Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

Wickie
Dokumentation

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

Story

Zur Geschichte braucht eigentlich nicht viel gesagt werden, denn wer kenn den kleinen Wickie nicht? Er segelt mit seinem Vater und den Wikingern durch die Weltmeere und muss dabei mit seinen genialen Ideen seinem Vater immer wieder aus kniffligen Situationen helfen. In diesem Spiel wollen wir seine Ideen und sein Köpfchen aber mal bei Seite lassen und schauen ob er schon ein richtiger Wikinger geworden ist und auch kämpfen gelernt hat. Dazu muss er in Griechenland eine schwierige Aufgabe erledigen: Sein Vater und die Wikinger wurden von den Griechen in ein Labyrinth gelockt. Sollte es ihnen gelingen zur anderen Seite zu gelangen, dürfen sie den Schatz der dort auf sie wartet behalten. Doch wie zu erwarten war, finden die Wikinger nicht mehr aus dem Labyrinth. Nun liegt es also wieder an Wickie einen Weg durch das Labyrinth zu finden, den Schatz zu holen und nach Flake zurückzukehren. Doch im Gegensatz zum Film muss Wickie im Spiel nicht nur seinen Kopf sondern auch seinen Bogen gebrauchen, da im Labyrinth eine Menge Wölfe auf ihn lauern. Und natürlich muss er erstmal einen Weg durch das Labyrinth finden. Aber unserem Wickie wird schon was einfallen…

Kurzbeschreibung

Das Ziel des Spiels ist es sich mit Wickie einen Weg durch das Labyrinth zu suchen und dabei den Wölfen aus dem Weg zu gehen, oder sie mit Pfeil und Bogen zu bekämpfen, je nachdem wie viele Lebenspunkte Wickie noch hat und wie gut man im Umgang mit Pfeil und Bogen ist. Auf der anderen Seite des Labyrinths wartet der Schatz, den es zu holen gilt. Um nicht die Orientierung zu verlieren kann der kleine Wickie Pfeile auf den Boden legen um so zu markieren wo er schon entlang gegangen ist.

„Installationsanleitung“

Einfach die Datei wickie.exe im „bin“ Ordner starten. Wenn man den Source Code übersetzten will und dann gleich aus Visual Studio starten will darf man nicht vergessen die zwei .dll Datein und die Ordner „models“ und „sounds“ vom „bin“ Ordner in den „src“ Ordner zu kopieren.

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

Weitere Gliederung

Der Rest der Dokumentation ist wie folgt gegliedert. Zuerst folgen nun die Beschreibungen der Anforderungen für die 3. Abgabe, danach kommt die Dokumentation die bereits bei der 2. Abgabe abgegeben wurde. Der Grund warum ich das hier noch mal anhänge ist, dass ich bei der 2. Abgabe schon einiges (Spieldesign, Steuerung, Kameramodi, komplexe Objekte, Loader, Collision Detection,…) realisiert hatte, das erst bei der 3. Abgabe verlangt war. Um nicht noch mal alles umschreiben zu müssen, bei den meisten Teilen hat sich da sowieso nichts mehr geändert, und um euch Tutoren ein hin- und herwechseln zwischen den verschiedenen .pdf Dateien zu ersparen ist das gleich hinten drangehängt.

3. Abgabe

Spieldesign

Das Spieldesign wurde von mir Großteils schon in der 2. Abgabe realisiert. Hinzugekommen ist die „Indiana Jones Lava-/Feuerkugel“ sowie drei verschiedene Levels und ein Menü zur Auswahl eines Levels. Weiters natürlich die verschiedenen Display und Textureinstellungen wie unten beschrieben. Weitere Sounds wurden verwendet und kommen vor allem bei der Feuerkugel zum Einsatz. Zum Spieldesign gibt’s sonst eigentlich nicht mehr viel zu sagen, falls ihr euch nicht mehr erinnern könnt, einfach unten in der Doku der 2. Abgabe nachschauen.

Nichttriviale Objekte

Auch das wurde von mir bereits in Abschnitt 2 erledigt. Einfach bei „Datenstruktur und Loader“ sowie „Modelle und Texturierung“ in der Doku der 2. Abgabe nachlesen (siehe unten).

