Projektdokumentation - Projekt 3

(Kevin Niehage – Matrikelnummer: 280888)

Cubes³

(Jump'n'Run Geschicklichkeitsspiel)

TAI06ABC - Berufsakademie Mannheim
Projektarbeit im Rahmen der Vorlesung Software Engineering II – 2008
Dozent: Dipl.-Wirt.Ing. K. Koochaki

Abstract

Dieses Dokument enthält die Projektdokumentation zu dem Spiel **Cubes³**, das im Kurs **TAI06ABC** im Zuge der Vorlesung **Software Engineering II** unter der Leitung des Dozenten **Dipl.-Wirt.Ing. K. Koochaki** an der **Berufsakademie Mannheim** erstellt wurde.

Sie ist Teil der Bewertungsanforderung der oben genannten Vorlesung und ersetzt, zusammen mit der erstellten Software, die Klausur in diesem Fach.

Das vorliegende Dokument enthält eine möglichst vollständige Übersicht über das Gesamtprojekt – von der Projektidee, über die Planung, bis hin zur Realisierung und dem aktuellen Status.
Ebenso ist eine kurze Einleitung in die Benutzung enthalten, die sich über die Installation, Bedienung und Deinstallation erstreckt.
Abschließend enthält diese Dokumentation noch ein Gesamtfazit über den Ablauf des Projektes.

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

Projektmotivation

Grundlage dieses Projektes war die Aufgabenstellung in der *Software Engineering II* Vorlesung, anstatt eine Klausur zu bestehen, ein Softwareprojekt umzusetzen.

Basisanforderung war das Entwerfen und Entwickeln eines Spieles, das über eine akzeptable Grafik (keine graphisch anspruchslosen Minispiele) und einen Realtime-Multiplayer-Modus verfügt.

Diese Anforderungen konnten teilweise durch andere Anforderungen ersetzt werden, falls das gewählte Spielprinzip oder die gewählte Umsetzung dies erforderlich machte.

Nachträglich wurde die Anforderung, zusätzlich einen Leveleditor und eine Replay-Funktion zu entwickeln, für fast alle Projektgruppen nachgereicht.

Projektidee

Die Idee bei **Cubes³** war es, ein dreidimensionales Jump'n'Run Spiel zu entwerfen – dieses sollte jedoch nicht, wie in dem Genre eigentlich üblich, einen festgelegten Spielverlauf haben.

An Stelle der festgeschriebenen Spielwelten sollte es möglich sein, eigene Welten zu designen und zu bespielen - weiterhin sollten die Welten surreal wirken.

Aus diesen Ideen entwickelte sich schnell **Cubes³** als Gesamtkonzept – ein Spiel, dessen Welt ausschließlich aus Quadern besteht, die sich um ihre eigenen Achsen rotieren und einen Weg vom Anfangspunkt zum Zielpunkt bahnen.

Die Idee, die gesamte Welt um einen Punkt rotieren zu lassen, wurde nach den ersten Gesprächen mit Dipl.-Wirt.Ing. K. Koochaki fallen gelassen.

Planung

Projektstruktur

Projektteam

Das Projektteam bestand lediglich aus Kevin Niehage als einzigem Teammitglied – dieser musste sich um alle Aspekte des Projektes selbstständig kümmern, von der Planung, über die Implementation, bis hin Dokumentation und Präsentation des Projektes.

Dieser Umstand brachte sowohl eine immense Mehrarbeit, jedoch auch eine einfachere zeitliche Planung mit sich.

Projektablauf

Das Projekt wurde in Form mehrerer Etappen absolviert, wobei jede Etappe („Milestone“) vom Dozenten, Dipl.-Wirt.Ing. K. Koochaki, abgenommen und für die Bearbeitung der nächsten Etappe freigegeben wurde.

Bei der ersten Etappe handelt es sich um das Sammeln von Projektideen – in diesem Schritt wurde die oben geschriebene Idee für das Spiel **Cubes³** entworfen (ebenso wie zwei oder drei weitere Projektideen, die jedoch verworfen wurden).

Im zweiten Schritt fand die Anforderungsanalyse statt, deren Resultat unter anderem die Festlegung der funktionalen und nicht-funktionalen Anforderungen war – die hier festgelegten Anforderungen bildeten die Basis der späteren Implementierung.

Im dritten Schritt wurden die vorher definierten Ziele weiter verfeinert, um eine präzisere Aufwandseinschätzung abgeben zu können – diese wurde anschließend in Form einer zeitlichen Planung festgesetzt. Des weiteren wurden erste Oberflächendesigns erstellt, um eine ungefähre Vorstellung von der späteren Bedienung zu erhalten.

Die nächste Etappe bestand dann in der Entwicklung eines rudimentären Prototyps zum Testen erster technischer Aspekte – dieser Prototyp wurde dann auch im letzten Schritt vor Beginn der tatsächlichen Implementierung der Gruppe in Form einer Präsentation vorgestellt.

Ab diesem Zeitpunkt verlief die Implementierung vollkommen eigenständig – es wurde lediglich zum 01.10.2008 ein Statusbericht von Dipl.-Wirt.Ing. K. Koochaki eingefordert, um den Fortschritt der Projekte einschätzen zu können.

Die Dokumentation des Projektes wird nun zum 03.11.2008 fällig.

Funktionale Anforderungen

Spielprinzip

Aufgabe innerhalb des Spiel ist es, einen Parcours zu absolvieren und so schnell wie möglich von einem Startpunkt zu einem Zielpunkt zu gelangen – dies kann entweder gegen einen oder mehrere Gegenspieler geschehen, oder aber allein gegen die Zeit, um eine neue Highscore zu erzielen.

Spieloberfläche

Die Spielwelten, die absolviert werden müssen, bestehen aus sich rotierenden Quadern – diese bilden nicht nur die Lauffläche, sondern gleichzeitig die Hindernisse.

Der Spieler kann sicher auf einem einzelnen Quader laufen, von einem Quader fallen oder springen – das Springen oder Fallen lassen von einem Quader kann nicht nur dazu dienen, dem eigentlichen Kursverlauf zu folgen, sondern auch, um zum Beispiel Abkürzungen zu verwenden.

Fällt die Spielfigur unter den untersten Spielstein, scheidet der Spieler so lange aus dem Spielgeschehen aus, bis dieser automatisch auf den zuletzt betretenen Spielstein zurückgesetzt wird.

Der Spieler selbst sieht die Spielwelt aus einer First-Person-Perspektive.

Spielsteuerung

Die Steuerung des Spiels erfolgt sowohl über die Tastatur, als auch über die Maus.

Während über die Maus die Spielansicht rotiert und die Laufrichtung bestimmt wird, werden über die Tastatureingaben die Aktionen der eigenen Spielfigur gesteuert – diese umfassen das Fortbewegen, das seitwärts Laufen, das rückwärts Laufen, das Springen und das Anrempeln anderer Spieler. Die Konfiguration der Tastatur ist frei wählbar.

Menüsteuerung

In Auswahl- und Optionsmenüs dient die Tastatur als Eingabemedium. Durch Drücken der Hoch-/Runter-Pfeiltasten kann ein Menüeintrag selektiert werden und durch das Drücken der Enter-Taste kann der entsprechende Eintrag aktiviert werden.

In einem Optionsmenü kann nach dem Aktivieren eines Eintrags durch Drücken der Links-/Rechts-Pfeiltaste die entsprechende Option geändert und durch Drücken der Enter-Taste die geänderte Option übernommen werden.

Highscore

Zu jeder Spielwelt werden die fünf besten Zeiten zusammen mit einem Namen abgespeichert.

