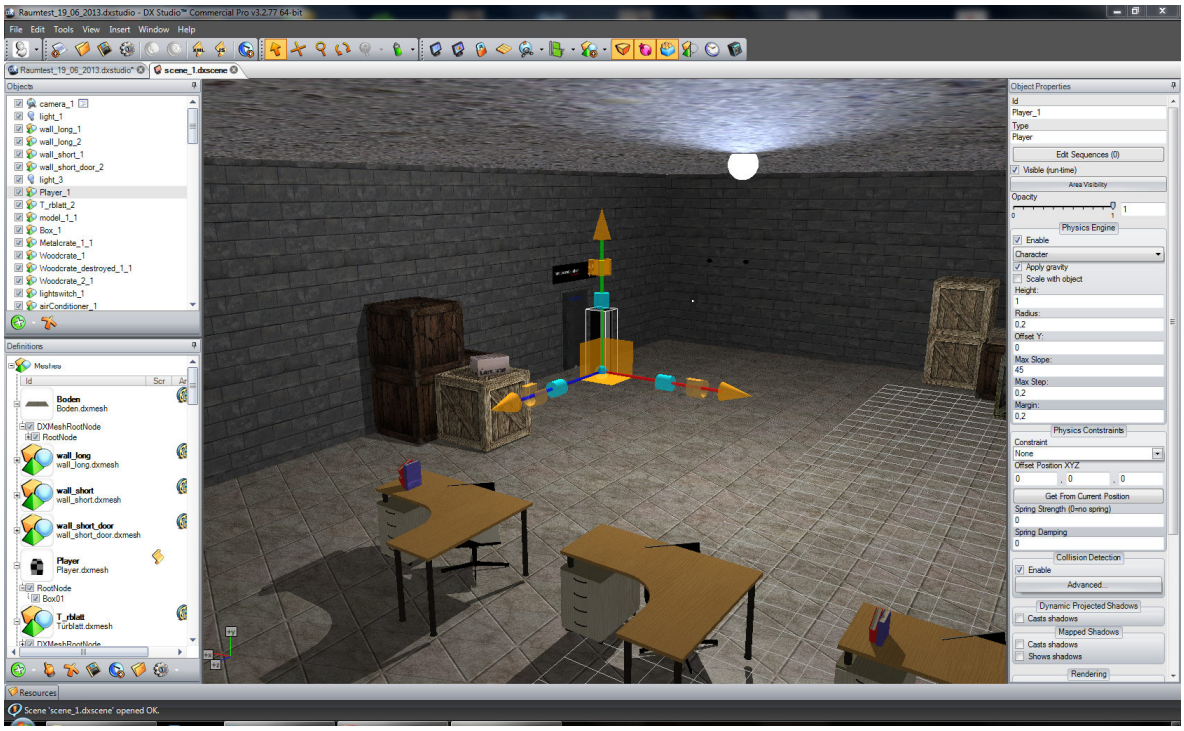


Abbildung 16: Objektdetails im Editor

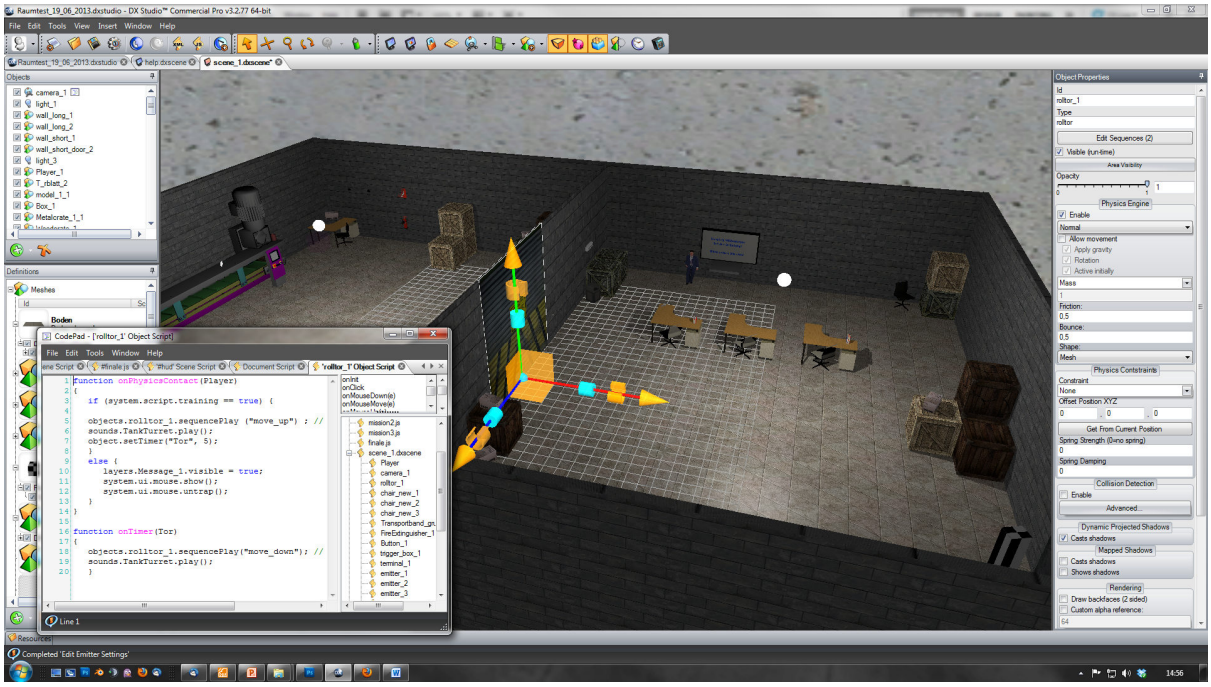


Abbildung 17: Objektdetails mit Objektscript im Editor

Abbildung 18: 3D-Umgebung in einer frühen Entwicklungsphase

Abbildung 19: 3D-Umgebung in einer mittleren Entwicklungsphase, noch ohne das 3D-Objekt der Verpackungsmaschine