View Frustum Culling

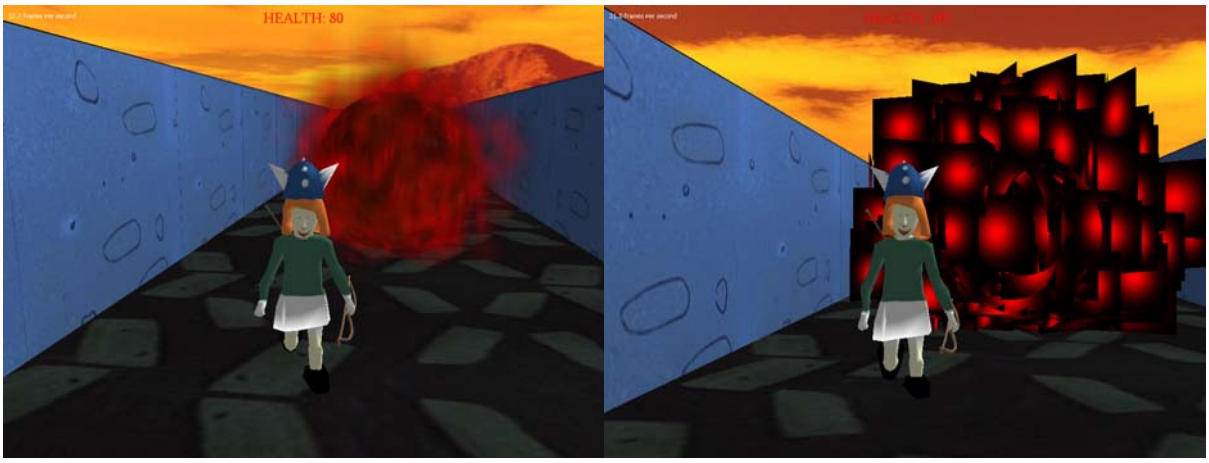
View Frustum Culling wurde von mir mittels Bounding Spheres durchgeführt. Super erklärt hab ich die Info dazu in folgendem Tutorial gefunden:

http://www.lighthouse3d.com/opengl/viewfrustum/index.php?clipspace

Transparenz – Effekte

Transparenz wurde von mir bei der die „Indiana Jones Lava-/Feuerkugel“ verwendet. Dabei wurde beim Partikelsystem (siehe unten) der Alpha Wert der einzelnen Partikel schrittweise verringert, bis bei 0 angelangt und dann stirbt das Partikel. Um nicht die Quadrate zu sehen sonder eben etwas das wie ein Partikel aussieht wurde von mir die Textur eingelesen und der R Kanal als Maske verwendet, der angibt wie durchsichtig/undurchsichtig das Partikel an dieser Stelle ist. Das war sehr einfach, da meine Partikeltextur ja nur Graustufen enthält und da bei Graustufen (inkl. Weiß und Schwarz) RGB Werte immer gleich vertreten sind, konnte ich mir einen Kanal aussuchen. Der R Kanal gibt also an, dass das Partikel, besser gesagt seine Textur, am Rand durchsichtig ist und in der Mitte undurchsichtig. So bekommt man einen ganz passablen Verlauf. Man sieht das auch sehr gut auf den Bildern wenn man die Transparenz ausschaltet.

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at



Experimentieren mit OpenGL

Nachfolgend wird beschrieben was ich alles implementiert habe, gleich davor auch die Taste mit der umgeschaltet werden kann.

F2 Framerate

Aus- und Einschalten der Framerate-Anzeige.

F3 Wireframe

Aus- und Einschalten des Wireframe Modus.

F4 Texturqualität und Mip Mapping

Ich hab mich dazu entschieden beides gleich mit einer Taste zu belegen, da das ja eigentlich sehr stark zusammenhängt, schließlich basiert ja trilineares Filtering auf Mipmaps. Es wurden alle 6 verschiedenen Texturmöglichkeiten implementiert:

- GL_NEAREST (point sampling)
- GL_LINEAR (bilineares filtering)
- GL_NEAREST_MIPMAP_NEAREST (point sampling mit mip map)
- GL_LINEAR_MIPMAP_NEAREST (bilinear mit mip map)
- GL_NEAREST_MIPMAP_LINEAR (point sampling mit mipmap interpolation)
- GL_LINEAR_MIPMAP_LINEAR (trilineares filtering)

F6 Geometrie Modi

Es werden drei verschiedene Modi unterstützt: Immediate Mode, Vertex Arrays und auch ARB_vertex_buffer_objects. Da meine Charaktere (Wickie, Wölfe) aufgrund der Animationen in jedem Frame unterschiedliche Vertex-Positionen haben, macht es natürlich wenig Sinn für diese dynamischen Objekte ARB_vertex_buffer_objects zu verwenden, denn was nützt es mir wenn das auf der Grafikkarte gespeichert wird, ich jedoch jedes Frame alle Informationen updaten/überschreiben muss. Daher hab ich mich entschieden diese Geometrie Modi, sowie die Display Lists nur für meine statischen Wände, also mein Labyrinth zu implementieren. Insgesamt gibt es also 6 Kombinationen: 3 Geometrie Modi, jeder davon mit oder ohne Displaylist.

