Dokumentation

Gameplay: Das Gameplay hat sich nicht geändert, man bewegt sich mit den Tasten WASD fort und mit der Leertaste springt man. Weiters gibt es nun Blöcke die durchsichtig werden und anschließend ganz verschwinden. Mit der TAB-Taste kommt man wie schon früher in den Debugmodus, hier kann mit den Tasten TFGH die Kamera ändern und mit IJKL das Licht bewegen. Hier sieht man dann den Lichteffekt.

Effekte: Ich habe die Effekte NormalMapping, ShadowMapping sowie Bloom implementiert. Ersteren sieht man leicht an der Haut des Heros. Sowie ganz leicht an den Blöcken. ShadowMapping ist schwer zu übersehen, kann aber im Debug modus überprüft werden. Und vom Bloomshader ist nur der Flammenball betroffen.

Animated Objects: Ich habe leider nicht so viele Objekte, welche von einem anderen Objekt abhängen können. ABER, ich habe eines … der Propeller, welcher am Hut des Players angebracht ist. Dieses bekommt zu Updatemethode zusätzlich die ModelMatrix vom Hero und die eigene ModelMatix wird mit der ModelMatrix vom Hero multipliziert.

FrustumCulling: Hier habe ich nicht die „komplizierte“ Methode implementiert. Weil es bei mir eh nur einen gewissen Raum gibt der beleuchtet ist habe ich die Würfel abhängig von der Distanz zum Hero oder zur Flamme gemacht.

Controls: siehe ersten Abatz.

Experimenting: Manche Blöcke können durchsichtig werden, sodass der Player, sofern er zu lange auf dem Objekt braucht, hinunter fällt. Alle Experimente können mit den F-Tasten, wie sie in der Beschreibung zu der Submission 2 stehen, getestet werden.

Features: Es gibt Von-Menschen-gesprochene-Sounds fürs Hüpfen, Fallen, Gehen und wenn der Flammenball zu nahe kommt.

Verwendete Librarys:

- ObjektLoader: Assimp
- Sound: irrKlang

Implemented Effects:

- NormalMapping
- ShadowMapping
- Bloom

Implementation of Effects:

NormalMapping

http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/. Die Tangents und Bitangent werden im Sceneobject::loadOBJ berechnet. Und anschließend im Shader verwendet

ShadowMapping

http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/. Dabei rendere ich die Tiefenpunkte in eine Textur mit einer 5x größeren Breite und Höhe. Diese übergebe ich im nächsten Renderschritt dem Basicshader. Bei einer Auflösung von 1024 funktioniert der Schatten ohne Probleme. Nur bei einer höheren Auflösung „verrutscht“ er langsam bis man ihn gar nicht mehr sieht. Ich kann mir dieses Problem leider gar nicht erklären.

Bloom

Bloom hab ich mit Hilfe des FBOs von ShadowMapping und euren Folien gelöst, obwohl er nicht wirklich gut aussieht. Hierfür habe ich 3 FBO Objekte verwendet.

Verwendete Programme:

- 3D-Object: Blender
- Sounds: Download oder SoundBooth