Bei Multiplayer-Spielen zählt nur die Zeit des lokalen Spielers und nicht die, der anderen Spielteilnehmer.

Replayfunktion

Replays werden (sollten sie im Optionsmenü aktiviert sein) während dem Spielverlauf aufgenommen – nach Beendigung des Spiels können diese dann angesehen und für das spätere Betrachten abgespeichert werden.

Ein Replay kann, während es abgespielt wird, pausiert werden und die Kameraposition kann zur besseren Analyse des Spielverlaufs frei bestimmt werden.

Multiplayermodus

Über einen Multiplayermodus können mehrere menschliche Kontrahenten gleichzeitig gegeneinander antreten – jeder Spieler sieht die Spielwelt dabei lediglich aus der Sicht seiner eigenen Position.

Spieler können in der Spielwelt mit anderen Spielern agieren (sie zum Beispiel anrempeln).

Ein Spieler stellt im Multiplayermodus seinen PC als Server bereitet, auf den sich die anderen Spieler verbinden können und der dann für die Verteilung der Spielinformationen an alle anderen Spieler verantwortlich ist – der Spieler, der den Server bereitstellt, ist dafür dann auch in der Lage, zu bestimmen, welches Level gespielt wird und wieviele Spieler zugelassen werden.

Levelеditor

Durch einen Levelеditor ist es möglich, über eine graphische Oberfläche neue Welten zu erschaffen oder bereits vorhandene Welten abzuändern.

Künstliche Intelligenz

Um einen dauerhaften Anreiz zu liefern, werden vom Computer gesteuerte Spielfiguren mit verschiedenen Schwierigkeitsstufen verfügbar sein.

Nicht-funktionale Anforderungen

Bei den nicht-funktionalen Anforderungen handelt es sich um Anforderungen, die nur mittelbar den Funktionsumfang des Spieles betreffen – die also nicht durch zum Beispiel zusätzliche Spieloptionen/-funktionen zu Tage treten.

Die Anforderungen wurden während dem zweiten Milestone erarbeitet und zum dritten Milestone hin weiter detailliert.

Eine selbst gesetzte Hauptanforderung war die einfache Handhabung des Spiels – auch einem ungeübten Spieler sollte es schnell möglich sein, in das Spielprinzip hinein zu finden und erste, einfache Level zu meistern.

Auch die Usability stand weit im Vordergrund – da das Spiel über einen Multiplayer-Modus verfügen sollte, musste besonderer Wert auf die schnell Datenübertragung gelegt werden, sodass alle Spieler schnell über die Aktionen und Positionen der anderen Spieler informiert werden.

Ebenso zählte die Ruckelfreiheit des Spieles zur Usability – diese ist jedoch vor allem abhängig von der eingesetzten Hardware.

Diagramme

Im Laufe der Planung wurden auch einige Diagramme erstellt, um die Konzeption besser visualisieren zu können - diese waren ebenfalls Teil der Bewertung des gesamten Projektes

Use-Case-Diagramm

Das Use-Case-Diagramm war das erste Diagramm, das erstellt wurde, um den Benutzungsablauf der Anwendung darstellen zu können - dabei wurden alle möglichen Aktionen in einem Gesamtsystem dargestellt.

Die Vorgehensweise, wie ein Use-Case-Diagramm zu gestalten ist, war Teil der Software Engineering Vorlesung.

Wie man sehen kann, war es angedacht, die Oberfläche in 3 Bereiche zu teilen - die Konfiguration, den Map-Editor und das Spiel selbst.

Das Spiel sollte zudem in mehrere Stati unterteilt werden - vom Eröffnen des Spiels, über das Spielen selbst, bis hin zum Betrachten des Replays und dem Abspeichern der Highscore.

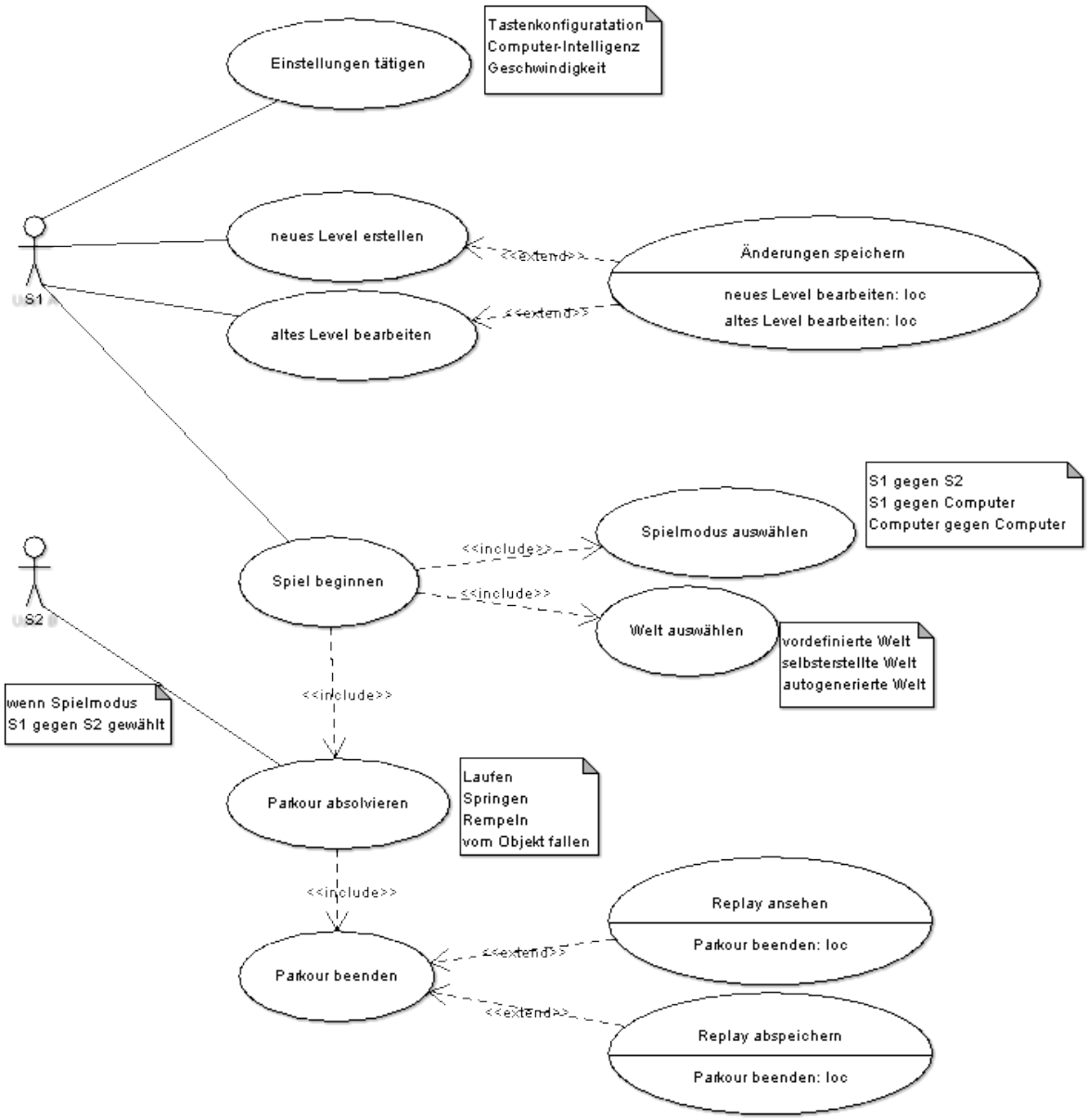

Abbildung I: Use-Case-Diagramm mit Stand vom 06.06.2008

Datenbank-Modell

Auch in diesem Projekt sollte ein Datenbankmodell erstellt werden - aufgrund dessen, dass in einem Echtzeit-Multiplayer-Spiel eine Datenbank jedoch nicht die nötige Performance bietet, um Daten in ihr zwischen zu halten und sie zudem die Flexibilität der Spielumgebung (auf dem PC muss ein Datenbank-Server installiert werden) zu sehr einschränken würde, wurde mit Zustimmung des Dozenten, Dipl.-Wirt.Ing. K. Koochaki, auf eine Datenbank als

Speichermedium verzichtet.

