

Hakumei

Iman Mujadzic (01650576)
Matthias Hürbe (00728266)

Gameplay

Sammle alle Schlüsseln im Level und laufe dann durch das Tor das sich öffnet.
Wenn man aus dem Level fällt wird die Map neu gestartet.
Das Tor befindet sich oben links von der Start Position.

3D Geometry

[LevelImporter.h/cpp Mesh.h/cpp SkinnedMesh.h/cpp Animation.h/cpp]
Assimp wird benutzt um StaticMeshes, SkeletonMeshes und Animationen zu importieren.

Intuitive Controls

[Player.cpp]
Standard Third-Person Character Controller

Intuitive Camera

[FreeflightCamera.h/cpp]
Es gibt im Moment 2 Modi für die Kamera. 1. Kamera folgt der Spielfigur und kann mit der Maus um die Figur rotiert werden. 2. Die Freefloating Kamera kann mit WASD durch den Level gesteuert werden.

Textures

[Textures.h; Textures2D.cpp; geometrytex.frag]
2D Textures & Cube Maps werden unterstützt und automatisch vom AssetManager geladen.

Moving Objects

[Player.cpp]
Die Kapsel die die Spielfigur darstellt wird kinematisch durch die Spielwelt bewegt. Dafür werden Beschleunigung und Verlangsamung simuliert. PhysX wird für Kollisionserkennung benutzt.

Adjustable Parameters

Siehe Assets/Engine.ini

Controls

Linker Mouseklick ins Window um die Mouse Movement zu capturen.

Key	Effekt
WASD	Character Movement
Mause	Rotate Camera
Spacebar	Jump
F1	Toggle Camera

Effects

Beleuchtung

Hakumei benutzt einen Deferred Renderer.

In den Gbuffer werden WorldPos, WorldNormal, Roughness, Depth, Metallic & Emissive gespeichert. Diese Texturen werden später als input benutzt um die Beleuchtung zu berechnen. [pointlight.frag directionallight.frag]

Shading Model: PBR Cook Torrance

Es gibt ein DirectionalLight und Punktlichtquellen. Jedesmal wenn die Figur springt wird eine neue Punktlichtquelle mit zufälliger Helligkeit und Farbe erstellt. Zusätzlich ist über der Figur noch ein Punktlicht platziert um die umgebung besser auszuleuchten.

DebugViews

F2 Shaded

F3 WorldNormal

F4 WorldPosition

F5 Albedo

F6 Metall

F7 Depth

Schatten

[directionallight.frag]

Für das directional Licht wird eine ShadowMap(DepthMap) gerendert und damit dann die Beleuchtung des Lichtes geblockt. PCF wird benutzt um das Aliasing zu minimieren.

Texturing&Shading

Siehe Mesh dass hinter der Insel versteckt ist. Benutze die FreeFlight Camera um dort hin zu kommen. (F1)

* Physically based shading

* NormalMapping [geometrytex.*]

* Specular map (Roughness) [geometrytex.*]

