Elemental Ball

Ziel

Der Ball muss ohne herunterzufallen das Tor mit dem Eisengitter berühren, die Druckplatten wechseln das Element des Balls. Um eine Elementtüre zu überwinden muss der Ball das selbe Element besitzen.

Steuerung

WSAD	= Bewegen
ESC	= Beenden
F3	= Hud
N	= Normal Mapping an/aus

Optional Gameplay

Collision detection

Der Spieler kollidiert mit Türen und dem Boden. Für die Implementierung würde die externe Libary PhysX verwendet.

Advanced Physics

Wenn der Spieler über Druckplatten rollt wird diese aktiviert, des weiteren kann der Ball bestimmte Objekte (die Bowlingpins) durch Kollision verschieben. Hierfür wurde ebenfalls PhysX verwendet.

Heads-Up Display

Das aktuelle Element wird in der linken oberen Ecke angezeigt. Hierfür wird die Model Matrix des Objektes das angezeigt werden soll mit der Kameramatrix berechnet so das es sich mitbewegt.

Object-Loader

Das Level mit Ausnahme des Balls wird mit Hilfe der Libary Asimp von einem obj-File geladen wobei in einer separaten Text-Datei definiert ist welches Objekt welches ist.

Effects

Shadowmap mit PCF

Mit Ausnahme des Hud Elements werfen alle Objekte einen Schatten. Hierfür wird zuerst die Szene von der Lichtquelle, bei uns ein paralleles Licht, auf eine Depthmap gerendert welcher dann beim eigentlichen rendern der Szene auf die Canvas verwendet wird um die Pixel zu bestimmen welche im Schatten liegen. Um grafische Fehler durch Shadowacne zu verhindern wurde in den Shader einen Bias für die depthmap und culling auf front-culling gesetzt damit der Bias nicht zu peter panning führt. Um die Schatten weicher zu machen wird bei der Berechnung auch alle umliegenden Texel berücksichtigt.

Glow/Bloom

Der Glow Effekt ist bei den Feuertüren und Platten gut zu sehen. Hierfür wird die Szene auf 2

Texturen gerendert, einmal alles und einmal nur die hellen Stellen. Die Textur mit dem hellen Stellen wird mittels Gaussian filter verwischt und mit der ersten Textur kombiniert.

Normal Mapping

Normal Mapping mit Tangents/Bitangents so wie im learnopengl.com tutorial beschrieben.

Particle System with Compute Shader

So wie im tuwel tutorial beschrieben. Der 4. parameter von dem velocity ssbo ist in unserer implementierung der Type des Particles (in Particlesystem.h können Sie sehen welcher int welcher typ ist). Es gibt Emitter Partikeln die im Compute Shader immer neue Partikeln erstellen. Sie werden mittels einem uniform immer auf der Player Position gehalten, sodass Elementspezifische partikeln vom Player ausgehen. zusätzlich gibt es auch Systeme die nur einmal “Explodieren”, wenn man Türen berührt durch die man nicht durch kann.

Procedural Textures

Procedural Textures für die verschiedenen Element texturen. Wasser und Feuer sind auch abhängig von der Zeit. Feuer ist mit einer 3d Nois funktion realisiert, Wasser sind 2 verschieden gescalede 2d (procedural) texturen die addiert werden. Es werden auch normals berechnet anhand der Diffuse Farbe.

Referenzen

Assimp:

[1]https://learnopengl.com/Model-Loading/Assimp abgerufen 25.April

PhysX:

[2]https://gameworksdocs.nvidia.com/PhysX/4.0/documentation/PhysXGuide/Manual/Index.html

[3]https://github.com/NVIDIAGameWorks/PhysX3.4/tree/master/PhysX_3.4/Snippets

ShadowMap:

[4]https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping

Bloom:

[5]https://learnopengl.com/Advanced-Lighting/Bloom

Normal Maps:

https://learnopengl.com/Advanced-Lighting/Normal-Mapping

Particle Systems:

https://tuwel.tuwien.ac.at/pluginfile.php/2415211/mod_page/content/32/GPU_Particles_SS18.pdf

Procedural Textures:

https://web.archive.org/web/20080724063449/http://freespace.virgin.net/hugo.elias/models/m_perlin.htm