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

F7 Display Lists

Wie oben beschrieben für das Labyrinth implementiert in Kombination mit den verschiedenen Geometrie Modi. Bringt natürlich für die relativ einfachen Wände nicht allzu viel, aber es geht ja mehr darum jedes dieser Konzepte mal kennen zu lernen und selbst zu programmieren.

F8 View Frustum Culling

Aus- und Einschalten des View Frustum Cullings.

F9 Transparenz

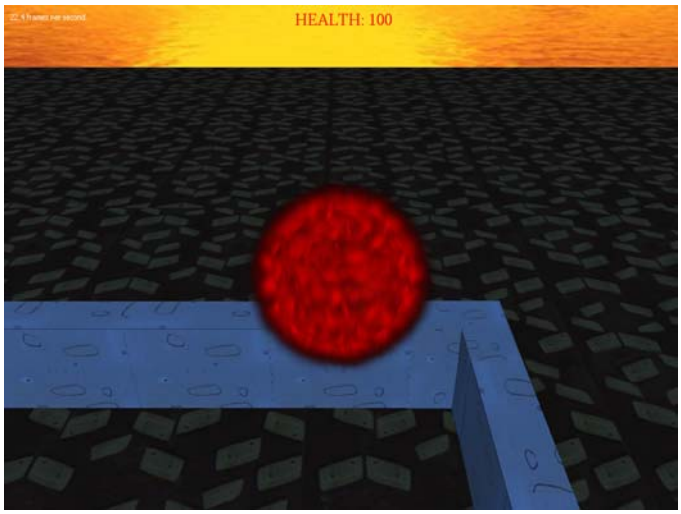
Aus- und Einschalten der Transparenz-Effekte.

Spezialeffekte

Ich habe folgende Spezialeffekte implementiert:

- Feuer
- Explosion

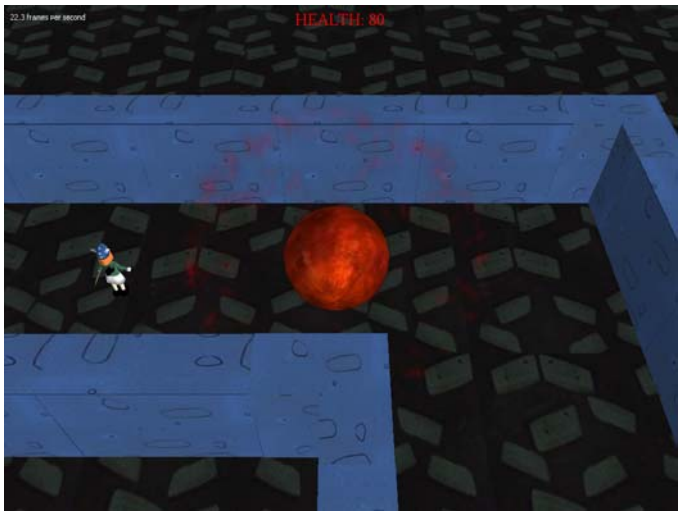
Beides kommt bei meiner „Indiana Jones Lava-/Feuerkugel“ zum Einsatz. Dabei handelt es sich um eine rollende Lava/Feuerkugel, die Wickie verfolgt. Den Namen hat sie klarerweise von Indiana Jones Teil 1, wo eine Steinkugel hinter Indiana Jones herrollt.



