CG2LU, SS 2010

Abgabe 3

Beschreibung der Anwendung „Schneckmania“

Boindl, Matthias, 033 532, 0727922, e0727922@student.tuwien.ac.at
Freude, Christian, 033 532, 0728278, e0728278@student.tuwien.ac.at

2010-06-17

1 Features

Object-Reader Dieser liest aus OBJ-Dateien Vertizes mit ihren Normalen, Textur-Koordinaten und Face-Zuordnung.

Material-Reader Liest eigenes Format, welches via Python aus Blender exportiert wird.

Texture-Wrapper Lädt TGAs und lädt sie in die gewünschte Texture-Unit.

Scene-Reader Liest aus einem eigenen Scene-Format, welches via Blender exportiert wurde diverse Informationen bezüglich Kamera (Positionierung, Orientierung), zu ladende Szenen-Objekte (und deren Attribute) und Lichtquellen.

Physik Durch Einbinden der *Bullet Pysics Library*[1] und Anpassen des Bullet Vehicle Demos aus den Beispieldateien der Library wird unserem Protagonisten das Fahren bzw. Unfallbauen auf der Strecke ermöglicht.

Frustum-Culling Entscheidet durch Bounding-Spheres ob ein bestimmtes Objekt gezeichnet werden muss. Die Visualisierung der Bounding-Spheres erfolgt über achsenparallele Boxen, deren Seitenlängen dem doppelten Bounding-Sphere-Radius entsprechen.

[1] `http://bulletphysics.org/`

2 Steuerung

Das Spiel wird ohne Menü durch das Drücken der beschriebenen Tasten gesteuert.

- **Pfeiltasten**: Steuert die Schnecke in konventioneller Rennspielmanier
- **C**: Wechselt zwischen der frei beweglichen und der Folgekamera
- **F2**: Gibt die aktuelle Framerate auf die Konsole aus
- **F3**: Zeichnet die Szene im Wireframe-Modus
- **F4**: Wechselt das Texturfiltering zwischen Nearest Neighbor und Bilinear
- **F5**: Wechselt die MipMapping-Qualität zwischen Aus, Nearest Neighbor und Linear
- **F8**: Aktiviert bzw. deaktiviert das Viewfrustum-Culling
- **F9**: Aktiviert bzw. deaktiviert Transparenz
- **WASD**: Steuert die Position der frei beweglichen Kamera
- **Maus**: Durch Ziehen der Maus bei gedrückter linker Taste kann die frei bewegliche Kamera ausgerichtet werden
- **R**: Setzt Kamera und Schneke auf Ursprungsposition zurück
- **Strg links**: Einsatz des Schleimturbos
- **L**: Startet den Tag/Nachtzyklus in verzerrter Zeitwahrnehmung
- **B**: Zeichnet Bounding Boxes der Objekte an deren Seitenlänge dem Durchmesser der eingesetzten Bounding-Spheres entspricht
- **O**: Zeigt an wieviele Objekte zur Zeit gezeichnet werden
- **V**: Zeichnet das Bullet-Vehicle-Gerüst der Schnecke
- **M**: Aktiviert bzw. deaktiviert geschwindigkeitsabhängige Schneckendeformation
- **Leertaste**: Betätigt die Handbremse

3 Umsetzung

Grundlegendes Gameplay Umgesetzt wurde ein Schneckenmodell, physikbasierende Steuerung und eine dekorierte Strecke. Es ist möglich durch betätigen der Pfeiltasten Runden zu absolvieren und die gefahrene Bestzeit anzeigen zu lassen. Links unten ist die aktuelle Geschwindigkeit und rechts unten der Schleimvorrat zu sehen.

Die geplanten Gameplay-Features Menü, Gegner sowie den streckenbezogenen Gimmicks konnten leider nicht zeitgerecht umgesetzt werden, da der Aufwand einer KI-Implementierung und der Content-Generation unterschätzt wurden.

Bewegte Objekte Das einzig sichtbare bewegte Objekt ist die Schnecke selbst. Weiters werden die Lichtquelle für Nachtfahrten sowie die für die Erzeugung der Shadowmaps in der Szene bewegt. Durch diese Bewegungen werden Tag/Nacht-Zyklen umgesetzt und ein Scheinwerfer vor der Schnecke positioniert.

Texturierte Objekte Texturierung wurde dort eingesetzt, wo deren ästhetischer Nutzen den Entwicklern einleuchtete. Die Generierung der UV-Koordinaten erfolgt bereits mittels Blender. Das Laden des dazugehörigen Bildes erfolgt durch GLFW. Nach dem Laden werden diese Daten OpenGL übergeben, welches auch die passenden Mipmaps generiert.

Beleuchtung Für die Beleuchtung von Strecke und Schnecke wurde im Shader auf das bewährte Phong-Modell zurückgegriffen. Sowohl Phong-Illumination sowie Phong-Shading kommen zum Einsatz. Schatten werden durch Shadow-Mapping erzeugt.

Modellierung Die Modellierung erfolgte ausschließlich mit Blender.

4 Spezialeffekte

Shadow-Mapping In dieser Abgabe wird Shadow-Mapping mit hardwareseitigem PCF (percentage closer filtering) eingesetzt. Um vom Shadow-Mapping sichtbaren Gebrauch zu machen kommt dieses Verfahren bei der Umsetzung der Tag/Nacht-Zyklen zum Einsatz. Bei der Umsetzung wurde auf Lehrveranstaltungsfolien und zwei[2] Online-Tutorials[3] zurückgegriffen

Motion-Blur Dieser Effekt wurde nach einem Online-Tutorial[4] umgesetzt. Sichtbar wird dieser Effekt aber erst, wenn der Spieler vom Schleimturbo Gebrauch macht. Die so erhöhte Beschleunigung wird durch diesen Effekt noch unterstrichen.

5 Zusätzliche Libraries

Alle externen Libraries sind im Unterordner *external* zu finden. Für den Prototypen wurden GLFW[5], GLEW[6], GLM[7] und Bullet[8] eingesetzt.

Übernommene Codefragmente sind in den Headerdateien und main.cpp festgehalten.

[2] http://www.fabiensanglard.net/shadowmappingPCF/index.php
[3] http://www.fabiensanglard.net/shadowmapping/index.php
[4] http://http.developer.nvidia.com/GPUGems3/gpugems3_ch27.html
[5] http://www.cg.tuwien.ac.at/courses/CG23/demos/glfw.zip
[6] http://sourceforge.net/projects/glew/files/glew/1.5.3/
[7] http://sourceforge.net/projects/glf/files/glm/glm-0.8.4.4/glm-0.8.4.4.zip/download
[8] http://bullet.googlecode.com/files/bullet-2.76.zip