Flight Race Dokumentation

Submission 1:

Free movable camera: Es gibt eine Kamera, welche dem Flugzeug folgt. Es gibt drei Kameramodi, wovon zwei nicht vom Benutzer steuerbar sind und eine dritte welche frei um das Flugzeug bewegbar ist.
Implementierung: Es gibt eine Kamera Klasse welche zwei Funktionen implementiert: Eine update Funktion, welche User eingaben abfragt, und eine Funktion um die Position zu einem Objekt zu berechnen.

Moving objects: Es gibt zwei bewegende Objekte. Das Flugzeug und die Ringe, welche eingesammelt werden müssen. Das Flugzeug ist vom Benutzer mittels Tastatur oder Controller steuerbar. Die Ringe drehen sich um sich selbst.
Implementierung: Flugzeug: In der Update-Funktion werden User eingaben abgefragt und das Flugzeug dementsprechend bewegt und rotiert.
Ring: In der Update-Funktion wird der Ring um sich selbst rotiert.

Texture Mapping: Im Spiel gibt es zwei Objekte mit Texture Mapping: das Flugzeug und die Skybox.
Implementierung: Die Modelle und UV-Maps wurden mittels Blender erstellt und mithilfe von Assimp geladen. Die Texturen werden mittels DevIL geladen. Diese werden an OpenGL übergeben und dann im Shader angezeigt.

Simple lighting and materials: In unserer Szene gibt es ein direktes Licht, welches die Sonne darstellt. Mittels des Blinn-Phong Beleuchtungsmodells wird daraus im Shader eine Schattierung mit Glanzpunkt berechnet. Die Materialien (Glanzstärke) werden mittels Assimp aus Blender importiert.

How and which objects were illuminated (description of light sources) or textured.

Wie: Mittels einer Sonne als Lichtquelle. Das Phong Beleuchtungsmodell berechnet daraus im Shader eine Schattierung mit Glanzpunkt.
Welche Objekte (beleuchtet): Die Landschaft, sowie das Flugzeug
Welche Objekte (Texturen): Das Flugzeug, sowie die Skybox

Submission 2:

Gameplay: Der Spieler steuert das Flugzeug und hat 3 Minuten Zeit durch Ringe zu fliegen. Fliegt er durch einen Ring bekommt er 100 Punkte. Der Ring wird daraufhin neu platziert. Man kann einen Booster einsammeln, damit das Flugzeug schneller fliegt. Kollidiert der Spieler mit der Landschaft, so ist das Spiel vorbei.

Animated Objects: Die Flügel am Flugzeug bewegen sich, je nachdem in welche Richtung man steuert. Das Booster-Item hat zwei Objekte, die um es kreisen.

View-Frustum Culling: Wurde mittels des Radar Approaches implementiert. Jedes Objekt hat eine Bounding Sphere mit einem fixen Radius. Solange die Bounding Sphere das View-Frustum schneidet, wird das Objekt gezeichnet.

Bernhard Bayer, 1228892
Florian Bayer, 1427466

Controls: Mittels Tastatur oder Controller wird das Flugzeug gesteuert.
Tastatur + Maus:
W/S: Hoch-/Runterziehen
A/D: Links-/Rechtsdrehen um die Achse des Flugzeugs
Q/E: Links-/Rechtsschwenken
Links Shift: Turbo
V: Umschalten der Kameramodi
Maus: Schwenken der Kamera in Kameramodus 3
Esc: Spiel beenden
F1: Help
F2: Frame Time on/off
F3: Wire Frame on/off
F4: Textur-Sampling-Quality: Nearest Neighbor/Bilinear
F5: Mip Mapping-Quality: Off/Nearest Neighbor/Linear
F8: Viewfrustum-Culling on/off
F9: Transparency on/off

Controller:
Linker Stick: Hoch-/Runterziehen und Links-/Rechtsdrehen um die Achse des Flugzeugs
Schultertasten (Xbox-Controller): Links-/Rechtsschwenken
Knopf 1 (Xbox-Controller: Knopf A): Turbo
Knopf 2 (Xbox-Controller: Knopf B): Umschalten der Kameramodi

Experimenting with OpenGL (Transparency, Texture Sampling, MipMapping Quality):
Transparency:

- Das Flugzeug hat eine Transparente Glaskuppel.
- Lens Flare zeichnet transparente Texturen.
- Der Winning-Screen benutzt ebenfalls Transparente Texturen

Complex Objects: Das Terrain, das Flugzeug, das Booster-Item und der Hut. Alle Objekte wurde mittels Blender erstellt.

Effects */ "Features" of the game.*

Level of Detail: Es gibt für manche Objekte mehrere Qualitätsstufen des Modells. Entfernt sich der Spieler vom Objekt, wird eine niedriger aufgelöste Version des Objekts angezeigt.
Folgende Objekte verwenden Level of Detail: Der Ring, das Boos-Item und der Hut

Shadowmapping with PCF: Mittels einer Shadowmap aus dem Blickpunkt der Sonne mit einer orthogonalen Projektion werden Tiefenwerte berechnet. Beim zeichnen der Objekte wird die Entfernung zur Lichtquelle mit der Shadowmap verglichen und dementsprechend schattiert.

Lens Flare: Mittels der Sonnenposition im Screenspace wird eine Linie durch den Bildschirmmittelpunkt gelegt. Auf dieser Linie werden vorgefertigte Texturen platziert.

Glow: Mittels Framebuffer werden beim zeichnen des Bildes zwei Texturen angelegt: eine für das Bild und eine für den Glow-Effekt. Die Textur für den Glow-Effekt wird danach mittels Gauß-Filter

Bernhard Bayer, 1228892
Florian Bayer, 1427466

geglättet und zum Hauptbild addiert.

Folgende Objekte verwenden Glow: Das Flugzeugttriebwerk und das Booster-Item

What additional libraries (eg for collision, object-loader, sound, ...) were used, including references (URL) (see restrictions)?

Derzeit werden folgende Libraries verwenden:

Assimp	(zum Objekte laden)
DevIL	(zum Texturen laden)
FreeType	(um Text darzustellen)
glew	(um OpenGL Libraries zu laden)
glfw	(um das Fenster darzustellen und Input abzufragen)
PhysX	(für Kollisionen)

Verwendete Tutorials:
FreeType: http://learnopengl.com/#!In-Practice/Text-Rendering

CubeMaps: http://learnopengl.com/#!Advanced-OpenGL/Cubemaps

Kamera: http://www.gamedev.net/page/resources/_/technical/game-programming/creating-a-useful-camera-class-r2160

Lens Flare: http://www.emagix.net/academic/item/lens-flares
http://www.gamedev.net/page/resources/_/technical/graphics-programming-and-theory/lens-flare-tutorial-r813

PhysX: Dokumentation

Glow: http://http.developer.nvidia.com/GPUGems/gpugems_ch21.html
http://xissburg.com/faster-gaussian-blur-in-glsl/

Bernhard Bayer, 1228892
Florian Bayer, 1427466