Es treten folgende Effekte auf: Zuerst schlägt die Kugel Flammen. Das passiert am Beginn der Animation wenn die Kugel zu Boden fällt und wird mit einem „Wusch“ Sound unterlegt. Beim Aufprall der Kugel hört man einen dumpfen Knall, wodurch auf ganz einfache Weise vermittelt wird, dass es sich dabei um eine massive Kugel handelt. Ich hab am Anfang gar nicht gedacht, dass dieser kurze Sound so eine große Wirkung hat, aber man glaubt dadurch wirklich dass das eine „bedrohliche“, massive Kugel ist. Kleiner Sound → Große Wirkung. Dann im Rollen hört man ein Rollgeräusch und die Kugel schlägt immer wieder Flammen. Zusätzlich wird der Indiana Jones Sound eingespielt, der das ganze noch untermauert. Sobald die Kugel auf eine Wand trifft kommt es zur Explosion, natürlich auch mit Explosions-Sound. Die Kugel wird im Rollen und Fallen gleichmäßig beschleunigt. Dabei wird die Position (bzw. der Weg s) und mit den physikalischen Gleichungen ($v = a * t$ und $s = v * t$) berechnet.

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

Realisiert wurde das ganze mit einem sehr kleinen einfachen Partikelsystem. Dabei starten die Partikel innerhalb der Kugel mit einer zufälligen Rotation und nehmen beim Flug nach außen durch die Kugel immer mehr an Transparenz zu. Die Partikel werden also mit Hilfe von Alpha Blending und einfachen Billboards simuliert, siehe Bild oben. Für die Explosion werden alle Partikel annähernd gleich gestartet und die Kugel zum richtigen Zeitpunkt ausgeblendet.



Noch ein kurzer Hinweis, dass man die Indiana Jones Kugel auch sieht: Einfach nach ganz hinten rechts laufen im Level 1, ist eh nicht zu verfehlen, so groß ist ja das Labyrinth im Level 1 noch nicht.

Abschließende Bemerkungen 3. Abgabe

Ich hätte gerne noch weitere Effekte eingebaut und auch noch länger an dem Spiel gearbeitet, leider geht sich das aber zeitlich nicht mehr aus, da ich mir jetzt Gedanken über mein Diss Thema machen muss. Naja, hat auf jeden Fall riesig Spaß gemacht und ich glaub mein Spiel kann sich für ein Ein-Mann Projekt schon sehen lassen. Ein paar originelle Ideen sind auch drin, was will man mehr. *gg* Wie gesagt, war zwar ein sehr großer Zeitaufwand, aber mit Abstand die beste LVA, die ich bis jetzt an der TU besucht hab.

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

2. Abgabe

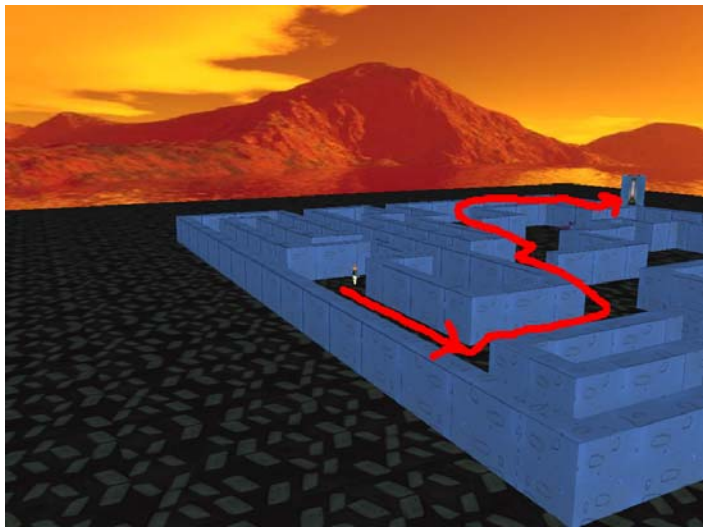
Steuerung

Tasten:	A	-	Links
	W	-	Vorwärts
	S	-	Rückwärts
	D	-	Rechts

Maus:	Linker Mausbutton:	Schiessen
	Rechter Mausbutton:	Pfeil auf den Boden legen
	Mausrad:	Zoomen bzw. Wechsel zwischen First Person und Third Person Modus

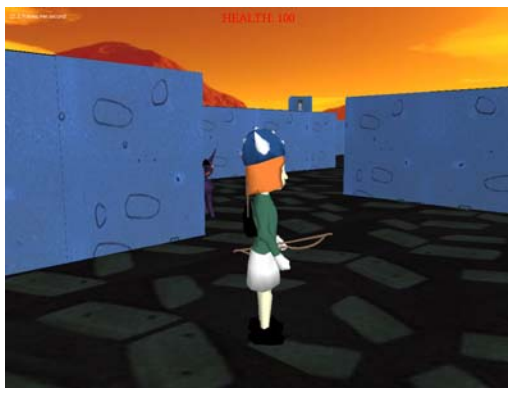
Walkthrough

Ist nicht so schwierig, einfach die Wölfe abschießen und alle Gänge des Labyrinths ausprobieren. Einfach überall Pfeile ablegen wo man schon hingelaufen ist. Die Wölfe aus der Entfernung abschießen, sie können nicht um die Ecke laufen. Außerdem sieht man beim Starten des Spiels in der Kamerafahrt schon ungefähr wie das Labyrinth aussieht und wo sich das Ziel befindet. Sollte also zu schaffen sein.