Abbildung II: Datenbank-Modell mit Stand vom 12.06.2008

UML-Diagramm

In dem zu erstellenden UML-Diagramm wurden erste Ideen für die Struktur der späteren Quellcode-Umsetzung visualisiert - diese Struktur wurde im Laufe der Entwicklung immer weiter angepasst, um den eigenen Anforderungen zu entsprechen.

Das Diagramm bot einen guten Ansatz zur Teilung des Gesamtprojektes in Unterprojekte.

Wie erkennbar ist, sollte das Hauptformular der Anwendung der Container des gesamten Spiels werden, in dem alle weiteren Aktionen des Benutzers ablaufen würden.

Um die verschiedenen Aktivitäten besser von einander abgrenzen zu können, sollte in *Paging-Framework* entwickelt werden, in dem jede Aktivität ihren eigenen Rahmen besitzt und nur das aktiv ist, wenn sie auch wirklich gebraucht wird - jede *Page* enthielt dann nur noch die für sie nötigen Objekte.

Diese Vereinfachung der Einzelaktivitäten war im späteren Verlauf der Entwicklung äußerst hilfreich, da Objekte, die einer anderen Aktivität fremd waren, nicht zu Störungen führen konnten.

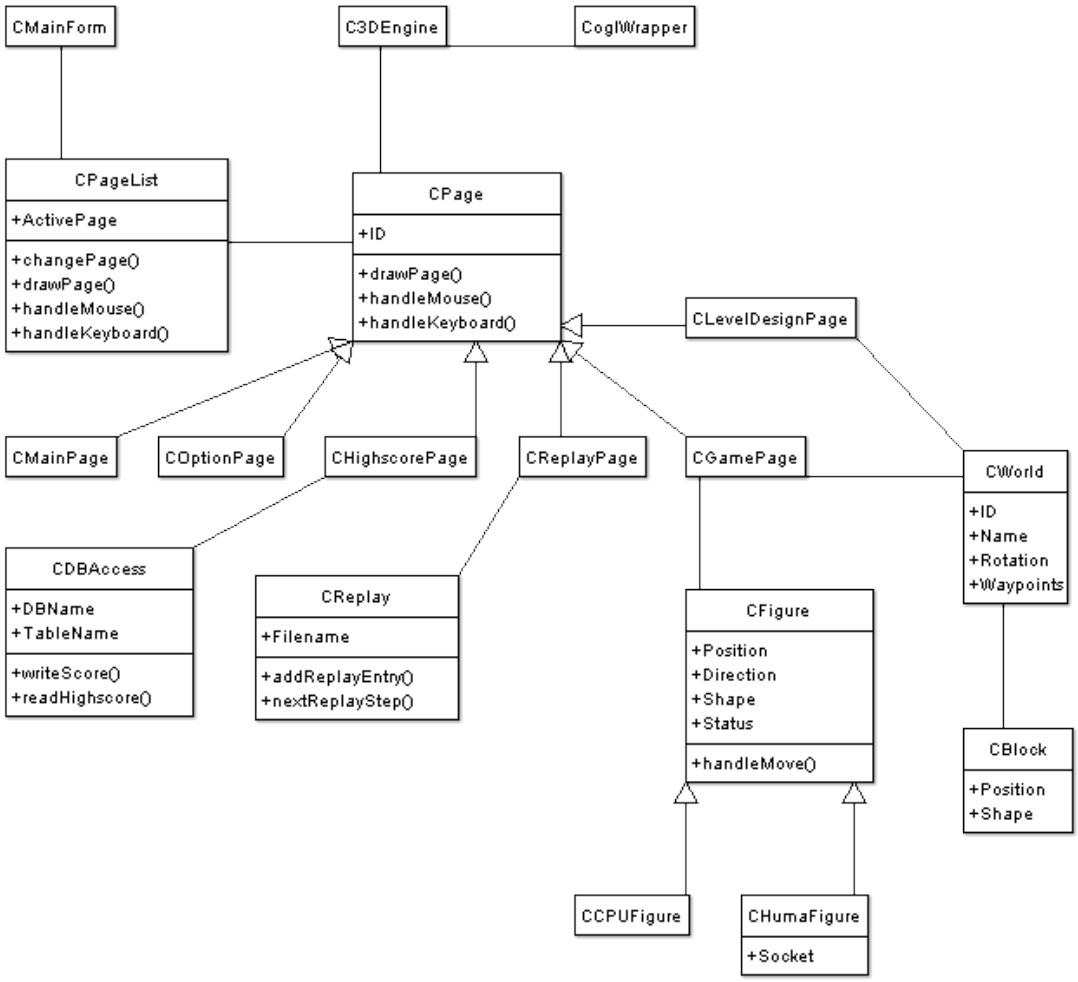

Abbildung III: UML-Diagramm mit Stand vom 13.06.2008

Realisierung

Eingesetzte Technologien

Um einmal einen Einblick in die Fähigkeiten des *.NET Framework* von Microsoft zu erhalten, wurde entschieden, das Projekt in C# zu realisieren. Für das 3D-Rendering für *OpenGL* als Engine ausgewählt. Da C# jedoch ohne OpenGL-Wrapper daher kommt, wird hierfür Collin Faheys *C# wrapper for OpenGL* verwandt – dieser übernimmt selbst keine Arbeit, sondern stellt lediglich die Funktionen von OpenGL unter C# bereit.

Für die Entwicklung der Software wurde auf das *Visual Studio 2005* von Microsoft zurückgegriffen – zum einen, da dieses für seine Qualität bekannt ist und zudem, da es für Studenten der *Berufsakademie Mannheim* im Zuge eines Programms der *Microsoft Developer Network Academic Alliance* (kurz MSDNAA) kostenlos zur Verfügung steht.

