THE BRAVE LAD

Dokumentation zur 3. Abgabe für CG2/3 Laborübung

Kurzfassung des Storyboards:

Das Spiel beginnt, als der Held das unterirdische Versteck des „Fürsten der Finsternis“ betritt. Während man den Stallburschen durch die Gänge steuert, muß man verschiede Herausforderungen überwinden. So muß man Feinde bekämpfen und Rätsel lösen. Wenn man es schafft all diese zu besteht, trifft man im Endkampf auf den „Fürsten der Finsternis“. Nachdem man ihn besiegt hat, hat man die Prinzessin befreit und das Spiel auch gleich gewonnen.

Wie wird das Programm zum Laufen gebracht:

Im Verzeichnis **`\bin`** die Datei **`thebravelad.exe`** aufrufen.

Kurzbeschreibung:

Es ist ein Labyrinthspiel aus der First Person View. Man sieht die Waffe die man aktuell benutzt, vor sich. Wenn man einen Schlag ausführt, bewegt sich die Waffe. Monster lauern auf dem Weg durch das Labyrinth. Sie sind animiert und sie attackieren. Monster greifen immer dann an, wenn man sich in ihrer Nähe befindet, egal ob man vor ihnen steht oder sich von hinten anschleicht. Ziel des Spiels ist es, den Fürsten der Finsternis in seinem Versteck zu finden und niederzustrecken. Wenn Monster sterben sind sie ebenfalls animiert. Man kann PowerUps aufsammeln, die in Form von Pizzen im Labyrinth verteilt sind. Man bekommt so wieder HitPoints zurück, die man durch Kämpfe verloren hat. 2 Pizzen füllen die HitPoints auf jeden Fall wieder bis zum Maximum auf. Wenn man besiegt wird, also die HitPoints unter 0 sinken, kann man sein Schwert nicht mehr bewegen, und man hat verloren. Die Monster reagieren auch nicht mehr auf den Helden.

Steuerung:

ESC	Programm beenden
a	links (Strafe)
d	rechts (Strafe)
s	zurück
w	vorwärts
Kombinationen von a/d/s/w	man läuft schräg
SPACE	springen
linke Maustaste	mit Waffe attackieren
Maus up	hinauf schauen
Maus down	hinunter schauen
Maus left	nach links schauen
Maus right	nach rechts schauen
F1	Hilfe
F2	Frames per Second Anzeige an/aus
F3	Wireframe Modus an/aus
F4	ändert die MipMapping Textur Qualität
F5	MipMaps ein/aus
F6	Umschalten zwischen Vertex Array / Immediate Mode Triangles / ARB_vertex_buffer_object Geometrie Modi
F7	DisplayLists ein/aus
F8	View Frustum Culling ein/aus
F9	Transparenz ein/aus
F11	wenn man im Immediate Mode ist (F6 einmal drücken), kann man hiermit die Lightmaps und die Wände ohne Lightmaps anschauen

Bei F6 existiert leider ein kleiner Bug am Abgabecomputer, und zwar sieht man nach dem Wechsel vom ARB_vertex_buffer_object Mode zurück zum Vertex Array Mode keine Objekte mehr. Wenn man weiter schaltet in den Immediate Mode, sieht man den Level dann wieder. Aber das Programm startet im Vertex Array Modus, und da funktioniert dieser Modus auch.

Umsetzung des Spieldesign:

Wir haben darauf geachtet, daß wir ein vollständiges Spiel entwickeln, mit Anfangsmenu, Musik, Soundeffekten und Text als Feedback. Man kann auch sterben und das Spiel von vorne beginnen. Man muß auch einfache Rätsel lösen, sonst kann man das Spiel nicht gewinnen. Die Kämpfe laufen nach den Regeln des „d20 System“ ab, was bewirkt, daß nicht jeder Schlag sitzen muß. Wir hoffen es macht ein bißchen Spaß „The Brave Lad“ zu spielen!

Umsetzung bezüglich nichttrivialer Objekte:

Im Spiel sind die meisten Objekte (außer den Wänden) nichttrivial. Alle Monster und Figuren haben nichtkonvexe Elemente (Hörner, Zöpfe, Antennen oder ein riesiges Becken bei den Höhlenorcs). Als Objektdateiformat haben wir an dem von Anim8or exportierten C-Files festgehalten, da man die so schön einfach ins Programm übernehmen kann. Nur wenn man

die Objekte texturiert ist Handarbeit angesagt, da man die Texturkoordinaten die man von UVMapper nach der Erzeugung der Texturemap für ein Modell erhält, händisch in die C-Files von Anim8or übertragen muß. UVMapper kann leider keine Multimesh – Modelle exportieren, es werden immer alle Meshes zu einem zusammengezogen. Für die Animation sind aber Modelle mit Multimeshes nötig.

Umsetzung bezüglich Beschleunigung der Sichtbarkeitsberechnung

ViewFrustum Culling wird bei uns gegen die Wände als ganzes und gegen die Bounding Spheres von Objekten gemacht. Die Bounding Spheres werden beim Einlesen der Modelle exakt an die Größe der Meshes angepaßt.

Umsetzung bezüglich Transparenzeffekte:

Im Spiel gibt es eine Vitrine, die durchsichtige Gläser hat. Die Gläser sind an 2 gegenüberliegenden Seiten hellblau, und an den beiden anderen gegenüberliegenden Seiten grün. Das gibt schöne Transparenzeffekte.

Umsetzung bezüglich Experimentieren mit OpenGL

Alle Funktionstasten (F2 bis F9) sind mit den in der Angabe vorgegebenen Funktionen belegt. Bemerkenswert ist, daß das Umschalten der Geometriemodi fast keine Unterschiede zu erkennen sind, höchstens 10 FPS. Abhängig von der Auflösung bringt Mipmapping eine Beschleunigung mit sich, und zwar je höher die Auflösung umso mehr. ViewFrustum Culling bringt was bei niedrigen Auflösungen und im FullScreen Mode. DisplayLists erzeugen fast keine Beschleunigung. Sehr schön zu erkennen ist, daß Lightmaps eine wesentlich bessere Beleuchtung der Wände erzeugen, als es OpenGL auf so großen Wänden, die nicht genügend fein tesseliert sind, selbst macht. Wenn man im Immediate Geometrie Mode ist, kann man mit F11 zwischen Lightmaps umschalten.

Bei F4 ist zusätzlich der Modus ohne bilineare Filterung auf einer MipMap Ebene auswählbar, da sonst nur schwer ein Unterschied zwischen den MipMap Modi zu erkennen ist. Nur wenn man geht sieht man bei den langen Gängen, daß trilineare Filterung noch besser interpoliert, und man keine Streifen zwischen den verschiedenen Levels der MipMaps erkennen kann.

Welche Spezialeffekte wurden implementiert

Es wurden 2 Spezialeffekte implementiert.

- Transparenz mit Sortierung der Objekte
- Lightmaps

Wie wurden die Spezialeffekte implementiert

Beide Spezialeffekte wurden auf Beispielprogrammen aufgebaut, die zum Paper „Advanced Graphics Programming Techniques Using OpenGL“ vorhanden sind. Die Transparenz wurde auf dem Programm „multialphablend.c“ aus den Beispielprogrammen von der Siggraph’97 aufgebaut. Erweitert wurde das Programm damit, daß die transparenten Objekte abhängig von der Spielerposition sortiert werden, und sie werden auch als letzte Objekte des gesamten Levels gezeichnet.

Die Lightmaps basieren ebenfalls auf einem Programm von der Siggraph’97, und zwar auf „lightmap.c“. Das Programm wurde extrem erweitert, da in der Originalversion nur eine pseudo Lightmap von einer entfernten Lichtquelle erstellt wird. Um auch die Geometrie eines Levels zu berücksichtigen, wurde ein einfacher Ray-Casting Algorithmus implementiert, der alle Lichtquellen berücksichtigt und die Distanz zur beleuchteten Wand mißt. Auch wird das Cosinus Gesetz berücksichtigt. Die Lightmaps werden erst berechnet, nachdem der ganze Level aufgebaut worden ist. Dadurch wird ein relativ genauer Schattenwurf von Wänden erzeugt. Die Lightmaps werden, nachdem sie einmal berechnet wurden, als TGA Datei gespeichert.

Links:
http://www.cg.tuwien.ac.at/courses/CG23/software/sig99advopengl.PDF
http://www.sgi.com/software/opengl/advanced97/programs/multialphablend.c
http://www.sgi.com/software/opengl/advanced97/programs/lightmap.c

Sonstige Besonderheiten

Es wurde ein komplettes Animationssystem entwickelt, mit dem alle Bewegungen der Figuren im Level erzeugt und gesteuert werden. Für jede Figur sind 11 Körperteile ansprechbar. Beim Gehen folgt das Becken den Bewegungen der Beine, es bleibt also nicht immer auf derselben Höhe, sondern hebt und senkt sich entsprechend der Schrittstellung. Die Idee zu diesem Animationssystem basiert auf einem Paper von dev-gallery.com, aber anscheinend ist deren Domain gerade nicht zugänglich, darum habe ich das Paper auf meinen Webspace an der TU gestellt. Die Methode `figure::animate_find_base_move()` ist aus diesem Paper entnommen, sonst nichts.

Die Kämpfe basieren auf den Regeln des „d20 Systems“, das verwandt ist mit den Advanced Dungeons&Dragons Regeln. Dabei werden die Spielzüge wie Treffer und Abwehr ausgewürfelt, und jeder Character hat Statistiken, die dann die Ergebnisse der Würfe beeinflussen. Das erhöht die Spannung bei den Kämpfen. Das Langschwert das man findet verändert auch die Statistiken des Helden zum Positiven.

Links:
http://stud3.tuwien.ac.at/~e9125272/arc/dev-gallery.com_3d_case_study_using_opengl.pdf
http://www.wizards.com/default.asp?x=d20/welcome

Welche Zusatztools wurden verwendet

Als WindowManager wird GLUT verwendet in der modifizierten Version von der CG23 Homepage. Als SoundSystem haben wir FMOD verwendet.

Der TGA-Loader, die Collision-Detection „Library“ (eigentlich sind es 2 Files, nämlich „3DMath.h“ und „3DMath.cpp“) und der Ansatz für das ViewFrustum wurden von GameTutorials.com verwendet. Für die Collision Detection gegen Wände wurden Ideen aus dem Camera-World Tutorial verwendet.

Von UltimateGameProgramming.com wurde die TGA-Abspeicherroutine verwendet. Es werden keine Screenshots, sondern die berechneten Lightmaps gespeichert.

Die Extensions werden mit Hilfe der von Intel entworfenen „GLsdk library“ angesprochen. Man hat damit alle Extensions genau mit ihrem Namen zur Verfügung und kann sie wie ganz normale OpenGL Befehle verwenden. Die Funktion `IsExtensionSupported()` stammt aus dieser Library.

Links:
http://www.gametutorials.com/download/OpenGL/TextureMapping4_OGL.zip
http://www.gametutorials.com/download/OpenGL/SpherePolyCollision_OGL.zip
http://www.gametutorials.com/download/OpenGL/FrustumCulling_OGL.zip
http://www.ultimategameprogramming.com/zips/Gl_TgaScreenShot.ZIP
http://oss.sgi.com/projects/ogl-sample/sdk.html

Mit welchen Tools wurden Objekte modelliert

Alle Objekte wurden mit Anim8or modelliert. Die Schnittmuster für die Texturen wurden mit UVMapper (classic) erzeugt, und dann mit GIMP bemalt. Die Texturkoordinaten der Modelle erhält man von UVMapper. Das File „Anim8or.h“ stellt die Strukturen zur Verfügung, um auf die Daten der von Anim8or exportierten Modelle zugreifen zu können.

Links:
http://www.anim8or.com/main/index.html
http://uvmapper.com/classic.htm

Walkthrough durch das Spiel:

Ganz am Anfang sieht man in der rechten hinteren Ecke des Raumes eine Vitrine die den Transparenzeffekt vorführt. Wenn man in Nord-Süd Richtung durchschaut ist das Glas grünlich, wenn man in Ost-West Richtung durchschaut ist das Glas bläulich.

Links hinten in den Gang hinein sieht man sehr gut den Schattenwurf der Wand, der durch die Lightmaps berechnet wurde. Bei der ersten Brücke nach links ins Wasser ist der Weg zu den Teletubbies. Den Fluß entlang, der um die Ecke biegt, und die Stufen hinauf zu den Teletubbies sollte man versuchen alle Teletubbies zu killen, da einer von ihnen ein sehr mächtiges Schwert in seinem Inventory hat. An der linken Wand, auf der entgegengesetzten Seite der Säule vom Licht, befindet sich eine Tür aus Holz ohne Schloß. Nachdem man das Schwert hat, also wieder zurück den Fluß entlang und jetzt nach links, weiter über die Brücke. Hier am Eck hat man einen sehr guten Blick auf die Effekte die die Lightmaps bringen.

Geht man den Gang weiter, sieht man 2 Höhlen-Orcs. Hier hat man 2 Möglichkeiten. Entweder man springt an der Seite vorbei, oder man greift sie dann an, wenn sie einem den Rücken zudrehen. Wenn sie auf einen zukommen, sollte man sie nicht angreifen, das überlebt man nur mit Glück. Hinten im Gang befindet sich ein Schalter, den man umlegen muß. Er öffnet die Holztür bei den Teletubbies.

Nachdem man den Schalter umgelegt hat, und die Höhlen-Orcs überlebt hat, geht man zur jetzt offenen Holztür bei den Teletubbies und folgt dem Gang bis zu einem Loch im Boden. Hier kann man wieder ein paar gute Lightmaps Effekte bestaunen. Dort sieht man den

Fürsten der Finsternis auf und ab gehen. Wenn man am Rand stehen bleibt, kann man so lange warten, bis der Fürst der Finsternis einem den Rücken zukehrt, und dann springt man hinunter. So kann man den ersten Schlag landen. Man wartet immer am besten, bis sich der Fürst der Finsternis von einem abwendet um ihn erneut anzugreifen. Im Hintergrund wartet die Prinzessin auf ihre Befreiung. Wenn man in die Lava steigt, gibt das Abzüge bei den Hitpoints. Hat man den Fürsten besiegt, hat man das Spiel gewonnen.