5.3. Code

An dieser Stelle folgen einige Codebeispiele, um die Scripte des Spiels zu illustrieren.

5.3.1. Dokumentscript

Das folgende Script definiert auf Dokumentebene die Standardwerte von Variablen und die Sichtbarkeit von Ebenen.

```
// Initial Setup
var training = false;
var width = 800;
var height = 600;
var gamestarted = false;
var initialEntry = true;
var mission = 1;
system.exitWithEscape = false;
system.window.fullScreenEscape = false;
function onInit()

{

system.ui.mouse.trap();

system.ui.mouse.show();

scenes.scene_1.objects.Boden_01.selectionMask = 1;

layers.Message_1.visible = false;
layers.Message_2.visible = false;
layers.Message_3.visible = false;
layers.HUD.visible = false;
layers.Stage_3D.visible = false;
layers.mission_goal.visible = false;
layers.mission_fail.visible = false;
layers.mission_success.visible = false;
layers.help.visible = false;
scenes.scene_1.objects.emitter_1.visible = false;
scenes.scene_1.objects.emitter_3.visible = false;
layers.Message_3.visible = false;
layers.finale.visible = false;
            }
```

5.3.2. Objektscript Verpackungsmaschine

Das folgende Script ist dem eigentlichen Paketomat 3000 zugeordnet und definiert das verhalten des Spiels bei bestimmten Ereignissen.

```
function onTimer (Mission3) {
      //@"mission3.js"
      }

function onCollisionBegin (Player_1){
      if (scenes.scene_1.script.equip == "fire"){
            scene.sounds.fireextinguisher02.play();
            scene.objects.emitter_2.emitter.active = false;
            scene.sounds.fire.stop();

      scenes.scene_1.objects.Transportband_gruppiert_1.killTimer("destructi
on");
```

```
            if (scenes.scene_1.objects.emitter_2.emitter.active == false &&
scenes.scene_1.script.call_fire == true){
                  scenes.scene_1.objects.Player_1.setTimer("finale", 10);
                  }
            else {}
            }
      else{}


      }
```

5.3.3. Externes Script für Aufgabe 3

Dieses Script wird aufgerufen, sobald der Paketomat 3000 den Timer „Mission3“ auslöst. Es ist im Objektscript der Verpackungsmaschine in Zeile 2 refrenziert, dort ist auch der Event-Listener definiert.

```
layers.mission_success.visible =false;
scenes.scene_1.sounds.fire.play();
      scenes.scene_1.objects.emitter_2.emitter.active = true;
      scenes.scene_1.script.machine_state = 3;
      scenes.mission_goal.objects.missionsziel.text = "Feuer! \r\n Feuer
löschen, Feuerwehr anrufen \r\n Mit Enter Annehmen.";
layers.mission_goal.visible = true;
scenes.scene_1.objects.terminal_1.setTimer("destruction", 180);
alert ( "Destruction Timer wurde ausgelöst!" );
```

5.4. Rohdateien

Aufgrund der Größe der Spieldatei ist diese nicht als Datei in dieser Projektabgabe enthalten, gleiches gilt für die gepackte Spielversion, die mit integriertem Player verfügbar ist und ein Paket mit entpackten Rohdaten.

Diese stehen als Download zur Verfügung.

Die Rohdateien befinden sich unter:

https://dl.dropboxusercontent.com/u/1536817/Serious%20Games/Abgabe_Entpackt.zip

Hinweis: Die vollständigen JavaScriptCodes werden von DXStudio innerhlab der xml-Dateien abgelegt und sind daher nicht direkt verfügbar. Ausnahme hiervon bilden extern eingebundene Scripte, diese sind als Einzeldateien verfügbar.

Das gepackte Spielpaket befindet sich unter:

https://dl.dropboxusercontent.com/u/1536817/Serious%20Games/Abgabe_Entpackt.zip

Hinweis: Das gepackte Paket ist ohne Installation und Player lauffähig, wird jedoch von manchen Antivirenprogrammen als potenzielle Gefahr eingestuft, daher ist die ausführbare Date zusätzlich in einer zip-Datei verpackt.

Die Spieldatei steht zur Verfügung unter:

https://dl.dropboxusercontent.com/u/1536817/Serious%20Games/Abgabe_Paketomat.dxstudio

Hinweis: Diese Datei benötigt zum Öffnen entweder den DX Studio Player oder die DX Studio Entwicklungsumgebung. Die Spieldatei liegt im nativen Format von DX Studio vor und kann mit dem Editor weiterbearbeitet werden.