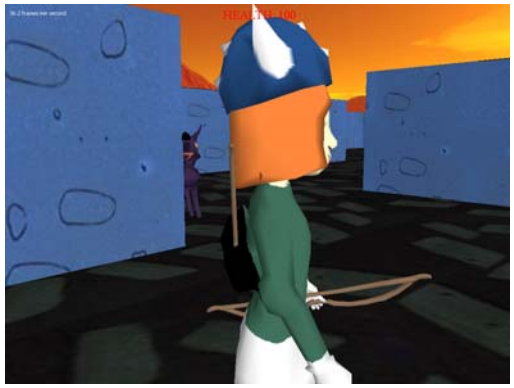


Kamera und View Modi

Das Kameramodell ist der Teil auf den ich besonders stolz bin. Angelehnt an Spiele wie Prince of Persia: Sands of Time und Splinter Cell,… wurde mein Kameramodell wie folgt entworfen:



Es gibt in meinem Spiel sowohl „Third Person“ als auch „First Person“.



Gestartet wird im „Third Person“ Modus, dabei kann man sich mit der Maus um Wickie drehen und in Abhängigkeit der Momentanen Blickrichtung mit den Tasten AWSD bewegen. Mit dem Mausrad kann man rein und raus

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

zoomen. Zoomt man soweit rein, dass man den maximalen Zoomstatus erreicht, dann findet ein Wechsel in den „First Person“ Modus statt. Zoomt man wieder raus, dann wechselt man wieder in die „Third Person“. Zur besseren Verdeutlichung was ich meine sieht man rechts drei Screenshots, alle vom selben Kamera Blickwinkel aus, aber mit unterschiedlichem Zoom. Beim maximalem Zoom (siehe letztes Bild) wechselt man in den „First Person“ Modus.

Im „First Person“ Modus wird ein Fadenkreuz angezeigt und man kann in beliebige Richtungen schießen. Im „Third Person“ schießt man nicht dahin wo die Kamera hinschaut, sondern in die Richtung in die Wickie blickt und zwar immer horizontal. Die Schusshöhe wurde so festgelegt, dass man die Wölfe treffen kann, wenn man im „Third Person“ Modus schießt.

Der Grund warum ich beides implementiert habe war erstens das Interesse daran wie man so was realisieren kann und zweitens wollte ich nicht die einfache Variante des First Person gehen, denn wozu hätte ich dann meinen Wickie so schön modelliert wenn ihn dann niemand sieht. Macht doch viel mehr Spaß wenn man nicht nur den Rücken des Hauptcharakters sieht, sondern ihn in allen möglichen Richtungen vor sich herlaufen sieht und auch seine Animationen beobachten kann, so wie bei Prince of Persia eben.

Datenstrukturen und Loader

Loader:
Meine Objekte (Wickie, Mauern,…) sind .md2 Objekte (zur Modellierung unten mehr) und werden mit einem MD2 Loader ins Spiel gebracht. Der Loader wurde vom md2 Tutorial aus den GameTutorials (http://www.gametutorials.com/) genommen und adaptiert. Dabei wurde von mir alles in Klassen gepackt, Normalvektoren generiert,…

Datenstrukturen:
Da der Loader schon einige Strukturen verwendet hat, habe ich den Großteil unverändert gelassen, um nicht den ganzen Loader umschreiben zu müssen. Die Draw Methode wurde zwar vom Loader abgekuckt, schließlich musste ich mich ja mit den Datenstrukturen auseinandersetzen, aber nicht abgeschrieben.