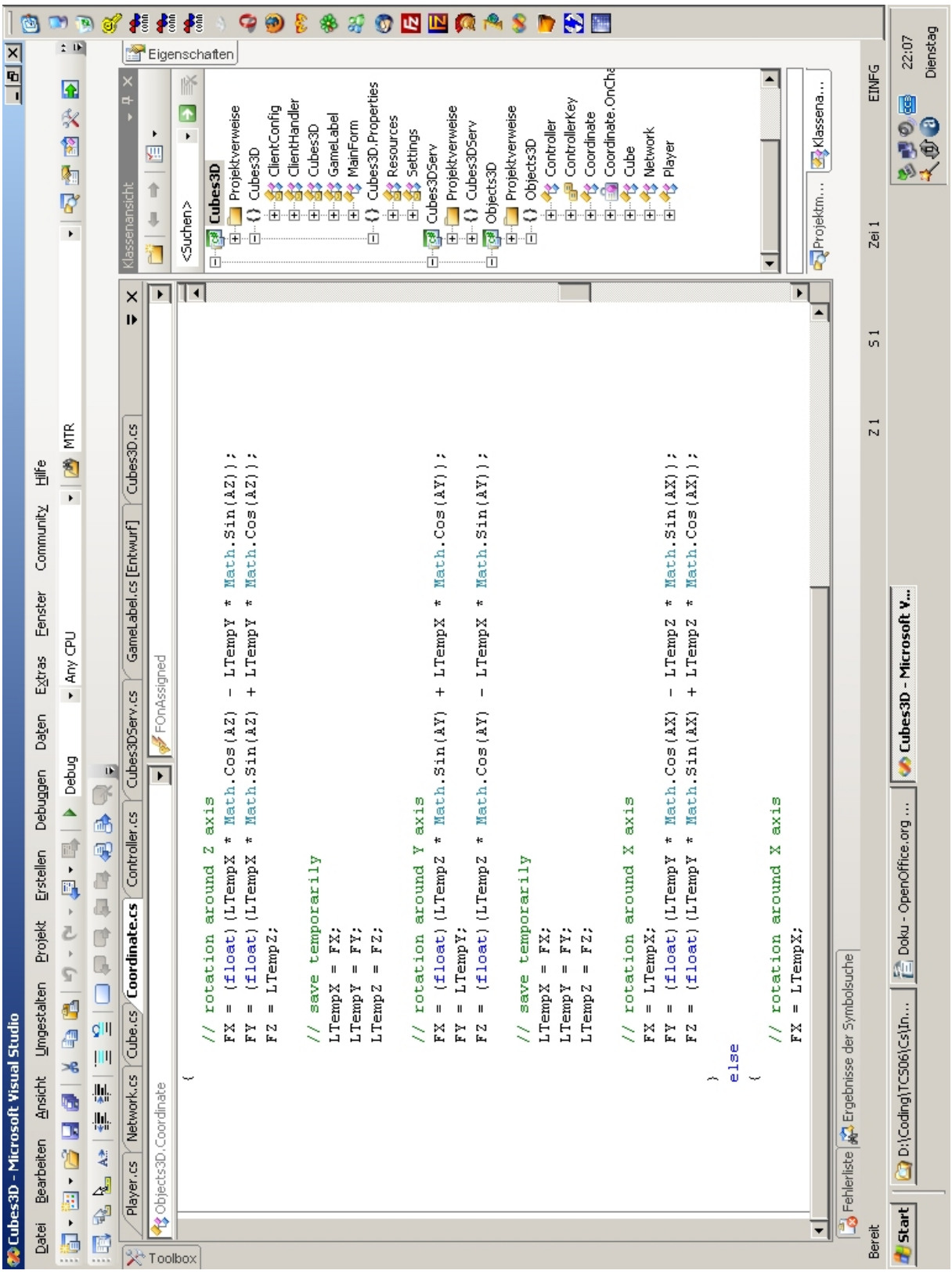

Abbildung IV: Screenshot von Microsoft Visual Studio 2005 (Code)

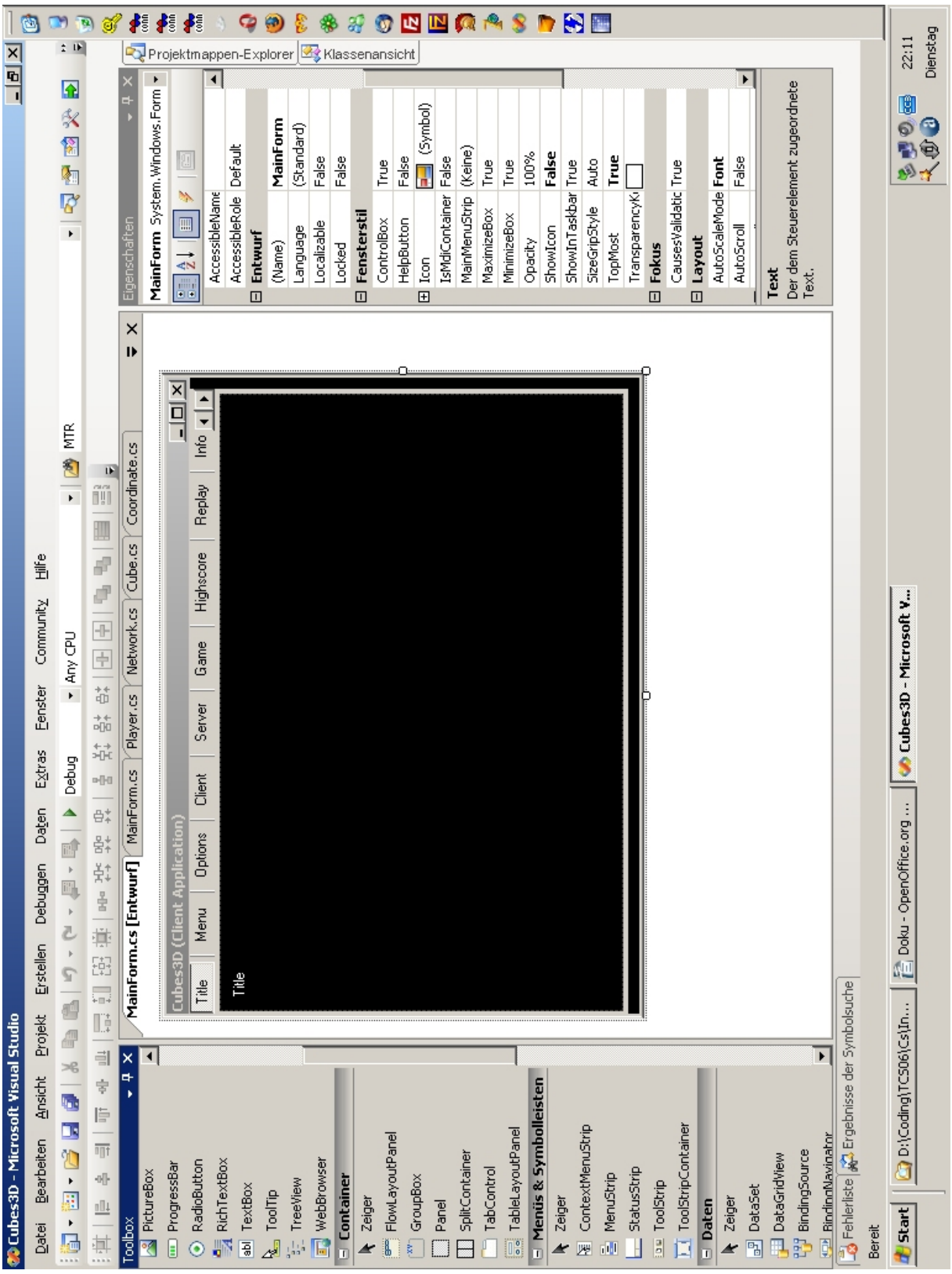

Abbildung V: Screenshot von Microsoft Visual Studio 2005 (GUI)

Implementierungsansätze

Spieloberfläche

Die Spieloberfläche besteht aus einer beliebigen Anzahl an Quadern – diese besitzen einen Mittelpunkt, Rotationswinkel für alle drei Achsen, einen Skalierungswert für alle drei Achsen und einen Skalierungsfaktor.

Anhand dieser Werte können alle Eckpunkte eines Quaders sowohl relativ, als auch absolut bestimmt werden.

Während der Mittelpunkt, die Skalierungswerte und der Skalierungsfaktor statische Werte sind, die in eine Leveldatei ausgelagert werden können, handelt es sich bei den Rotationswinkeln um dynamische Werte, die ständig verändert werden.

