ROCK N ROLL

Andreas Ritzberger | Nicolas Pühringer

Gameplay

2 Spieler treten gegeneinander in einer „Arena" an um in einem Wettkampf der Geschwindigkeiten die begehrten Punkte zu ergattern. Zu Spielbeginn starten beide Spieler nebeneinander und nehmen langsam Geschwindigkeit auf. Durch abwechselndes Hämmern auf die Steuerungstasten beschleunigen sie immer mehr! Liegt ein Spieler vor dem anderen, so verwandelt sich dieser in den „Rock" und wird zum Verfolgten. Der andere Spieler wird zum mächtigen „Roll" dessen Ziel es ist den „Rock" zu überrollen. Gelingt ihm das, so erhält er einen Punkt, läuft „Rock" zu weit davon, so entkommt er der Bedrohung und erhält seinerseits einen Punkt. Nach der Vergabe des Punktes werden beide Spieler wieder gleich auf gesetzt und die ewige Jagt kann erneut beginnen! Siegreich ist jener Spieler, der sich zuerst 9 Punkte erkämpfen konnte. Aber nicht nur der Gegner, sondern auch die Arena selbst hält einige Überraschungen für die Spieler bereit! So beschleunigen manche Items den Spieler auf Maximalgeschwindigkeit während ihn andere bremsen. Auch vor Hindernissen auf der Bahn müssen sich die Spieler in Acht nehmen!

Der Startscreen

Den Einstieg in das Spiel bildet der Startscreen. Hier sieht man eine Auswahl spielbarer Arenen die man durch drücken der entsprechenden Tasten („1" – „3") auswählen kann. (Achtung! Etwas geduld bitte, die Levels brauchen ein wenig zum Laden)

Das Spiel Interface

An den oberen Ecken des Bildschirmes befinden sich die Spielersymbole, Punkte und Geschwindigkeitsanzeige. Rechts für Spieler 1, links für Spieler 2.

Die Spielsteuerung

Spieler 1: Steuertasten sind „A" und „D". Durch abwechselndes Drücken der Tasten bewegt sich der Spieler nach vorn (je schneller die Tasten gedrückt werden, desto schneller wird der Spieler). Wird nur die „A" – Taste gedrückt, bewegt sich der Spieler nach links (auch hier umso schneller je schneller gedrückt wird). Wird nur die „D" – Taste gedrückt, bewegt er sich nach rechts.

Spieler 2: Steuertasten sind „←" und „→" – Pfeiltasten. Ansonsten ist die Steuerung analog zu Spieler 1.

Hat ein Spieler gepunktet, wird das Spiel angehalten und kann durch Drücken der „R" Taste wieder angefangen werden.

Möchte man zurück zum Startscreen um eine andere Arena auszuwählen genügt ein einmaliges Drücken der „Esc" Taste.

(Eine Auflistung aller Tasten ist auf der letzten Seite zu finden)

Nicht triviale Objekte

Sämtliche Objekte (Spieler, Boden, Wände, Goodies, Hindernisse, Dekoelemente) wurden im Milkshape3d gebaut und anhand eines Modelloaders (http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=31), der adaptiert wurde um VBOs und Vertex Arrays zu bekommen, werden diese ins Spiel geladen.

Animierte Objekte

2 Objekte werden in unserem Spiel animiert: das „Rock" und das „Roll" Modell. Ziel war es, die Spieler abhängig von ihrer Geschwindigkeit zu animieren. So wird der „Roll" abhängig seiner Geschwindigkeit um seine Z-Achse rotiert, während beim „Rock" – Modell die Dauer der Keyframes geändert werden.

Beschleunigung der Sichtbarkeitsberechnung

Zur Beschleunigung der Sichtbarkeitsberechnung wurde Frustum Culling implementiert. Hier werden die die Bounding Spheres der Modelle zur Berechnung herangezogen. (Die Anleitung dazu stammt aus dem Realtimerendering Buch **ISBN-10:** 1568811829)

Transparenz Effekte

Da in unserem Spiel transparente Objekte an sich nicht sinnvoll eingesetzt werden können, haben wir uns damit beschäftigt, unser Spiel Interface aus transparenten Texturen aufzubauen. Diese Texturen werden per Alpha – Test transparent gesetzt und durch ausschalten des Depth – Tests immer im Vordergrund angezeigt.

Experimentieren mit OpenGL

Siehe Tastaturbelegung

Zusatztools

CML (http://cmldev.net/)

Glew (http://glew.sourceforge.net/)

DevIL (http://openil.sourceforge.net/)

GLUT

Per Pixel Lighting nach Phong

Normale, Lichtrichtung und Half- Vektor werden im Vertex Shader berechnet. Ambient, Diffuse, Specular Anteile werden im Fragment Shader berechnet und mit der Objekt Textur gemixt.

Shadowmapping in GLSL

Es wird pro Frame eine Shadowmap aus Sicht der Lichtquelle berechnet und in ein Framebufferobject geschrieben. Die Texturmatrix wird so berechnet, dass die Vertex positionen in Shadowmap Koordinaten umgerechnet werden können. In GLSL wird mittels der shadow2DProj() Funktion berechnet ob ein Fragment im Schatten liegt oder nicht. glPolygonOffset wird beim zeichnen in den Framebuffer benutzt um z-Fighting zu verhindern.

