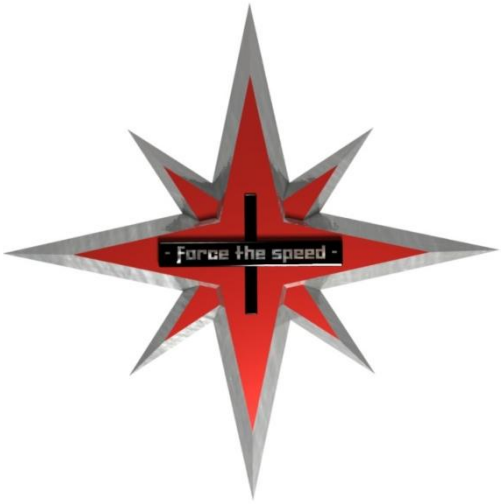
G R



3. Abgabe

aus

CG2 LU

TU Wien, SS08

~~Michael May~~	~~0126164~~	~~534~~ [1)]
(Christian Niederreiter	0726258	932) [2)]
Günther Voglsam	9955844	521

[1)] Hat die Gruppe kurz vor der 2.Abgabe verlassen.

[2)] Hat zum Projekt nur marginal beigetragen.

Allgemeines

Wie einige Tutoren bereits Bescheid wissen, ist diese Arbeit zu einem Einzel-Projekt geworden.

Der C++-Code stammt annähernd komplett (bis auf die Vektor-Klassen) aus meiner (Günther Voglsam) „Feder“. Der Code ist mehr gehackt als sauber implementiert, da u.a. auch am Anfang des Projekts die Planungs-Phase zugunsten der grandiosen Idee u.a. der Plattformunabhängigkeit ausgelassen wurde.

Dies ist eine erneute Abgabe. Seit der letzten Version wurden verbessert/hinzugefügt:

- Partikel-System
- Keyframe-animierte Objekte
- Rendermodi immediate/Display-List/VBO
- einfache Transparenz
- selbst aufgenommenes In-Game-Audio
- erweitertes Gameplay (Extras, Extras-Respawn, Force-Field-Reloading, kaum noch Bouncen zw. den Track-Wänden, usw.)

Gameplay

G-R ist ein „futuristischer Racer“.

Ziel war und ist es, entweder im Single-Player gegen die Zeit Runden zu drehen, oder gegen einen zweiten Spieler zu fahren.

Das Raumschiff bleibt dabei am Track durch Magneten „haften“. D.h. aber auch, wenn die Geschwindigkeit zu hoch ist kann die Fliehkraft überwiegen und man saust ins Weltall.

Es werden pro Spiel drei Runden gefahren, wenn man nicht vorher alle Leben aufbraucht. Um über „Löcher“ im Track zu springen muss man das Magnetfeld deaktivieren (mehr dazu siehe Abschnitt „Input“). Es wird dazu auch rechtzeitig eine Meldung angezeigt.

Die Steuerung erfolgt für beide Spieler über die Tastatur. Im Spectator-Modus kann man die Kamera mit der Maus bewegen.

Bewegte und animierte Objekte

Es bewegen sich die Raumschiffe in der Szene. Die Extras sind Keyframe-animierte Modelle.

Texturierte Objekte, Materialien

Alle Objekte sind texturiert.

Die Texturen werden mit DevIl geladen und die „texture-object“-Funktionalität wird verwendet.

In den 3DS-Models stehen grundsätzlich Material-Werte drinnen, allerdings schreibt der Blender-Exporter dafür scheinbar wirres Zeugs rein, weshalb die Materialien händisch gesetzt werden.

Einfache Beleuchtung

Die Beleuchtung der Szene erfolgt über eine direktionale Lichtquelle.

Kamera und Steuerung

Für die Steuerung für das/die Raumschiff(e) siehe Abschnitt „Input“.

Die Kamera-Bewegung ist mit Catmull-Rom-Splines interpoliert. Es werden pro n Frames neue Kamera-Punkte berechnet. N hängt dabei von der Geschwindigkeit ab. Die Kamera ist nie weiter weg vom Raumschiff als eine maximale Distanz.

Im Spectator-Modus kann man sich mit der Kamera frei in der Szene bewegen. Die Skybox wird in diesem Kamera-Modus nicht gerendert (Falls dies erwünscht ist, stellt das kein Problem dar).

Datenstrukturen und Implementierung

Es gibt folgende Namespaces:

- System:
  Systemspezifische Klassen wie Timer, Initialisierung u. dgl.
- Rendering:
  Verwalten des Renderings. In diesem Namespace sind Objekte wie Kamera, Licht, Menü, Szenengraph, Viewport und andere vorhanden.
- Physics:
  Einbinden der Ageia PhysX-Engine. Für jedes Raumschiff werden Properties in einer eigenen Klasse Ship gespeichert.
- GeometryPrimitive:
  Mathematische und geometrische Primitive wie Vektoren, Matrizen und Meshes.
- GameLogic:
  Verwalten der Gamelogik durch diverse Game-, Player- und Level-States.

- Data:

  Der Spiel-Content. Es werden zwei Modelformate, MD2 und 3DS, unterstützt. Beide sind von einer gemeinsamen abstrakten Basisklasse AModel abgeleitet. Des weiteren werden die Bounding-Spheres automatisch berechnet und die Materialien und Texturen verwaltet.
  Auch die Audio-Verwaltung findet hier statt. Audio wird derzeit nur über PlaySound() abgespielt.

Verwendete Libraries sind GLEW, OpenGLUT, DevIl, PhysX und lib3ds (http://lib3ds.sourceforge.net/).

Die gerenderten Models vom Track und das Raumschiff wurden von Michael May im Blender erstellt. Der Polygon-reduzierte Track, auf den getestet wird um das Schiff auf dem Track zu halten, wurde von Christian Niederreiter im Blender erstellt. Die Skybox von Günther Voglsam.

Spezialeffekte

Als Spezial-Effekt wurde ein Partikel-System für das Raumschiff implementiert.

Input

Player 1:

wasd == vor / links / zurück / rechts

e == Magneten abschalten (zum Springen)

q == Wiederbeleben

Player 2:

ijkl == vor / links / zurück / rechts

o == Magneten abschalten (zum Springen)

u == Wiederbeleben

Spectator-Mode:

m == Aktiviere Spectator-Modus

c == Kamera rücksetzen

wasd == Kamera bewegen

linke Maustaste == Kamera drehen

Sonst:

F1 == Hilfe -> siehe Hilfe für alle Commandos

F2 == Zeige FPS-Counter

F3 == Wechsel der Render-Modi Solid/Wire-Frame/Points

F6 == Wechsel der Render-Modi immediate/Display-List/VBO

F9 == Transparenz ein/aus

F11 == Textur ein/aus

b == Zeichnen der Bounding-Volumes

n == Zeichne Triangle-Normalvektoren