Die Vertices und Texturkoordinaten werden als Array gespeichert und die Faces haben dann nur mehr Indizes in diese Felder. Es werden Vertex Normalvektoren generiert und auch Face Normalvektoren, je nach Objekt (Krümmung wie bei Wickie bzw. keine Krümmung wie bei meinen Wänden) werden dann entweder Vertex Normalvektoren oder Face Normalvektoren verwendet. Alle meine Objekte sind texturiert, die Texturkoordinaten werden ebenfalls über .md2 eingelesen. Um die Texturen zu laden wurde für BMP Dateien zu Beginn der Ansatz von der CG2 Homepage verwendet, dann wurde auch eine JPEG Library von GameTutorials verwendet.

Grundsätzlich kann ich sagen, dass von mir alles ganz gut in Klassen verpackt wurde, Operator Overloading wird für Vektoren- und Matrixmultiplikation ausgenutzt. Die Vektoren- und Matrizenklassen wurden von mir aus der CG1 Übung von Java nach C++ portiert.

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

Bezüglich der Klassen findet sich folgende Aufteilung:

Die Klasse GameEngine ist die Kernkomponente und stellt das Interface zum Anwender dar. Sie hält Referenzen auf alle weiteren Komponenten der Spiellogik:

- RenderEngine
- Scene (bestehend aus Labyrinth, Wickie, Wölfe, Pfeile,…)
- StateMachine (kümmert sich darum ob Wickie gerade läuft,…)
- Simulator (bewegt unsere Objekte (AI))
- InputHandler (führt Polling durch)
- CollisionDetection
- Sounds

Modelle und Texturierung

Alle im Spiel vorhandenen Modelle wurden von mir eigenhändig mit 3dsMax modelliert, dann nach Milkshape exportiert, dort key frame animiert mit den vorhandenen Bone/Skeleton Möglichkeiten und im LithUnwrap mit Textur versehen. Dann habe ich ein .qc file geschrieben und nach .md2 exportiert.

Die Texturen wurden von mir mit Photoshop erstellt nachdem ich mir aus den Filmen mit VirtualDub einige Bilder extrahiert habe.

Die Texture object Funktionalität von OpenGL wird natürlich verwendet.

Animationen

Besonders stolz bin ich ebenfalls auf die selbst erstellten Animationen, besonders auf den Bogenschuss, wobei Wickie einen Pfeil aus seinem Köcher zieht und schießt. Das Schießen an sich sieht man auch ganz gut im „First Person“ Modus, wenn Wickie den Bogen vor sich hält und aufzieht. Lineare Interpolation wird verwendet zum Übergang zwischen den Key Frames der Animation.

Beleuchtung und Materialien

Es wird eine direktionale Lichtquelle verwendet, die die Sonne repräsentieren soll. Die diffuse Beleuchtung sieht man sehr gut an den Wänden. Zusätzlich wird auch ein geringer glänzender Anteil verwendet. Alle Objekte werden mit dem selben Material gezeichnet.

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

Spieldesign

Collision Detection
Collision Detection ist momentan noch die Achillesferse meines Spieles, da es momentan leider wie eine ungewollte Framebremse wirkt bei größeren Levels. Das sollte sich aber spätestens dann legen sobald ich etwas Hierarchie in die Szene bringe und Collision Detection nur mehr mit der Umgebung durchführe und nicht mehr mit allen Objekten der ganzen Szene.

Das Abprüfen der Kollisionen wird über das Vergleichen von Bounding Boxen realisiert. Ganz gut ist dabei auch die Kollisionsabfrage mit der Kamera gelungen, sodass man an der Wand ansteht mit der Kamera. Sollte man aus irgendeinem Grund doch mit der Kamera durch die Wand wollen dann ist auch das möglich wenn man sehr schnell nach links oder rechts zieht mit der Maus und so den Rotationswinkel sehr schnell erhöht.

Sound
Zum Abspielen der Sounds wird FMOD verwendet. Die mp3 Dateien wurden von mir mit VirutalDub und GoldenWave aus den Filmen extrahiert bzw. nachbearbeitet. So hören sich z.B. der Schrei des Wolfes wenn man ihn abschießt, der Sound beim Bogenschuss und auch die Hintergrundmusik meiner Meinung nach ganz gut an und passen gut zum Spiel.

AI
Die Wölfe rennen immer Richtung Wickie. An den Ecken stehen sie dann aber leider an. Ich hab zwar mit dem Gedanken gespielt auch das noch zu implementieren, ich werd die Zeit aber lieber verwenden um irgendwelche grafischen Spezialeffekte zu implementieren.

