CG2LU, SS 2007

GRAVITON

Dokument:

Dokumentation der 3. Abgabe

Dokumentversion: 1.0
Letzte Änderung: 20.06.2007

Name	E-Mail	Matrikelnummer	Studienkennzahl
Michael Weiß	weiss_m@gmx.at	0527751	033532
Georg Molzer	uni@molzer.at	0525148	033532

Inhaltsverzeichnis

1. Anforderungen

1.1. Gameplay

Graviton hat sich grundsätzlich planmäßig entwickelt: Der Spieler steuert einen schwebenden Gleiter durch Achterbahn-ähnliche Strecken, wobei sich ein starkes Geschwindigkeitsgefühl einstellt. Aufgrund der eingesetzten Physik-Engine hat der Gleiter auch ein sehr realistisches Fahrverhalten, welches nach intensiver Abstimmung so ziemlich unseren Erwartungen entspricht.

In einer Kooperation mit BMW ;) wurden so z.B. Fahrdynamik-Features wie eine geschwindigkeitsabhängige Lenkung und auch elektronisch abgeriegelte Höchstgeschwindigkeit implementiert.

Leider mussten aber auch Abstriche gemacht werden: So konnten keine gegnerischen Fahrer eingebaut werden, weshalb das Spiel sich im Moment auf einen 3-Runden-Time Trial Modus beschränkt. Auch gibt es keine Power-Ups und die Nutzung einer Waffe konnte in dem Zeitraum leider auch nicht implementiert werden.

Uns war ein gut steuerbares Fahrverhalten kombiniert mit guter Grafik scheinbar wichtiger, als halb-fertige Features zu implementieren.

Features:

- Dreidimensionale, physikalisch korrekte Streckenführung inkl. Loopings, Korkenzieher, Sprüngen etc.
- Waypoint-System, dass den Spieler nach dem Herausfliegen aus der Strecke wieder auf die letzte Position zurücksetzt.
- Verschiedene Kameraperspektiven, wobei 2 innerhalb des Gleiters positioniert sind und mit einem 3D-Visor die Kopfbewegung direkt ins Spiel umsetzen.
- Bei 1235 km/h wird die Schallmauer durchbrochen, was sich in einem Überschnallknall bemerkbar macht. Auch bietet die Graviton-Physik hier einen Geschwindigkeits-Impuls, der für kurzfristig höheren Speed sorgt.
- Zeitmessung auf 1/1000 Sekunde genau ;)
- Musik-Player

1.2. Nichttriviale Objekte

Wir verwenden, außer für Partikeleffekte und HU-Displays ausschließlich in **3ds max** generierte und nach FBX exportierte Meshes.

1.3. Animierte Objekte

Außer dem physikalisch animierten Gleiter gibt es in Graviton den „Antigravitationsgenerator“, der hierarchisch animierte Ringstrukturen zur Aufhebung der Schwerkraft nutzt.

1.4. Beschleunigung der Sichtbarkeitsberechnung

Außer Backface-Culling bietet Graviton zur Zeit keine Algorithmen zur Sichtbarkeitsbeschleunigung.

1.5. Transparenz-Effekte

Wir verwenden Alpha-Blending bzw. Transparenzeffekte im Menü/HUD, bei den Antriebs-Partikelsystemen und in der Kraftfeld-Barriere am Streckenrand.

1.6. Experimentieren mit OpenGL

1.6.1. Geometrie-Modi: Vertex Arrays vs. Immediate Mode Triangles vs. ARB_vertex_buffer_object

Wir haben zwar mit Immediate Mode und Vertex Arrays zu Beginn experimentiert, aber sind dann auf VBO's umgestiegen. Bei dem Versuch, die verschiedenen Geometrie-Modi letztlich wieder einzubinden stießen wir auf seltsame Probleme, die ein rechtzeitiges Fertigwerden nicht möglich machten. Daher ist derzeit nur das Rendering über VBO's integriert.

1.6.2. Mip Mapping (ein/aus)

Durch betätigen der F5 Taste kann der Spieler zwischen den Mip Mapping Arten „Aus", "Nearest Neighbor" und „Linear" wählen.

1.6.3. Textur-Qualitätseinstellungen

Durch betätigen der F4 Taste kann der Spieler Bilineare Texturfilterung ein- bzw. ausschalten.

1.6.4. Display Lists (ein/aus)

Wegen Zeitmangel noch nicht implementiert.

1.6.5. Weitere Punkte

FPS-Anzeige wurde implementiert – F2 aktiviert/deaktiviert sie.

Wireframe-Modus aktivieren/deaktivieren durch betätigen von F3.