Prinzipiell funktioniert das gesamte Spiel durch Ausnutzung der Collision Detection – durch diese wird erreicht – befindet sich ein Spieler auf einem Objekt, sorgt die ständige Kollision mit dem Untergrund dafür, dass der Spieler nicht herunterfallen kann.

Auch die Interaktion mit anderen Spielern basiert auf diesem Prinzip.

Spielsteuerung

Da das .NET Framework eigentlich nicht primär auf das Entwickeln von Spielen, sondern das Erstellen von Plattform-übergreifenden Anwendungen (vor allem im Bereich User-Software) ausgerichtet ist, gibt es bei der Abfrage der aktuellen Tastatureingaben einige Probleme, die anderweitig gelöst werden muss.

Hauptproblem ist, dass pro Tastendruck nur ein *Keydown*-, ein *Keypress*- und ein *Keyup*-Event stattfindet – bei Dingen wie dem ständigen Drücken der Vor-Taste zum vorwärts Laufen, ist dies natürlich denkbar ungeeignet.

Aus diesem Grund wird ein interner Flag-Satz verwandt, der die aktuellen Stati der Tasten dauerhaft zur Verfügung stellt und seine bereitgestellten Werte durch Auswerten der *Keydown*- und *Keyup*-Events aktualisiert – dadurch führt z.B. ein dauerhaftes Drücken der Vor-Taste auch zu einem dauerhaften vorwärts Laufen.

Menüsteuerung

Im Gegensatz zur Spielsteuerung werden für die Menüsteuerung einfach die *Keypress*-Events ausgewertet – dies entspricht eher der gewollten Nutzung des .NET Framework.

Highscore

Zu jedem Level können die besten Gesamtzeiten abgespeichert werden.

Dies geschieht, zur Verringerung des Installationsaufwand nicht in einer Datenbank, sondern in einer Datei.

Replayfunktion

Um das eigene Spiel zu verbessern, werden alle Weltinformationen über das Spiel hinweg in einer Datei mitgeloggt – die Daten in dieser Datei können dann im Nachhinein erneut abgespielt werden.

Auf Grund der benötigten Performance und aufgrund des geringeren Installationsaufwand wird für die Datenhaltung keine Datenbank benutzt.

Multiplayermodus

Um einen Multiplayermodus anbieten zu können, muss die Spiellogik in 2 Teile aufgeteilt werden können – zum einen die Dateneingabe/-ausgabe und zum anderen die Datenberechnung.

Dies entspricht einem klassischen Client-/Server-Prinzip – des weiteren müssen allen Spielern die Positionen aller anderen Spieler bekannt sein – dies könnte zum Beispiel durch eine einheitliche/parallel Datenverteilung an alle Spieler gleichzeitig realisiert werden.

Leveleditor

Bei dem Leveleditor sollte es sich um eine graphische (möglichst 3D) Anwendung handeln, die durch eine intuitive Maussteuerung nutzbar ist – es wäre zu überlegen, ob für die Ansicht evtl. der selbe Quellcode wie für das Anzeigen der eigentlichen Spielwelt verwandt wird.

Künstliche Intelligenz

Die Grundidee war es, die künstliche Intelligenz auf einem Wegpunktesystem basieren zu lassen - es könnte dann eine Schwierigkeitsabstufung dadurch erfolgen, dass für unintelligente KI-Gegner eine Abweichung von diesen Wegpunkten berechnet wird, während bei intelligenten KI-Gegnern diese Abweichungen reduziert werden.

Status

Was wurde bisher umgesetzt?

Spieloberfläche

Die gesamte Basis der Spieloberfläche wurde stark objektorientiert aufgesetzt – alle Zugriffe auf die OpenGL-Schicht wurden durch abstrakte Objekte gekapselt und sind auf diese Weise ohne spezielle Kenntnisse in diesem Bereich nutzbar.

Jedes Objekt besitzt Eigenschaften wie den eigenen, absoluten Mittelpunkt, die aktuellen Rotationswinkel für alle drei Achsen, die Skalierung werte für alle Achsen, sowie einen Skalierungsfaktor - anhand dieser können alle Eckpunkt der einzelnen Objekte errechnet werden.

Das Zeichnen der Objekte erfolgt dann anhand der Eckpunkte – diese werden intern vorgehalten und mit jeder Wertänderung neu berechnet.

Die Eckpunkte bieten gleichzeitig auch die Basis der Collision Detection – diese funktioniert, indem versucht wird, zwischen der aktuellen Position des Spielers und den Flächen der Objekte eine Äquivalenzgleichung aufzustellen; ist die Gleichung erfüllbar, so besteht eine Kollision.

Die Gegner werden bisher noch als Kugeln dargestellt und eine Kollision mit einem Gegner über eine Bounding Box überprüft.

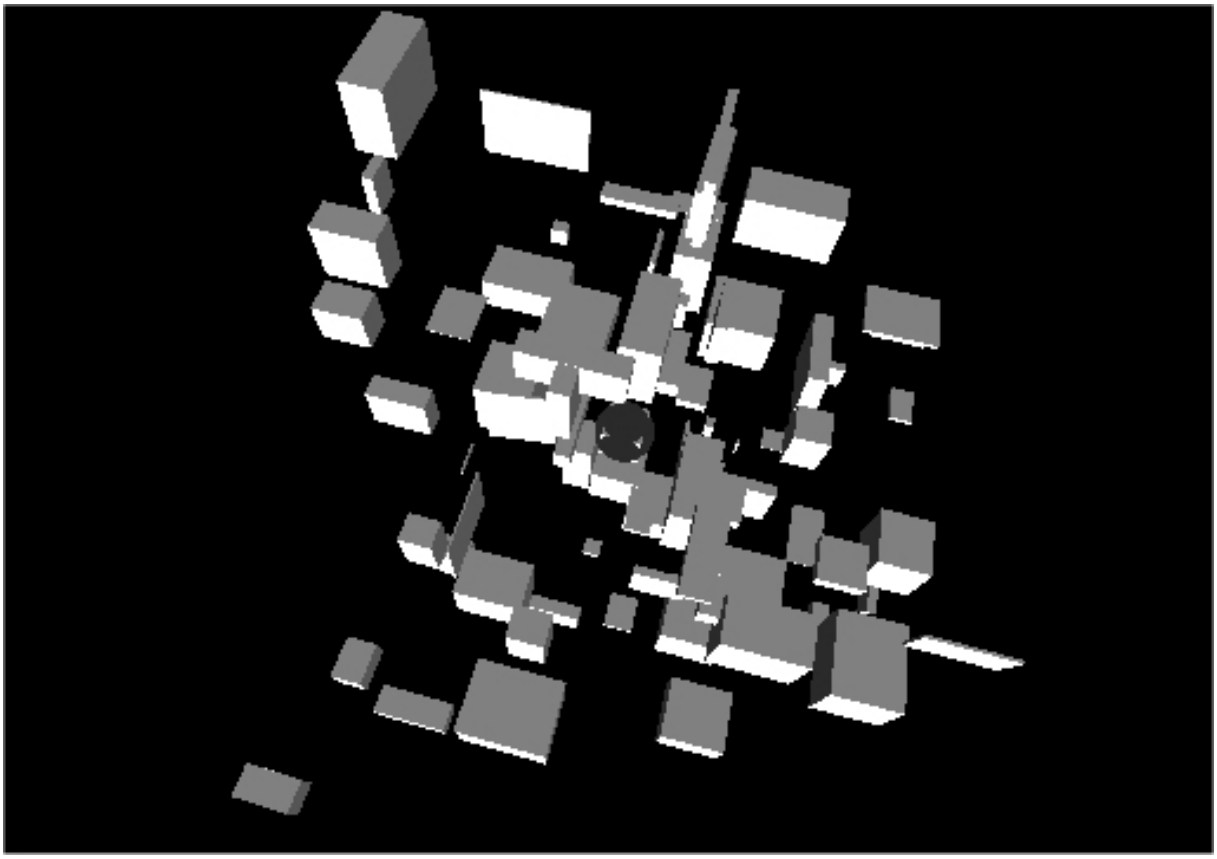

Abbildung VI: In-Game-Screenshot

Spielsteuerung

Die Spielsteuerung ist implementiert – wie geplant, wurde dafür ein Status-Arrays verwandt, das immer die aktuellen Stati der einzelnen Tasten beinhaltet.