FrameTimer

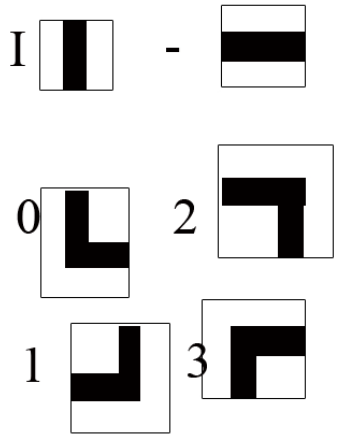
Alle Animationen und Bewegungen werden durch einen „FrameTimer“ an die Frame Rate angepasst und dadurch zeitlich konstant abgespielt, unabhängig davon wie viele Frames angezeigt werden.

Sonstige Features

Das Labyrinth ist aus lauter einzelnen Blöcken zusammengestoppelt. Man kann es sich wie ein Schachbrett vorstellen wobei jeder einzelne Block verschiedene Wandformen enthalten kann. Insgesamt gibt es zwei verschiedene Arten von Wänden, gerade Wände und Ecken. Für die geraden Wände gibt es zwei verschiedene Rotationsmöglichkeiten (0° und 180°) und für eine Eckmauer gibt es vier verschiedene Rotationsmöglichkeiten (0°, 90°, 180°, 270°). Die Modelle wurden so entworfen, dass sie annähernd nahtlos aneinander gereiht werden können, eigentlich ziemlich ähnlich zu Lego *gg*. Die Information wie ein Level aussieht ist ein einem .txt File gespeichert, wobei folgende Syntax verwendet wird bzw. werden muss:

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

Links das Zeichen und rechts sieht man seine grafische Entsprechung. Die schwarzen Balken stellen Mauern in der Vogelperspektive dar. Die Boxen sind in Wirklichkeit gleich groß und können daher dann gut aneinander gekoppelt werden.




```
'S' - Start von Wickie

'W' - Position eines Wolfes
'E' – Das Ende
'I' – Gerade Wand mit keiner Rotation, --> also
      von Norden nach Süden
'-' – Gerade Wand mit 90° Rotation, --> also von
      Westen nach Osten
'0' – Eine Eckwand mit keiner Rotation
'1' – Eine Eckwand mit 90° Rotation
'2' – Eine Eckwand mit 180° Rotation
'3' – Eine Eckwand mit 270° Rotation
' ' – "Nichts", kein Wolf, kein Wickie, nur
      Labyrinthboden
```

Libraries:

- MD2 Loader von GameTutorials (http://www.gametutorials.com/)
- JPEG Loader von GameTutorials (http://www.gametutorials.com/)
- Skybox Texturen (http://www.scentednectar.com/skyboxes/)
- FMOD für Sound, siehe CG2 Homepage
- Bitmap Font für OpenGL
- GLUT Version mit Mouse Wheel Support

Abschließende Bemerkungen

Ich bin wirklich sehr stolz auf mein Spiel. Hat zwar eine riesige Menge an Zeit verschlungen, hat aber dafür soviel Spaß gemacht wie keine andere Lehrveranstaltung und die Zeit ist ja auch im Flug vergangen. Die ganzen Osterferien und sonst auch noch eine Menge an Zeit neben dem Studium sind dafür draufgegangen, was mir aber in keinster Weise Leid tut. Aufgrund meines Doktorratsstudium weiß ich leider nicht, ob ich für die 3. Abgabe auch wieder soviel Zeit investieren kann, deshalb hoffe ich dass mir ein Großteil der Features auch bei der 3. Abgabe wieder eine Menge Punkte bringen wird.

Mit Ausnahme der unten angegebenen Quellen wurde von mir alles selbst erstellt, programmiert,…, aber natürlich hab ich mir Anregungen in einigen Tutorials geholt und nachgesehen wie man ein Kamera Klasse intelligent implementieren könnte, oder wie eine Skybox aussieht,…

Ich bin ehrlich gesagt sehr froh, dass ich schon in den Semesterferien angefangen habe das Spiel zu entwickeln, sonst wäre sich das bei mir als Ein-Mann Projekt niemals in dem Umfang ausgegangen.

Zum Abschluss noch ein kurzer Vergleich zwischen meinem Spiel und den Filmen

CG 2/3 Übungen SS 2006

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

Spiel:

Film:

Christoph Stiedl
E 086 881
0552781
christoph.stiedl@gmx.at

Wölfe im Spiel:

Wölfe im Film: