Kabu's Journey

Abgabe 3 - Doku

Fritz-Michael Gschwantner - e0627004
Gabriele Hebart - e0626511

20. Juni 2007

1 Implementierung

1.1 Gameplay

Ziel ist es, das Tor auf der finalen Plattform zu erreichen und dabei nicht zu sterben.

1.1.1 Orbs

Wichtiges Gameplay Element in unserem Spiel sind die sognenannten „Orbs". Zielt man auf einen Orb und drückt die *linke Maustaste*, kann ein Orb „aktiviert" werden (sofern er sich in Reichweite befindet). Aktiviert man einen Orb, wird der Player von diesem angezogen und bewegt sich darauf zu. Aktiviert man mehrere Orbs auf einmal bzw. nacheinander wird man von allen Orbs angezogen. Um einen Orb wieder zu deaktivieren reicht es wieder auf ihn zu zielen und die *linke Maustaste* zu betätigen. Mit einem Klick auf die *rechte Maustaste* kann man alle aktivierten Orbs auf einmal deaktivieren. Diese Technik benötigt man vor allem, um auf andere Plattformen zu gelangen.

1.1.2 Gegner

Auf Gegner kann draufgesprungen werden, um sie zu erledigen. Die Gegner können dem Spieler Schaden zufügen und versuchen ihm den Weg abzuschneiden, sobald er in Ihre Nähe kommt. Sie werden auch möglichst nicht von der Plattform springen.

1.1.3 Kollisionserkennung

Wir verwenden Raycasting (Ray vs. Mesh Kollision) um Objekte auf Meshes gehen zu lassen, also um den Player und die Gegner auf den Plattformen wandern zu lassen. Zwischen Gegner und Spieler ist außerdem noch eine

Mesh-Mesh Kollisionserkennung eingebaut. Gegner kollidieren leider nicht miteinander.

1.1.4 Steuerung

Unser Spiel verwendet die Standard „WASD Steuerung“. *W* ist also vorwärts, *A* rückwärts, *S* seitwärts links, *D* seitwärts rechts, relativ zur Blickrichtung. Mit der Maus kann die Blickrichtung verändert werden, wobei die Maus nicht invertiert ist (also Maus nach vorne bedeutet Blick nach oben). Mit der *Leertaste* kann man den Player springen lassen. Die Sprungrichtung während des Sprunges ändern ist nicht möglich.

1.2 Nichttriviale Objekte

Sämtliche zu rendernden Objekte in unserem Spiel werden von einem Fbx File geladen, d.h. das gesamte Level wird in Maya modelliert und in einem FbxFile abgespeichert, beim Laden des Spieles wird dieses Fbx File ausgelesen, und die entsprechenden Objekte werden generiert.

1.3 Animierte Objekte

Bei uns werden 2 Arten der Objekte animiert, der Player und die Orbs. Die Orbs rotieren sich und der Player verfügt über eine Animation seiner Füße sobald er sich bewegt. Realisiert wurden die Animationen nicht über Animationen die aus dem Fbx File geladen wurden, sondern diese wurden in den jeweiligen Klassen hardgecodet. Der Player besteht beispielsweise aus drei Meshes, wobei auf die Meshes der Füße Rotationen angewendet werden.

1.4 Beschleunigung der Sichtbarkeitsberechnung

In unserer Applikation ist View-Frustum Culling eingebaut. Dabei wird einfach nur die Bounding-Sphere jedes Objektes gegen den Frustum getestet. Mit F10 kann man sich die Bounding-Spheres anzeigen lassen, mit F11 kann man sich den aktuellen Vertex- und Triangle-Count anzeigen lassen.

1.5 Transparenz-Effekte

Orbs besitzen eine transparente Textur. Die Orbs werden untereinander relativ zur Distanz des Players sortiert (also Orbs die weiter hinten liegen, werden zuerst gezeichnet). Die Orbs an sich werden nach der anderen Geometrie gezeichnet.

Das HUD behinhaltet außerdem auch transparente Texturen und wird daher natürlich auch zuletzt gezeichnet.

1.6 Experimentieren mit OpenGL

Natürlich wurden wie gefordert folgende Funktionen auf den F-Tasten implementiert:

- F2: FPS Anzeige.
- F3: Umschalten zwischen normalen Rendermodus und Wireframe.
- F4: Umschalten der Texturfilterung zwischen Nearest Neighbour und Bilinearer Interpolierung.
- F5: Einschalten von Mip-Mapping und umschalten zwischen Nearest Neighbour und Linearer Interpolierung im Mip-Mapping Modus.
- F6: Umschalten zwischen Vertex Buffer Objects, Arrays und Immediate Mode.
- F7: Ein- und ausschalten von Display Listen. Der Array und Immediate Mode kann mit Display Listen kombiniert werden.
- F8: View Frustum Culling ein- und ausschalten.
- F9: Transparenz ein- und ausschalten.
- F10: Bounding-Spheres anzeigen.
- F11: Vertex- und Triangle-Count anzeigen (um den Effekt von View Frustum Culling nicht nur durch FPS zu sehen).
- F12: Alternative Ansicht.

Es wird ein Feedback im HUD angezeigt, sobald man etwas an diesen Einstellungen ändert.

2 Effekte

2.1 Per-Pixel Lighting

Wir verwenden Blinn Shading. Unser Shader berechnet das shading komplett im Fragment Shader, wodurch wir Per-Pixel Lighting erreichn.

2.2 Shadow Maps

Shadow Maps wurden über den Shader implementiert. Die Shadow Map der direktionalen Lichtquelle wird immer ungefähr um den Player positioniert. Schatten die an die Shadow Map angrenzen werden weich ausgeblendet.

Wir hatten allerdings Probleme auf NVidia Grafikkarten. Die `tex2Dproj` Funktion von Cg verursachte plötzlich, dass sich das gesamte Objekt selbst

abschattet, wie wenn man kein Front-Face-Culling bei der Shadow Map machen würde. Natürlich wird bei uns Front-Face-Culling eingesetzt und auf ATi Grafikkarten funktioniert auch alles. Als temporären workaround haben wir daher die Objekte in Shadow-receiver und Shadow-caster eingeteilt, damit sich Objekte gar nicht mehr selbst abschatten können. Im Endeffekt sind also derzeit unsere Plattformen die Shadow-receiver, der Rest sind Shadow-caster.

3 Besonderheiten

3.1 Levels

Die Objekte werden beim Laden eines Levels generiert. In der Klasse *Level* wird pro Level eine FBX Datei geladen, worin sich alle Objekte befinden. Der *Player* wird anschließend noch gesondert in der Klasse *Scene* gespeichert. Aus einem Level kann anschließend der *SceneGraph* (eine Liste aller Entities des Levels) geholt werden.

4 Tools

4.1 Libraries

- Collision Detection: OPCODE[1]
- Model Loading: FBX[2]
- Math Library: Wild Magic 4[3]
- Image Loading: DevIL[4]
- OpenGL Utility: freeGLUT[5]
- Shader: CG[6]

4.2 Modellierung

Zum Modellieren und Texturieren unserer Modelle haben wir Maya von Autodesk[7] verwendet.

[1] `http://www.codercorner.com/Opcode.htm`
[2] `http://www.autodesk.com/fbx`
[3] `http://www.autodesk.com/fbx`
[4] `http://openil.sourceforge.net/`
[5] `http://freeglut.sourceforge.net/`
[6] `http://developer.nvidia.com/page/cg_main.html/`
[7] `http://www.autodesk.de/maya`