Dieses Register wird in regelmäßigen Abständen zum Server geschickt und von diesem ausgewertet – nach der Auswertung wird dann die aktuelle Position des Spielers (die sich durch das Drücken der Tasten und das Einwirken der Kollisionen ändert) an diesen zurück geschickt.

Menüsteuerung

Die Menüsteuerung erfolgt ausschließlich lokal und muss, im Vergleich zur Spielsteuerung an keinen Server geschickt werden.

Highscore

Die Highscore-Funktion und -Ansicht ist fertig.

Die Highscore-Stände werden in einer lokalen Datei abgespeichert und sind so auch dann zugänglich, wenn keine Verbindung zu einem Server besteht.

Bei der Highscore-Liste werden nur die Bestzeiten des lokalen Spielers behandelt – Highscores von anderen Spielern aus Multiplayerspielen werden nicht berücksichtigt.

Replayfunktion

Die Replayfunktion ist ebenfalls einsatzbereit.

Diese basiert darauf, dass alle Informationen, die vom Server während des eigentlichen Spielverlaufs empfangen werden, abgespeichert werden.

Beim Abspielen werden die gespeicherten Nachrichten dann einfach erneut interpretiert und so die zum Zeitpunkt des Spiels aktuellen Informationen angezeigt.

Multiplayermodus

Der Multiplayermodus ist das Herzstück des Spiels, da es ebenfalls die Grundlage des Singleplayermodus ist.

Sowohl beim Single-, als auch beim Multiplayermodus ist die Spiellogik zweigeteilt – während der Client für die Eingabe und Ausgabe der Informationen verantwortlich ist, ist der Server für das berechnen der aktuellen Positionen und Stati zuständig – die Kommunikation zwischen den beiden Parteien erfolgt via UDP-Broadcasts und - Multicasts.

Während diese UDP-Casts bei einem Singleplayer-Spiel auf den lokalen Rechner beschränkt sind, werden sie bei einem Multiplayer-Spiel über das gesamte Local Area Network verteilt.

Der Vorteil der UDP-Casts ist, dass allen Spielern gleichzeitig sämtliche Informationen bereitgestellt werden können – dies verringert den Kommunikationsaufwand und erhöht damit die Spielgeschwindigkeit.

Was wurde bisher nicht umgesetzt?

Levelleditor

Der Levelleditor ist bisher immernoch der, der für die Entwicklung testweise geschrieben wurde – dieser ist bei weitem noch nicht so gut nutzbar, wie es für einen normalen Anwender notwendig wäre.

Alle Einstellungen werden zur Zeit über die Eingabe der einzelnen Parameter eines Objektes getätigt – eine Konfiguration über die Maus ist bisher noch nicht möglich.

Jedoch ist es bereits möglich, korrekt Map-Dateien zu erzeugen.

Für die weitere Entwicklung ist geplant, den Levelleditor mit der Grafikfunktion des eigentlichen Spiels auszustatten.

Hauptproblem ist es, eine praktische 3D-Anwendung mit einer ordentlichen Benutzersteuerung zu versehen – die Maussteuerung erweist sich leider innerhalb der 3D-Ansicht als sehr unpräzise, da mit dem zweidimensionalen Eingabegerät „Maus“ Einstellungen innerhalb einer dreidimensionalen Umgebung getätigt werden müssen.

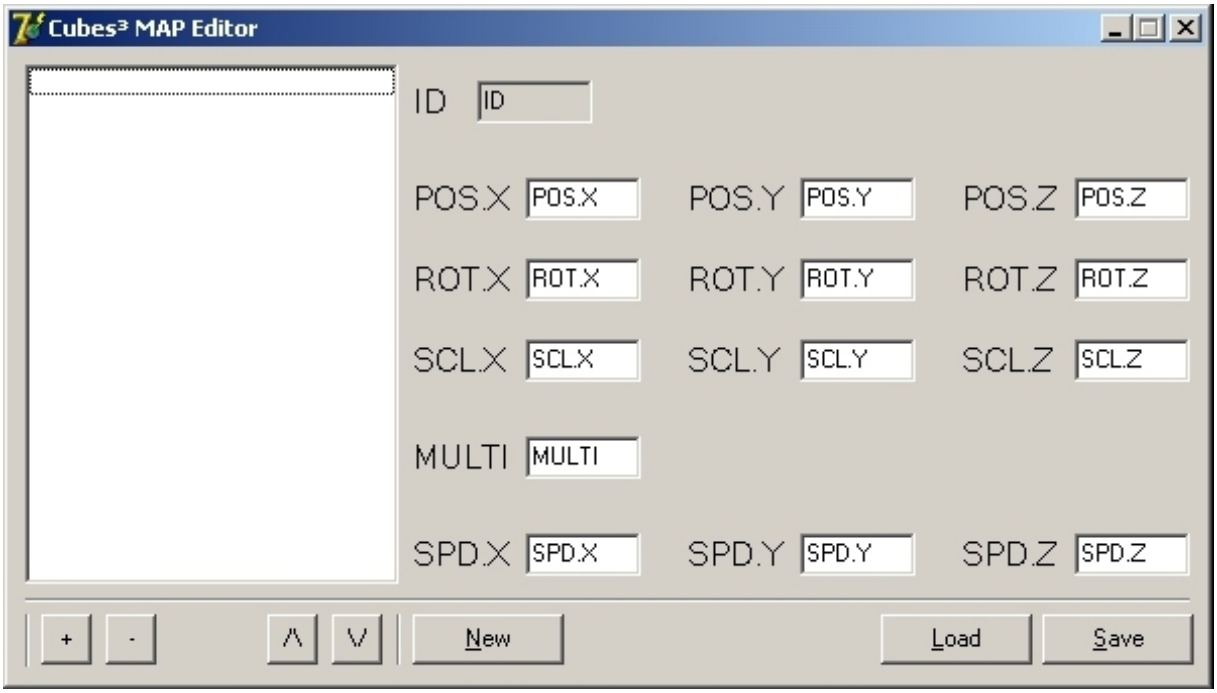

Abbildung VII: Screenshot des testweisen Map-Editors

Künstliche Intelligenz

Die künstliche Intelligenz stellt das größte Problem in der Entwicklung dar.

Das angedachte Wegpunktesystem hätte zwar für die Variante des Spiels funktioniert, in der alle Blöcke einen durchgängigen Parcours gebildet hätte, scheitert jedoch kläglich bei der Verwendung einzelner, separat rotierender Objekte.

Es ist mit enormen rechnerischen Anstrengungen verbunden, die nächste Aktion der künstlichen Intelligenz zu berechnen, da sehr viele Sprünge im Spiel notwendig sind, die genau positioniert und zeitlich geplant werden müssen – während einem Sprung kann auf die weitergehende Rotation des Zielobjektes nicht mehr reagiert werden.