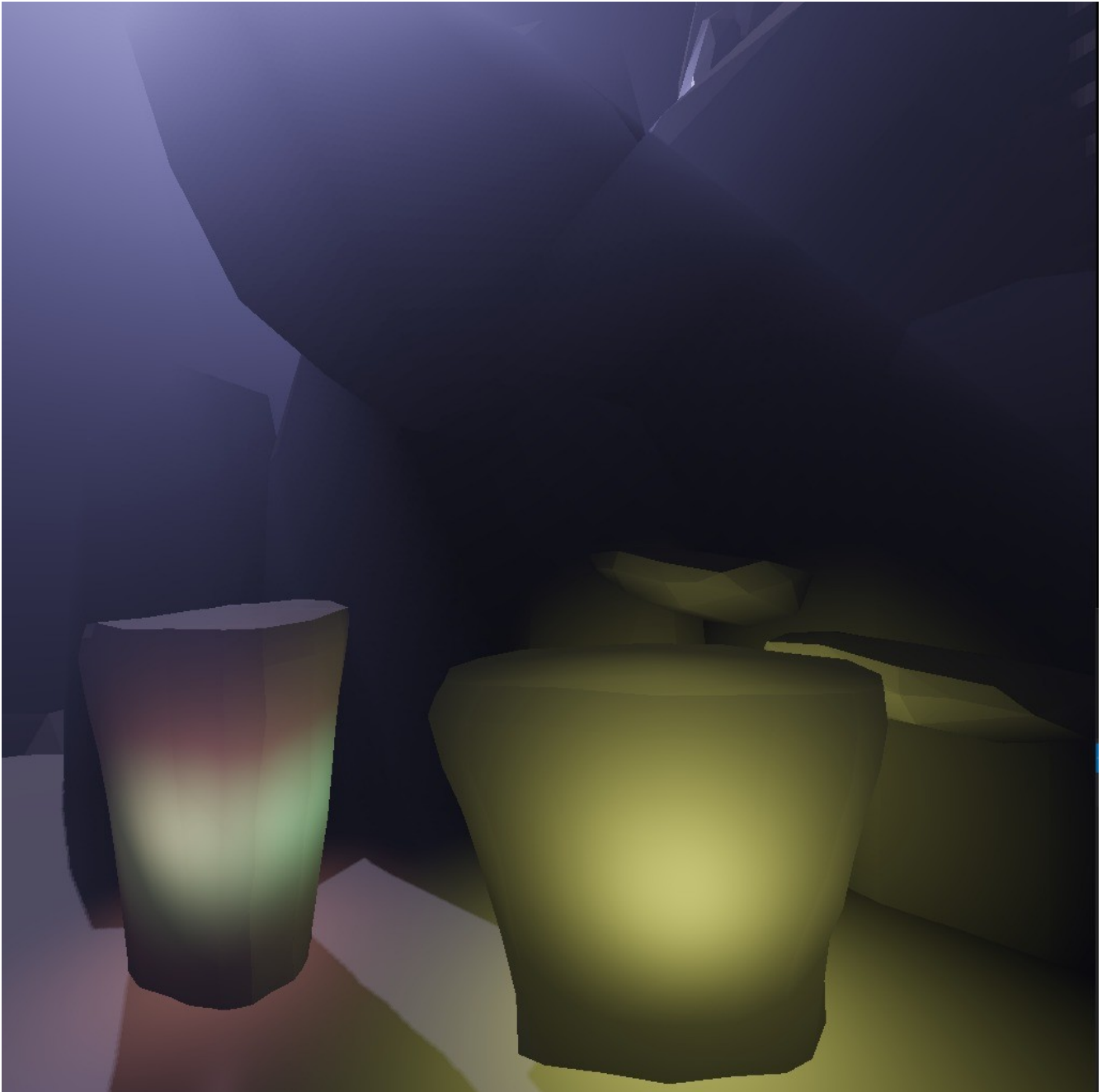
* Environment map

Translucency

[pointlight.frag]

Punktlichter die sich in einem Mesh befinden verursachen einen Translucency Effekt wenn dieses nicht metallic ist.

GDC2011 Frostbite3



Outline

Es werden in einen CustomDepthBuffer noch einmal alle Elemente gerendert die eine Outline haben sollen.

Vertex shader animation

[grass.vert/.frag]

Grass das sich im Wind bewegt..

WorldNormal wird auf (0 1 0) gesetzt damit es besser mit dem Boden blendet

GPU Vertex Skinning

[SkinnedMesh.cpp geometryanim.*]

Fuer den Character werden Animationen abgespiel.

Pro Vertex werden maximal 4 Bones unterstuetzt.

Environment map

[geometaball.frag]

Der Reflectionsvektor im Pixelshader wird benutzt um eine Cubemap zu sampeln.

Hierarchical Animation

PrimitiveComponents haengen zusammen. ZB.: Player besteht aus einem CapsuleComponent & ein Skeleton und ein PointLightComponent sind daran attached.

Ein PrimitiveComponent kann 0 bis 1 Parent haben & 0 bis N Children.

Blobby Objects using Raycasting

[geometaball.frag]

Es gibt MetaBalles die als Keys die man einsammeln muss in der Welt existieren. Mesh wird two sided gerendert.

