High Speed Racer (Thomas Appler & Christian Zartl)

Anmerkung: Lautsprecher einschalten!

Programmstart

Ohne Parameter entspricht Fullscreen mit 1920x1080.
Alternativ das Programm mit 3 Parametern starten:
„HighSpeedRacer.exe WIDTH HEIGHT FULLSCREEN“
WIDTH/HEIGH = Pixelangabe in Integer
FULLSCREEN = 0 für nein, 1 für ja

Steuerung

Im Menü:

Q -> Beenden

Space -> Start eines neuen Spiels

Im Spiel:

Esc -> Zurück zum Hauptmenü

Pfeiltasten Links/Rechts -> Navigation des Schiffs

Pfeiltaste Rauf -> Beschleunigen

Pfeiltaste Runter -> Bremsen

Space -> Aktiviert boost (Nur wenn Boost verfügbar ist. Siehe links unten)

F1 -> Zeigt die Hilfe an

F2 -> De/Aktiviert die Framerate Anzeige

F3 -> De/Aktiviert die Wire-Frame Anzeige

F4 -> Ändert Texture Sampling

F5 -> Ändert MipMapping

ACHTUNG: F4/F5 wird grafisch erst nach 3 Streckenabschnitten Sichtbar. Aufgrund des Generators werden bestehende Texturen nicht laufend vollständig neu erstellt.

F7 -> Hintergrundmusik Pause/Play

F8 -> Aktiviert/Deaktiver Viewfrustum-Culling

F9 -> De/Aktiviert die Transparenz

Entwicklungsstatus / Effekte

- Light

Es gibt eine direktionale Lichtquelle im Spiel. Diese beleuchtet die ganze Scene. Bei jedem Renderschritt wird die aktuelle Richtung des Lichts an den Shader weitergegeben, um die Schattierung berechnen zu können. Es wurde das Blinn-Phong Schattierungsmodell implementiert.

- Modeling & Object Loader

Für die Modellierung unserer Objekte wird Blender verwendet. Die exportierten Object Dateien werden im Spiel dann mit Hilfe von Assimp geladen. Auch werden Normalen und UV Koordinaten mit exportiert und in das Spiel geladen. (Siehe zum Bsp. Affe)

- Texturen

Bilder für Texturen werden mit der Library SOIL geladen und an die GPU weitergegeben. Wie erwähnt werden mit Blender auch gleich UV Koordinaten exportiert, weswegen man sich im Code nicht darum kümmern muss. Außerdem werden die Texturen nur einmal geladen und anschließend gecached, weshalb das Umschalten von Nearest-neigbour, Bilinear with mipmaps & Trilinear erst nach 3 Streckenabschnitten sichtbar wird.

- Skybox

Siehe Himmel/Hintergrund.

- Musik

Die Hintergrundmusik bzw. Effekte wurden mittels FMOD umgesetzt.

- Shadow Mapping

Im Spiel gibt es eine direktionale Lichtquelle. Diese wird in jedem Update Schritt an die Kameraposition gebunden und etwas in Sichtrichtung verschoben. Im Draw Schritt wird dann aus Perspektive des Lichts die gesamte Scene in eine Depth Textur gerendet, die dann für die Schattenberechnung heranzgezogen wird (Wie im Tutorial).

- Motion Blur

Vor jedem Draw Schritt wir die gesamte Scene in eine Textur gerendert. Anschließend wird beim finalen Rendering im Shader die mit hilfe der alten MVP Matrix (vom Vorschritt) die "Pixelgeschwindigkeit" berechnet. Mit diesem Wert wird dann entlang des "Pixelvektors" aus der Scenen-Textur das aktuelle Fragmet im Fragmentshader geblured.

- GPU Partikel System

Das System wurde mit VAOs und dem Transform Feedback realisiert. Für jedes Partkel wird die Position und die verbleibende Lebensdauer gespiechert. Zusätzlich gibt es noch eine fixe Position (definiert im Shader), zu welcher siche die Partikel in jedem Updateschritt etwas bewegen. Auch wird die Lebensdauer veringert. Umso kleiner diese ist, umso roter (dunkelblauer) und transparenter wird das Partikel.

- Collision & Physics

Für die physikalische Simulation und die Kollisionserkennung wurde Bullet verwendet. Das Gefährt ist mit dem Bullet Vehicle Feature realisiert. Als Collision Bounding Box kann sowohl eine selbstdefinierte Box sein oder das geladene obj-Modell von Blender.

Collision:

- Affe & Rad: Reduziert die Panzerung.

- Würfel Schwarz: Erhöht Boost

- Würfel Bunt/Textur: Erhöht die Panzerung.

Gravity:

Siehe Sprungschanzen. Abhängig von der Geschwindigkeit ist die Flugdistanz.

- Map/Object Generator:

Es gibt einen Scenengraphen (SceneNode) welcher die verschieden Objekte enthält. Sobald der Racer mehr als ca. 1,5 Streckenabschnitte zurückgelegt hat, wird das alte SceneNode (Streckenteil & Objekte) gelöscht und es wird eine Querwand aktiviert, was ein Zurückfahren unterbindet. Dies dient zur Steigerung der Performance. Es gibt immer nur 3 Streckenabschnitte im SceneNode . Dies ist im Prinzip eine simple Implementierung von View Frustum Culling, zumindest "nach hinten". Die Strecke verläuft einfachheitshalber nur in Richtung einer Z- und einer X-Achse. Zusätzlich gibt es ein einfaches deaktivierbares Viewfrustum-Culling, wo die nicht sichbaren objekte nicht gerendert werden.

- Schriften

Im Hauptmenü sind die Texte im Blender modelliert und importiert. Im Spiel selbst wurde die Textanzeige mittels Freetype umgesetzt.