Tutorial:
http://www.cs.cmu.edu/afs/cs/academic/class/15462/web.06s/asst/project3/shadowmap/

Partikel System mit Point Sprites

Hinter dem Verfolger (Kugel) wird mittels Point Sprites eine kleine Staubwolke gezeichnet. Das Partikelsystem wird je nach Spielerposition und Bewegungsrichtung neu orientiert und für ein fixiertes Zeitintervall gezeichnet.

Tutorial: http://www.codesampler.com/oglsrc/oglsrc_6.htm

Besonderheiten

Das Laden des Spielfeldes erfolgt bei uns in 2 Stufen. Zum einen werden die in der Szene vorkommenden Modelle aus einem Textfile geladen, dann werden diese Objekte anhand des Level Files angeordnet. Das Spielfeld besteht aus so genannten Level Elementen, die den Boden repräsentieren. Zu jedem Boden Element gibt es 2 Wand Elemente, die durch die Attribute „left, right, front, back" auf dem Level Element angeordnet werden. Dadurch ist der Aufbau eines Levels einfach und Flexibel zu gestalten. Genauso wie die Wände werden alle zusätzlichen Elemente wie „Goodies" oder Hindernisse hierarchisch als Child – Nodes dem Level Element hinzugefügt. Zusätzlich besteht die Möglichkeit Dekoelemente frei zu positionieren und dem Szenegraph hinzuzufügen.

Kamera

Unser Ziel war es, eine dynamische Kamera zu implementieren, die die Dynamik und Geschwindigkeit unseres Spieles weiter verstärkt. Unsere Kamera folgt den Spieler nicht nur, sondern rotiert sich und ändert die Entfernung zu den Spielern abhängig von deren Distanz zueinander und dem Winkel den die Spieler zur Laufrichtung haben. Das ermöglicht es uns, dass stets beide Spieler auf dem Bildschirm zu sehen sind. Um unser Kameramodell zu implementieren, müssen für jeden Frame die Center und Eye Punkte der gluLookAt(..) gesetzt werden. Der Center Punkt ist die Hälfte der Linie zwischen Player1 und Player2 und der Eye Punkt ist auf der Normalen auf diese Linie von Center ausgehend angeordnet. Die Höhe der Kamera errechnet sich aus der Entfernung der Spieler zueinander die mit dem Winkel der Spieler zueinander gewichtet wird.

Steuerung

Die Steuerung und die damit verbundene Bewegung der Spieler Objekte ist Framerate unabhängig (wie bei den „hilfreichen Tipps" beschrieben). Dann werden die Tasten überwacht. Werden die Tasten gedrückt, wird dieser Zeitpunkt mit der Zeit des vorigen Tastendrucks verglichen. Je geringer diese Zeit ist, desto höher wird der Geschwindigkeitsfaktor gesetzt.

Minimap

Zum Erstellen der Minimap ist ein 2. Renderpass nötig. Hierzu wird eine 2. Kamera verwendet. Die Szene wird dann aus Sicht dieser Kamera in ein Framebuffer Object gerendert das dann als Textur dem Minimap Plane zugewiesen wird. Um Performance zu sparen wird die Minimap nur alle 10 Frames neu gerendert.

Mit welchen Tools wurden die Objekte modelliert

Die Models wurden mit Milkshape3D modelliert

Tastaturbelegung

A	Wird nur die A Taste gedrückt -> Spieler 1 bewegt sich nach links, wird sie abwechselnd mit der B Taste gedrückt -> Spieler 1 bewegt sich nach vorn
B	Wird nur die B Taste gedrückt -> Spieler 1 bewegt sich nach rechts, wird sie abwechselnd mit der A Taste gedrückt -> Spieler 1 bewegt sich nach vorn
←	Wird nur die ← Taste gedrückt -> Spieler 2 bewegt sich nach links, wird sie abwechselnd mit der → Taste gedrückt -> Spieler 2 bewegt sich nach vorn
→	Wird nur die → Taste gedrückt -> Spieler 2 bewegt sich nach rechts, wird sie abwechselnd mit der ← Taste gedrückt -> Spieler 2 bewegt sich nach vorn
R	Wird das Spiel nach dem Punkten eines Spielers angehalten, wird es mit der R Taste wieder aufgenommen
Esc	Startscreen: Spiel beenden, Gamescreen: zurück zu Startscreen
1	Startscreen: Auswahl Level 1
2	Startscreen: Auswahl Level 2
3	Startscreen: Auswahl Level 3
	Experimentieren mit OpenGL
F2	Ein / Ausschalten der Anzeige der aktuellen Einstellungen (Framerate, Frustum Culling, Mipmap, etc.
F3	Wireframe / Normal
F4	Texturqualität ändern
F5	Mipmap ein / aus
F6	VBO / Vertex Arrays / Immediate Mode

F8	Frustum Culling ein / aus