1.6.6. Übersicht

F1 - Hilfe (falls vorhanden)	**n/a**
F2 - Framerateanzeige ein/aus	**OK**
F3 - Wire Frame ein/aus	**OK**
F4 - Texturqualität verändern: Nearest Neighbor/Bilinear	**OK**
F5 - MipMapping-Qualität ändern: Aus/Nearest Neighbor/Linear	**OK**
F6 - Vertex Arrays vs. Immediate Mode vs. den von euch gewählten Modus	**n/a**
F7 - Display Lists ein/aus	**n/a**
F8 - View frustum culling ein/aus	**n/a**
F9 - Transparenz ein/aus.	**OK**
Die durchgeführte Änderung sollte auch durch einen kurz am Bildschirm eingeblendeten Text bestätigt werden.	**OK**

1.7. Sonstige Tastenbelegung

1	Kamera 1
2	Kamera 2
3	Kamera 3
4	Kamera 4
5	Kamera 5
9	Musik – Pause
0	Musik – skip track
p/Esc	Pause
r	Reset d. Kamera
z	Physx debug mode
m	Motionblur on/off
y	Diffusemapping on/off
x	Parallaxmapping on/off
c	Normalmapping on/off
b	S on/off
Steuerung:	
w,a,s,d (auch über Cursortasten möglich)	Gleiter beschleunigen/bremsen/lenken

2. Spezialeffekte

Folgende Spezialeffekte wurden implementiert:

- **Motion Blur**

  Wurde sehr simpel durch Nutzung des Accumulation-Buffers umgesetzt.

- **Partikel-Effekte**

  Wird bei den Triebwerksauslässen des Gleiters eingesetzt. Anregungen gab es durch Artikel von „nehe“. Die Partikel variieren je nach Alter in Farbe und Transparenz und bewegen sich auch mit einer geringen Streuung.

- **Phong Shading**

  Anreize bot hier die CG1-Übung der TU Wien; Umgesetzt wurde dieser Effekt in GLSL und läuft daher sehr effizient.

- **Normal Mapping (funktioniert noch nicht fehlerfrei)**

  Nachdem das Einfügen von Phong-Shading schön funktionierte, war dies der nächste Schritt: Leider erwies sich die Errechnung der Tangenten etwas problematisch, weshalb es bei diesem Effekt im Moment zu einer inkorrekten Beleuchtung kommt.

  Ressourcen:
  http://www.paulsprojects.net/tutorials/simplebump/simplebump.html
  http://facweb.cs.depaul.edu/JBrzezinski/341/BumpMappingNormalMapInPhotoshop.htm
  http://www.blacksmith-studios.dk/projects/downloads/tangent_matrix_derivation.php

- **Parallax Mapping (funktioniert noch nicht fehlerfrei)**

  Normal Mapping legte auch den Einsatz einer Heightmap für Parallax Mapping nahe. Leider gibt es aufgrund der bisher fehlerhaften Tangenten auch hier keine völlig korrekte Darstellung. Als Referenzen dienten großteils die oben genannten Papers/Links.

3. Verwendete Libraries

Folgende zusätzliche Libraries werden verwendet:

AGEIA PhysX
www.ageia.com

Autodesk FBX
www.autodesk.com/fbx

DEVIL
openil.sourceforge.net

fmod
www.fmod.org

AGEIA PhysX

PhysX übernimmt die Kollisionserkennung und die Physik des Gleiters bzw. dessen zugrundeliegendes Fahrwerksmodell.

Im Sourcecode gibt es zwei Files („Stream.h", „UserData.h" bzw. *.cpp), welche für das generieren von sogenannten „actors" (das sind Akteure, die innerhalb einer PhysX Szene miteinander interagieren) nötig sind. Diese sind NICHT von uns implementiert worden – aber derzeit für die Kompilierung nötig.

Des weiteren findet sich noch ein File „DrawObjects.cpp", welches für das Rendering von PhysX-Akteuren nötig ist. *DIESE FUNKTIONEN WERDEN LEDIGLICH FÜR DEBUG-ZWECKE EINGESETZT* (und zwar für das Rendering des im Spiel unsichtbaren Fahrwerks) und haben mit dem eigentlichen Rendering von Graviton nichts zutun.

Autodesk FBX

Diese Library ermöglicht uns den Zugriff auf die Vertex-/Index-/Texturkoordinaten unserer modellierten 3D-Objekte, welche wir extrahieren und in unsere eigenen Klassen integrieren. Hier wurden außer dem Laden von Vertex-Normals und Texturkoordinaten ausschließlich Funktionen benutzt, die schon im Repetitorium empfohlen wurden.

DEVIL

Wird für das Laden von Texturen genutzt.

fmod

Verwenden wir für das laden/modulieren/abspielen von Musik und Soundeffekten.