Bloom

[BloomPass.cpp; RenderTagrget.h/cpp; fullscreen.vert; fullscreen.frag; brightness.frag; horizontalblur.frag ;verticalblur.frag]

Es wird das die gerenderte Scene geblurred und dann wieder mit dem Originalbild kombiniert.

1. Render der Scene in ein Render Target(SceneColor).
2. Filter weniger helle Bereiche des Bilde.
3. Horizontaler Gaussblur.
4. Vertikaler Gaussblur
5. Kombiniere das Ergebnis aus Schritt 4 mit dem originalen Szenenbild

ToneMapping

Um wieder ein LDR Bild aus dem HDR LightBuffer zu bekommen muessen die Dependencies

VolumetricLight

[directionallight.frag]

Raymarche durch die Scene und sample die ShadowMap an jedem punkt. Wenn dieser nicht im Schatten liegt dann benutze die Henyey-Greenstein phase function um die Beleuchtung an diesem pixel zu berechnen.

Alle Dependencies können im ThirdParty Ordner gefunden werden.

Name	Use Case
ArticleEnumClass-v2	Enables bitmask operator with c++11 enum classes
assimp	Mesh importer
Catch2	Unit Test Framework
Delegate	Impossibly fast delegate in C++11
Glad	OpenGL Binding
GLFW	Window Handling
glm	Math Library
jsoncpp	C++ JSON Library
PhysX	Physics Library
Premake	Build Tool
renderdoc	Graphics Debugger
spdlog	Logger

Quellen

<https://learnopengl.com/>

<http://ogldev.atspace.co.uk/index.html>

<https://www.udemy.com/graphics-with-modern-opengl>

<http://www.opengl-tutorial.org>

COMPUTER GRAPHICS PROGRAMMING IN OPENGL WITH C++

Spieleentwicklung – Mathematik, Physik, KI, Animation, Beleuchtung, GLSL Shader, Post Processing(German Edition)

OpenGL-Programming – Mathematik, GLSL Shader, Post Processing, Beleuchtung Animationen (German Edition)

Real-Time Rendering, Fourth Edition

ToneMapping

- * <http://filmicworlds.com/blog/filmic-tonemapping-operators/>
- * <https://forums.odforce.net/topic/25019-hable-and-aces-tonemapping/>
- * <https://knarkowicz.wordpress.com/2016/01/06/aces-filmic-tone-mapping-curve/>
- * <https://www.shadertoy.com/view/lslGzl>
- * <http://filmicworlds.com/blog/filmic-tonemapping-with-piecewise-power-curves/>
- * https://tuwel.tuwien.ac.at/pluginfile.php/1220878/mod_page/content/36/HDR_SS17.pdf

ray marching

- * <http://www.michaelwalczyk.com/blog/2017/5/25/ray-marching>
- * <http://jamie-wong.com/2016/07/15/ray-marching-signed-distance-functions/>
- * <https://computergraphics.stackexchange.com/questions/6308/why-does-this-gl-fragdepth-calculation-work>

volumetric light

- * <https://www.guerrilla-games.com/read/taking-killzone-shadow-fall-image-quality-into-the-next-generation-1>
- * <http://www.alexandre-pestana.com/volumetric-lights/>

lighting PBR

- * https://seblagarde.files.wordpress.com/2015/07/course_notes_moving_frostbite_to_pbr_v32.pdf
- * <https://imdoingitwrong.wordpress.com/2011/01/31/light-attenuation/>
- * https://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf
- * https://blog.selfshadow.com/publications/s2014-shading-course/hoffman/s2014_pbs_physics_math_slides.pdf

billboard

- * <https://www.geeks3d.com/20140807/billboarding-vertex-shader-glsl/>

scattering

- * <https://colinbarrebrisebois.com/2011/03/07/gdc-2011-approximating-translucency-for-a-fast-cheap-and-convincing-subsurface-scattering-look/>

Animation

- * <http://ogldev.atspace.co.uk/www/tutorial38/tutorial38.html>

Deferred Shading

- * <https://learnopengl.com/Advanced-Lighting/Deferred-Shading>

Translucency

- * <https://colinbarrebrisebois.com/2011/03/07/gdc-2011-approximating-translucency-for-a-fast-cheap-and-convincing-subsurface-scattering-look/>