



Source Code Übersicht

/meshes	Modelldaten und deren Texturen. Die Mesh-Vertexdaten liegen als OBJ Files vor, deren Materialeigenschaften als MTL Files.
/shader	Vertex- und Fragmentshader
/sounds	Sound Files
/src	Das eigentliche Spiel. In der Datei gameEngine.h werden die Includes gesetzt.
/src/Actors	Models als OOP-Objekte definiert. Im Subordner „Decorations“ befinden sich die statischen Dekorationsobjekte, im Subordner „Vehicles“ die Autos.
/src/gameEngine	Rendering- Engine Implementierung
/src/ gameEngine /items	Verhalten der Spielrelevanten Items (Rockets, EpicMode, Mine, Pickupmarker)
/src/ gameEngine /levelTypes	Implementierung von verschiedenen Spiellogiken bzw. Spielszenarien
/src/ gameEngine /resourcehandler	Ressourcenmanagment. Die MeshHandler Klasse dien zum Laden der einzelnen Meshes, die Facegroup-Klasse beschreibt eine Gruppe von Faces eines Objektes, welche dieselben Materialeigenschaften bzw. Textur haben.
/src/ gameEngine /physics	Berechnung der Explosionen, Explosionsradien, Anwendung von Kräften auf beeinflusste Objekte
/src/ gameEngine /particleSystem	Partikelsystem bestehend aus Rauch- und Explosionseffekten
/src/ gameEngine /utils	Helper-Klassen zur Erledigung von einfachen Aufgaben, zB Logger, JoypadInput, FileHandler.
/src/ gameEngine /vehicle	Vehicle Klasse und einige Helper-Klassen für die Vehicle Klasse benötigt. Dazu noch ein eigener Kamera-Typ (VehicleCam) der speziell für die Anzeige von Fahrzeugen gedacht ist.

Gameplay

Bei unserem Spiel treten 2 Spieler im Splitscreen-Mode gegeneinander an. Beide steuern jeweils ein Auto welches sich in einer Arena befindet. Ziel des Spieles ist es, den Gegner sooft aus der Arena zu schießen, bis dieser keine Lebenspunkte mehr hat. Um dies zu erreichen behilft man sich den Items welche man beim Überfahren eines Pickup-Markers erhält.

Es gibt 3 verschiedene Items:

- Rakete (1 Schuss)
- Epic Mode (Spieler verwandelt sich in eine große Rakete, und muss den anderen Spieler rammen. Der Modus endet nach 10 Sekunden, wenn ein Objekt gerammt wurde oder man aus der Arena fliegt. In diesem Modus beschleunigt man automatisch, bremsen ist nicht möglich)
- Mine (Einfache Mine welche man ablegen kann. Sie sieht einem Pickupmarker zum verwechseln ähnlich)

Spezialeffekte

Motion-Blur

Der Motion Blur Effekt wird im Epic Mode verwendet, um dem Spieler ein Geschwindigkeitsgefühl zu vermitteln. Gleichzeitig verwirrt es den anderen Spieler indem es ihm eine beschränkte Sicht bietet.

Um den Effekt zu implementieren wurde ein Texture-FBO verwendet, in welches die Szene parallel gezeichnet wird. Die Textur wird dann auf einem Quad angebracht, welches sich (als HUD Objekt) genau „vor der Szene“ befindet. Im Vertex-Shader wird die aktuelle Kameraposition mit der Vorherigen verglichen und daraus die Bewegungsrichtung ermittelt. Entlang dieses Bewegungsvektors wird schließlich im Fragment Shader ein Blur auf der FBO-Textur berechnet.

Quellen:

- http://http.developer.nvidia.com/GPUGems3/gpugems3_ch27.html
- http://www.songho.ca/opengl/gl_fbo.html

Per-Pixel-Lighting

Im Fragment Shader wurde eine per Pixel Beleuchtung implementiert welche alle Objekte betrifft, die keine Texturen haben. Als Ansatz wurde das Beleuchtungsmodell aus CG1 gewählt.

