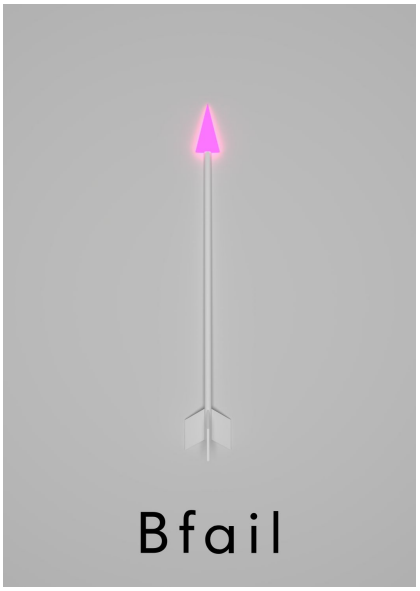
Bfail (G64) Dokumentation Submission 2

Daniel Ratzinger (1427247)
Aaron Zingerle (1527398)

Implementierung



- Gameplay

Der Spieler kann sich in der Szene mit der Maus umsehen. Er befindet sich im Spiel auf einer Kirche ohne Dach. Im Inneren irrt ein Roboter umher, der auf dem Kopf eine Kokosnuss trägt. Diese Kokosnuss wurde dem Spieler geklaut. Der Spieler kann nun mit seiner Sicht auf den Roboter zielen und einen Pfeil in seine Richtung schießen. Während der Pfeil fliegt kann er den Flug des Pfeiles noch mit ‘wasd’ korrigieren. Wenn der Spieler den Roboter mit den Pfeil trifft, verschwindet die Kokosnuss auf dem Kopf des Roboters. Nun ist die Kokosnuss wieder im rechtmäßigen Besitz des Spielers.

- Effects

- Shadow Mapping

Für das Shadow Mapping benutzen wir zwei Shader. Mit dem ersten generieren wir eine Depth map, die wir in einer Textur speichern. Die Depth map generieren wir aus der Sicht der Lichtquelle, die sich oberhalb des Spielbereichs befindet. Diese Map übergeben wir dann dem zweiten Shader, der mit Hilfe derer und einem Blinn-Phong, die Szenen-Elemente zeichnet.

- CPU Particles mit Instancing

Die CPU Particles generieren wir einmal am Anfang und zeichnen sie dann mit einem eigenen Shader. Hier verwenden wir Instancing, damit alle Particles zur gleichen Zeit gerendert werden (quasi das gleiche Model benutzen) um die Performance akzeptabel zu halten.

- Environment Mapping

Die Umgebung ‘füllen’ wir mit einer skybox, auf der wir eine cube-map Textur legen. In der Szene befinden sich ein paar Cubes die sich in einer schlangen-artigen Bewegung bewegen. Sie werden mit der benutzen cube-map ‘gesamplet’ und die Reflektion zur skybox gerendert. (Zudem wird die Skybox unabhängig von der Transformation der Kamera gerendert, heißt der Spieler kann nicht aus ihr hinausfliegen.)

- Animated Textures

In der Szene befindet sich ein Würfel, auf dem ein ‘Video’ auf alle 6 Planes ‘gestreamt’ wird. Dabei haben wir ein Video-File in ein .ppm File umgewandelt (mit ffmpeg). Das ist ein Bild-Format, das die Daten in Text-Form speichert und es auch ermöglicht mehrere Frames

eines Bildes (=Video) zu speichern. Durch diese Frames iterieren wir in update() und setzen die gezeichnete Textur neu.

- Animated Objects

Bei den Cubes, bei denen wir Environment Mapping durchführen, rotiert eine Kugel um einen Cube. Die Animation von dieser Kugel ist abhängig von der Position des einen Cubes.

- Frustum Culling

Wir haben versucht View-Frustum-Culling nach dem “Radar Approach” zu implementieren (Lighthouse3D Tutorial). Zur überprüfung, ob die Objekte im Frustum liegen, berechnen wir für jedes Mesh einen Minimum- und Maximum-Punkt. Daraus berechnen wir Bounding-Spheres für jedes Mesh und testen diese gegen das Frustum. Leider funktioniert die implementierte Lösung nicht, da wir nicht herausgefunden haben, was beim Sphere-Check nicht funktioniert.

- Controls

R - Physik/Kamera reset
F - Gegner reset
Q - Physikberechnung anhalten

F1 - Help
F2 - Frame Time on/off
F3 - Wire Frame on/off
F8 - Viewfrustum-Culling on/off (broken)

Features:

- der abzuschießende Roboter bewegt sich in der Kirche willkürlich umher
- wenn der Roboter getroffen wurde, ändert sich sein Model (es wird ausgewechselt, so dass er keine Kokosnuss mehr auf dem Kopf hat)
- die Physik des Pfeils wird mit Bullet simuliert (und mit derer wird die Kollision zwischen Roboter und Pfeil detektiert)
- die Physik-Welt kann zurückgesetzt werden (neu initialisiert)
- der Roboter kann resetted werden (Kokusnuss zurück auf seinen Kopf)
- die Kamera fliegt mit dem abgeschossenen Pfeil mit
- der Pfeil wird von der Position der Kamera in die Blickrichtung der Kamera geschossen (das hat viel Zeit gekostet, weil die Dokumentation von bullet schlecht ist und quaternionen ein Spaß sind)

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

Wir benutzen eine globale Lichtquelle zum Beleuchten der Szene.

Es werden alle Elemente damit beleuchtet, ausgenommen: skybox, animierten-texture-Cube, environment-mapped-Cubes;

Additional libraries (with url):

- Assimp
- Bullet

Which Effects are implemented:

- Shadow Mapping
- CPU Particles mit Instancing
- Environment Mapping
- Animated Textures

References for Effects etc.:

opengl:

- https://learnopengl.com/
- http://www.lighthouse3d.com/tutorials/view-frustum-culling/
- http://podricing.org/rendering-video-to-texture-in-opengl.html

zum Konvertieren des Videos zu .ppm

- https://ffmpeg.org/

assimp:

- https://learnopengl.com/

bullet:

- http://bulletphysics.org/mediawiki-1.5.8/index.php/Main_Page
- Foren :>

Tools for Model creation:

- Blender