Die Idee hinter der künstlichen Intelligenz ist zur Zeit, dass sie sich an der Nummerierung der Steine orientiert und versucht, von Stein **n** auf dem kürzesten Weg zu Stein **n+1** zu gelangen - bevor der Versuch unternommen wird, zum nächsten Stein zu gelangen, versucht die künstliche Intelligenz sich genau in der Mitte des aktuellen Steins zu positionieren und dann erst die nächsten Schritte durchzuführen.

Generell agiert die künstliche Intelligenz wie ein normaler Spieler – er sendet seine Aktionen via UDP (Abkürzung für „User Datagram Protocol“) an den Server, wertet die Positions- und Status-Meldungen des Servers jedoch nicht aus.

Dies liegt daran, dass die künstliche Intelligenz als einzelne Threads innerhalb des Servers läuft und dadurch direkt Zugriff auf die Spielwelt-Daten erhält – dies ist notwendig, um die künstliche Intelligenz mit zusätzlichen Informationen, wie der Steinnummerierung zu versorgen.

Es scheint, als ob für die Programmierung einer guten KI zur Zeit nicht die nötigen Fähigkeiten zur Verfügung stehen – des weiteren erwies sich die Idee mit dem Wegpunktesystem als nicht umsetzbar.

Diagramme

Um einen Vergleich zwischen Planung und Umsetzung zu haben, wurden nochmals UML-Diagramme erstellt, die die Beziehungen zwischen den einzelnen Klassen repräsentieren sollen - wie zu erkennen ist, ergibt sich ein völlig anderes Bild, als die Planung vermuten ließ.

Der Schwerpunkt liegt eindeutig auf der Strukturierung der Daten als dem Aufbau der Oberfläche - evtl. auch deshalb, da für die Oberfläche Standardklassen verwandt wurden, die im UML-Diagramm nicht auftreten.

Bei dem dem ersten Diagramm handelt es sich um das Paket *Objects3D* - dieses enthält alle wichtigen Klassen, die sowohl für die Client-, als auch für die Server-Funktion wichtig sind.

Alle Informationen der Klassen von *Objects3D* können und werden via UDP zwischen den beiden beteiligten Programmen ausgetauscht werden (mit Ausnahme der Klasse *Network*, die für die Aufbereitung der Daten zuständig ist).

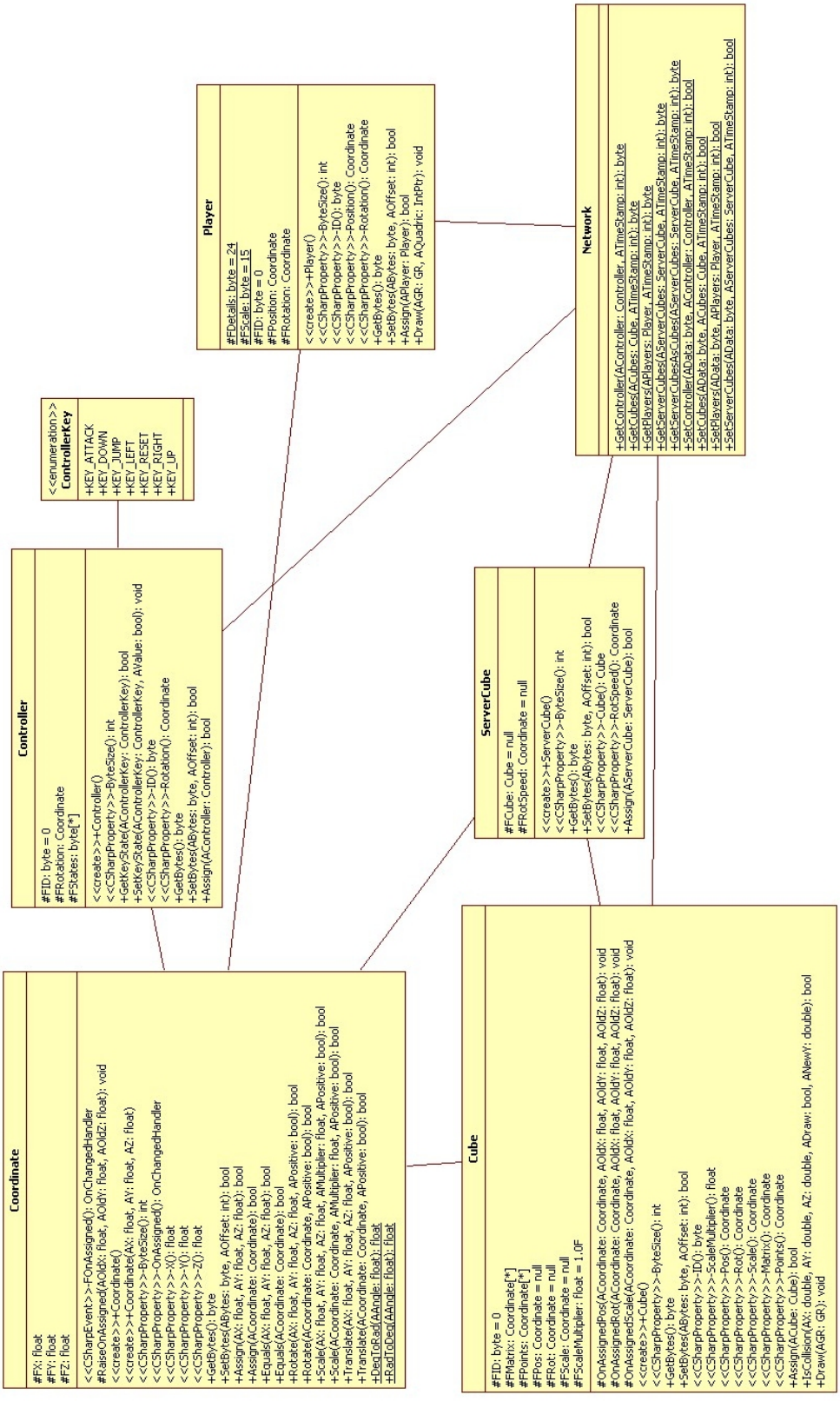

Abbildung VIII: UML-Diagramm des Paket Objects3D

Im zweiten Diagramm ist der Aufbau des Clients, in Form des Paket *Cubes3D,* zu sehen - in diesem sind alle Spiel-relevanten Teile in den Klassen *ClientHandler und ServerHandler* konzentriert - diese Klassen haben wiederum mehrere Verbindungen zu Klassen aus dem Paket *Objects3D* .

