## IMPLEMENTATION

### Gameplay

Der Spieler kann sich auf folgende Arten im Level bewegen: Laufen, Kriechen und über Objekte steigen (-> Stiegen laufen). Es gibt zwei „Game Values“: Energie (blau) und Furcht (rot). Die Energie sinkt beim Einsatz der Taschenlampe, die Furcht steigt in der Dunkelheit – zusätzlich gibt es Energiebälle, die die Energie wieder voll füllen und mit denen man haushalten muss, um durch das Level zu kommen, ohne dass die Furcht maximal wird.

### Complex Objects

Im Level sind mehrere komplexe Objekte wie Statuen, Stühle oder Tische vorhanden, die wie auch schon bei der ersten Submission mittels Assimp geladen werden.

### Animated Objects

Der Teddy wird mit dem Player mitbewegt und schwankt beim Gehen auf und ab. Zusätzlich bewegt er sich auf Grund von Sichtänderungen zum/aus dem Zentrum. Um den Teddy kreisen (falls vorhanden) die Energiebälle. Im Raum mit den Statuen schwingt ein Licht.

### View-Frustum-Culling

Aufgrund des Sehwinkels, der Position und weiteren Werten wird bei in jedem Frame das View Frustum aktualisiert.

Fürjedes Objekt wird überprüft, ob es im View Frustum liegt. Dieses wird eine Schittprüfung des Frustums mit den Bounding Boxen erledigt. Es wird nur geculled, wenn die Box komplett außerhalb liegt.

### Transparency (dead)

Zur Darstellung von Transparenz werden die Objekte danach geordnet, ob sie durchsichtig sind oder nicht. Die „massiven“ Objekte werden zuerst gezeichnet. Anschließend werden die transparenten Objekte gezeichnet. Hierbei werden aber die DepthWrites deaktiviert und nur die vorhandenen Werte im DepthBuffer für den Depth Test verwendet. Zusätzlich wird GL_BLEND aktiviert, die Blend Funktion angegeben und BackfaceCulling deaktiviert, da wir diese Faces jetzt ja sehen können.

### Experimentingwith OpenGL

VBO – wurden für Positions, Normals, UVs, Tangents und Bitangents von Objekten verwendet

VAO – wurden pro Objekt erstellt und kapseln oben genannte VBOs

FBO – wurden für das Shadow Mapping und den Ping-PongGaussBlur verwendet

Mip Mapping und Texture-Sampling-Quality – kann ein/ausgeschaltet werden; es werden für alle Texturen beim Laden der Texture Handle gespeichert; beim ändern der Settings wird über alle iteriert und die Texture Parameter angepasst

**STEUERUNG**

ESC – Beenden
LCTRL – Crouch
LMB – Licht benutzen
E – Energiebälle benutzen

**ILLUMINATION**

Die Szene wird von der Taschenlampe des Spielers beleuchtet. Zusätzlich gibt es im Raum mit den Statuen ein Spotlight. Beide werfen Schatten (VSM).Der Teddy wird mit einem Rim-Light beleuchtet.

**LIBARIES**

Assimp, FreeImage, FMOD Ex, PhysX

**MODELS**

Das Level, der Teddy sowie das UV-Mapping wurden von uns in Blender und Maya erstellt. Die restlichen Modelle stammen von TF3DM.com.

**EFFECTS**

**Variance Shadow Mapping**

Dazu wird zunächst der Depth- und der quadrierte Depth-Kanal der Szene aus Sicht der Lichtquelle in ein FBO gerendert (RG32F- und Depth-Attachment). Mit einem zweiten FBO (RG32F-Attachment) und einem Ping-Pong-Blur werden die Kanäle gauß-weichgezeichnet.
Zur Auswertung werden beim Rendern der Szene die Licht-Transformationsmatrix und die Shadow Map (Textur des ersten FBOs) an den Shader übergeben. Die Positionen werden mit der Licht-Transformationsmatrix multipliziert und werden als Texturkoordinaten für die Shadow Map verwendet. Auf Grund der Werte in der Shadow Map werden der Wert p, die Varianz und die Standardabweichung berechnet und daraus der Wert p_max errechnet. Der größere der p-Werte ist unser „Schattenfaktor“.

**Normal Mapping**

Das Normal Mapping kommt in machen Durchgängen und an den Decken zum Einsatz. Dazu wurde zunächst beim Model-Laden die Tangente berechnet, die dann im Vertex-Shader mittels Crossprodukt mit der Normale die Bitangente ergibt. Aus diesen drei Vektoren wurde die TBN-Matrix erzeugt und dem Fragment-Shader übergeben und dort zur Transformierung der Normalen verwendet.

**Bloom**

Der Bloom-Effekt ist in BloomFBO gekapselt. Diese beinhaltet vier FBOs: renderFBO, thresholdFBO, blurFBO, mixFBO. Die Szene wird normal in renderFBO gerendert. Aus dieser Textur werden alle Pixel ausgewählt, deren Helligkeit über einem bestimmten Threshold liegt. Die ausgewählten Pixel werden mit einem Gaußfilter geblurred und anschließend mit der Ausgangstextur mittels Screen Blending

zusammengemixt. Das Ergebnis wird dann auf ein Quad gemapped und in den Backbuffer gerendert. Sehen kann man den Effekt bei den Energiebällen.