Quelle:

- <http://www.cg.tuwien.ac.at/courses/CG/textblaetter/09%20Licht%20und%20Schattierung.pdf>

Partikelsystem

Bei Explosionen werden Flammen und Raucheffekte erzeugt. Beim Abschuss einer Rakete sieht man den Rauch der „nachzieht“.

Hier wurde pro Partikel ein texturiertes Quad verwendet. Die Quads werden vom Renderer automatisch zur Kamera hin ausgerichtet (Billboards). Zur Laufzeit werden die Partikel an den Explosionsstellen eingefügt und blenden sich nach einer bestimmten Zeit langsam aus um den Effekt zu verstärken(mittels Alphawert-Veränderung).

Beim Abschießen einer Rakete wird auf dessen „Laufbahn“ alle paar Meter hinter der Rakete ein Rauchpartikel eingefügt um einen Raucheffect zu simulieren.

Quellen:

- <http://csserver.ucd.ie/~thinks/docs/ProceduralSmokeParticleSystem.pdf>
- <http://www.openframeworks.cc/forum/viewtopic.php?f=9&t=2639>

Objekte

Model-Loader

Der Model-Loader wurde selbst geschrieben und importiert OBJ Files und MTL Files. Er liest alle Vertices, Texturkoordinaten, Normals und Faces aus dem OBJ File und speichert diese in VBOs. Danach liest er das dazugehörige MTL-File, liest die Materialdaten und lädt die Texturen in den Graphikspeicher.

Objektquellen

Die Autos, die Würfel und der Tisch wurden mit dem Modellierungsprogramms Milkshape erzeugt. Die anderen Objekte wurden aus dem Google 3D Warehouse (<http://sketchup.google.com/3dwarehouse/>) heruntergeladen und mit dem Programm Google SketchUp in das OBJ Format exportiert.

Animierte Objekte

Das Auto ist ein animiertes Objekt. Die Räder drehen und rotieren sich, je nach Richtung bzw. Geschwindigkeit.

Beschleunigung der Sichtbarkeitsberechnung

View-Frustum-Culling wurde implementiert. In der Konsole wird angezeigt welche Objekte gecullt wurden.

Quelle:

- <http://www.lighthouse3d.com/opengl/viewfrustum/>

Transparenz-Effekte

Folgende Objekte werden mit Transparenz gerendert:

- Die einzelnen Partikel des Partikelsystems
- Die Text-Anzeigen in HUD
- Der Glastisch in der Arena
- Die Scheiben des Autos ($\alpha = 0,5$)

Experimentieren mit OpenGL

Alle hier geforderten Punkte wurden implementiert.

Steuerung

Player 1 steuert mit den Pfeiltasten am Keyboard, und aktiviert das Item mit der Leertaste. Player 2 steuert mit WASD und schießt mit Left Shift. Zusätzlich kann man stattdessen mit USB-Joypads spielen. Dazu muss in der Datei „skof.conf“ die Tastenbelegung geändert werden. Die Steuerungsanleitung kann im Spiel mit der Taste F1 aufgerufen werden.

Verwendete Libraries

GLFW	Grafikausgabe, Window-Handling und Eingabeerkennung	http://www.glfw.org/
GLEW	Extension Loading für OpenGL	http://glew.sourceforge.net/
GLM	Wird verwendet, um mathematischen Herausforderungen nicht auf den Grund gehen zu müssen	http://glm.g-truc.net/
Bullet	Physik-Engine	http://www.bulletphysics.com/
FreeImage	Laden der Textur-Bilddateien (JPG, PNG, TGA)	http://freeimage.sourceforge.net
OpenAL / ALUT	Einbinden der Soundeffekte und Hintergrundmusik	http://connect.creativelabs.com/openal/