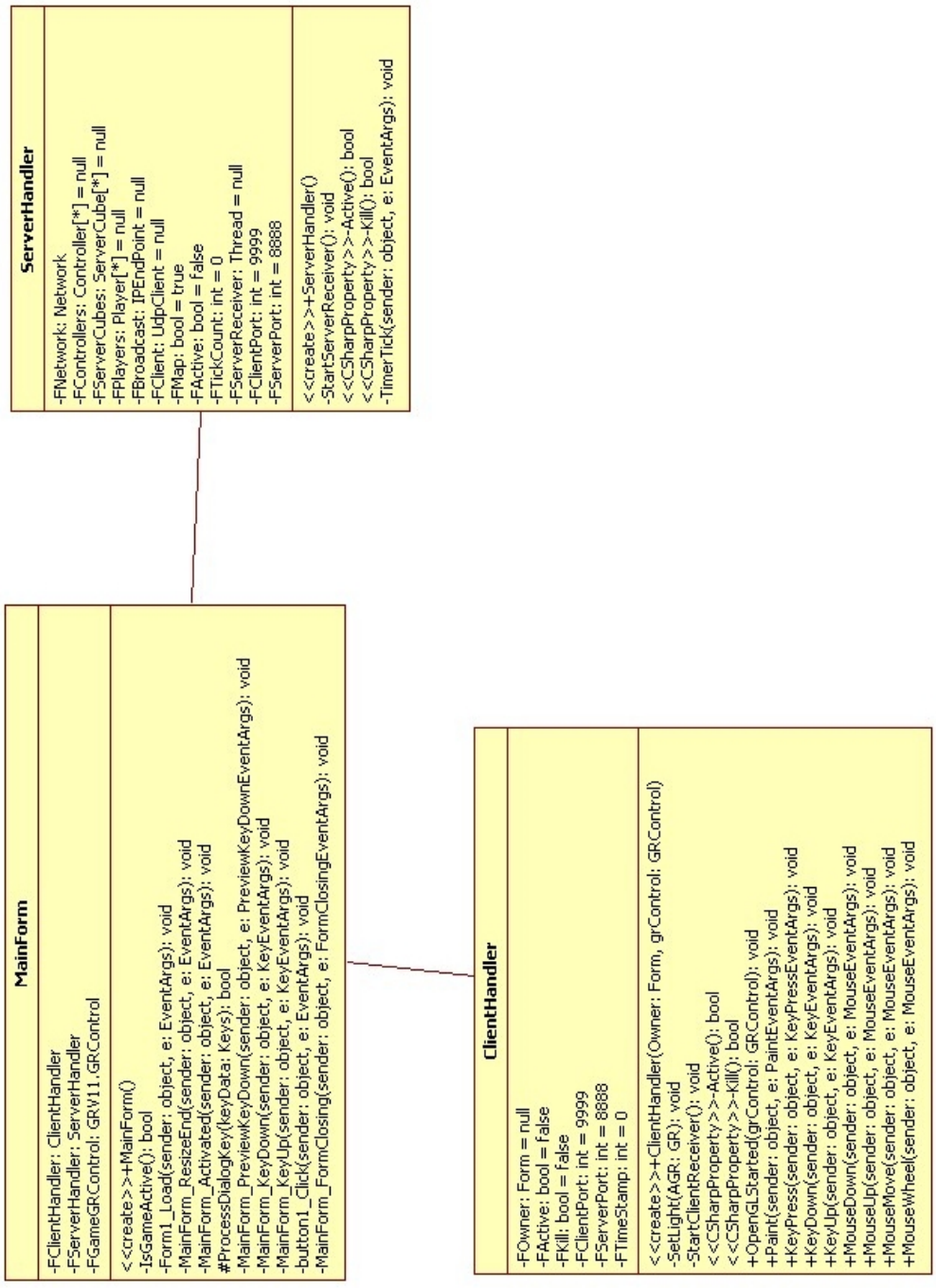

Abbildung IX: UML-Diagramm des Paket Cubes3D

Einsatz

Systemanforderung

Die Software wurde unter *Windows XP* mit installiertem *Service Pack 3* entwickelt und bisher auch nur unter dieser Konfiguration getestet.

Der PC sollte mit *mindestens 512MB RAM* ausgestattet sein und es sollte das *.NET Framework 2.0* installiert sein - für einen reibungslosen Spielablauf sollte zudem eine Grafikkarte mit 3D-Beschleuniger und OpenGL-Unterstützung in dem Computer verbaut sein.

Um ein Multiplayerspiel zu ermöglichen, muss der Computer über eine eingebaute Netzwerkkarte verfügen – das Spielen von Multiplayerspielen ist nur innerhalb eines Local Area Network möglich; Spiele über das Internet werden nicht unterstützt.

Installation

Soweit dies noch nicht geschehen ist, installieren Sie bitte die für Ihr System passende Version des Microsoft .NET Framework 2.0 – gehen Sie dabei bitte nach den Anweisungen vor, die mit dieser Software mitgeliefert wird.

Kopieren Sie anschließend alle Dateien des Spiels in einen Ordner Ihrer Wahl.

Um das Spiel zu Spielen, müssen Sie nun nur noch die Datei ***Cubes3D.exe*** ausführen.

Deinstallation

Um das Spiel zu deinstallieren, müssen sie lediglich den Ordner löschen, in den Sie die Dateien des Spiels kopiert haben – das Spiel schreibt sich weder in die Registry, noch werden Daten in Systemverzeichnisse kopiert.

Anschließend können Sie (sollte es für keine weiteren Anwendungen benötigt werden) ebenfalls das Microsoft .NET Framework 2.0 von Ihrem Rechner deinstallieren – gehen Sie dabei nach der Anleitung dieser Software vor.

Bedienung

Singleplayer

Um ein kurzes Singleplayer-Spiel zu starten, wählen Sie im Auswahlmenü „Singleplayer“ aus.

Geben Sie im nächsten Schritt die gewünschten Optionen für das Spiel ein – die gewünschte Weltkarte, die Anzahl und Stärke der künstlichen Intelligenz *(bisher noch nicht implementiert)* und gehen Sie dann auf „Fertig“ - das Spiel beginnt dann in wenigen Sekunden

Multiplayer

Um ein Multiplayer-Spiel zu starten, wählen Sie im Auswahlmenü „Multiplayer“ aus.

Geben Sie im nächsten Schritt an, ob Sie sich einem bestehenden Spiel anschließen wollen, oder selbst ein Spiel eröffnen wollen.

Wenn Sie ein Spiel eröffnen wollen, geben Sie bitte anschließend die gewünschten Optionen für das Spiel ein – die gewünschte Weltkarte, die Anzahl der maximal teilnehmenden Spieler, die Wartezeit in Sekunden, sowie ein eventuelles Passwort und gehen Sie dann auf „Fertig“ -

das Spiel beginnt dann, wenn die maximale Anzahl an Spielern dem Spiel beigetreten ist, oder wenn die definierte Wartezeit abgelaufen ist.

Fazit

Grundsätzlich möchte ich mich erst einmal bei unserem Dozenten, Dipl.-Wirt.Ing. K. Koochaki, dafür bedanken, dass er uns die Möglichkeit gegeben hat, einmal in die Welt der Spielentwicklung einzutauchen.

Auch möchte ich mich dafür bedanken, dass er es erlaubt hat, eigene Erfahrungen im Umgang mit 3D-Engines zu machen, ohne zwangsläufig auf professionelle Tools wie einem Game Maker oder ähnliches zurück zu greifen.

Die Erfahrungen, die bei der eigenhändigen Vektorrechnung usw. gemacht wurden, werden später sicherlich wertvoller sein, als zu wissen, wie man in Programm X diese und jene Spielaktion erzeugt – alles komplett selbst zu entwerfen und umzusetzen war eine ganz eigene Erfahrung.

Es war nicht immer leicht, den Frust zu ertragen, wenn mal wieder irgendetwas nicht so lief, wie es eigentlich geplant war – wenn zum Beispiel. die Vertex-Reihenfolge unter OpenGL nicht stimmte und somit Objekte nicht korrekt gezeichnet wurden, oder das gesamte System zu Bruch ging, weil irgendeine wilde Referenz den Debugger in die Knie zwang.

Doch der stetig sichtbare Fortschritt war der Lohn der ganzen harten Arbeit.

Ich denke, dies gilt vor allem für die zwei Gruppen, die sich auch selber um die Grafik-Engine gekümmert haben – es war ein riesiger Aufwand, die gegebene Basis soweit zu abstrahieren, um damit in der alltäglichen Programmierung zurecht zu kommen.

Aber es hat geklappt – darüber bin ich sehr stolz und denke, dass ich dies auch ruhigen Gewissens sein kann.