

HIM - HERMANN IM MUSEUM [WORK IN PROGRESS]

Realtimographics Projektbeschreibung

Gruppe 13 Philip Krachler (1027173)
 Julia Gleichweit (1325844)



Szenenbeschreibung| Ausstellungsraum mit Brunnen

Hermann hat sich vor Kurzem eine neue Kamera angeschafft. Um diese auf Herz und Nieren zu testen möchte er sich die neue Kunstaussstellung im örtlichen Museum ansehen. Das Museum ist recht klein, jedoch bietet es eine exquisite Auswahl an Ausstellungsstücken. Das Herzstück der Ausstellung ist ein kleiner Brunnen am Ende der geführten Touren. Gespannt auf die Aufnahmequalität des sagenhaften Brunnens ist Hermann unvorsichtig und stolpert vor dem Brunnen über seine eigenen Füße. Unglücklicherweise fällt dabei seine neue Kamera in den Brunnen. Hermann ist todunglücklich über diesen Vorfall.

Anmerkung zum Ausführen: Das Laden der Szene kann etwas Zeit in Anspruch nehmen, daher wird um etwas Geduld gebeten (ca. 10-15 Sekunden). In der Konsole kann dies mitverfolgt werden.

Derzeitiger Stand|

Eine sehr simple Rendering Engine liegt vor, die:

- Modelle und Texturen laden kann (Modelle wurden mittels Blender Software erstellt)
- Phong-Beleuchtungsmodell
- Kontrollierbare Kamera und Fähigkeit einem Pfad automatisch zu folgen
- Anzeige nachfolgender Effekte

Auf den Podesten befinden sich unterschiedliche, texturierte Ausstellungsstücke. Am Ende des Raumes befindet sich der Brunnen mit Brunnenfigur. Während des Ladens der einzelnen Modelle (`ObjLoader.cpp`) wird sichergestellt, dass Texturen mit demselben Pfad bzw. Namen nur einmal geladen werden.

Der Raum ist über ein Punktlicht erleuchtet. Dieses befindet sich an der Decke des Raumes in der Nähe des Podestes mit dem Dreieck.

Beim Laden der Objekte werden die BoundingBoxes erstellt, um die Performance mittels Frustum Culling zu verbessern. Dies ist angelehnt an "Fast Extraction of Viewing Frustum Planes from the WorldView-Projection Matrix" von Gil Gribb und Klaus Hartmann. (<http://gamedevs.org/uploads/fast-extraction-viewing-frustum-planes-from-world-view-projection-matrix.pdf>)

Nach Ende der automatischen Kamerafahrt kann weiterhin manuell durch die Szene navigiert werden. Die Kamera lässt sich mit den WASD-Tasten durch die Szene bewegen, über die Maus kann die Kamera rotiert werden. Über die R-Taste lässt sich die Kamera wieder zum Anfang des Ganges zurückgesetzt werden mit Blick zum Start des Pfades.

Effekte

Nachfolgende Effekte wurden unter Bezugnahme der angeführten Referenzen implementiert und zusammengefügt (z.B. Schatten und Normal Mapping oder Schatten und Texturierung) entsprechend der Szene und Geräte angepasst (einige Kommandos aus den Referenzen haben nicht funktioniert).

Normal Mapping

Verwendete Referenz:

<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/>

Normalen, Tangenten und Bitangenten werden aus den Normalmaps gelesen. Assimp kann für die Modelle diese auch selbst berechnen (aiProcess_CalcTangentSpace), wobei dies aber nicht für jedes Modell/Datenformat gewährleistet ist oder man berechnet sie selbst über die Normalen.

Normal Mapping ist sowohl auf dem Boden als auch den Wänden, Podesten und Säulen vorhanden. Der Kamerapfad läuft an mindestens einem der genannten vorbei, damit der Effekt betrachtet werden kann.

Omnidirectional Shadow Maps

Verwendete Referenzen:

- <http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping> ,
- <https://learnopengl.com/#!Advanced-Lighting/Shadows/Point-Shadows>
- bzw. Folien vom Repetitorium
https://www.cg.tuwien.ac.at/courses/Realtime/repetitorium/rtr_rep_2016_omni_shadow_mapping

Der Algorithmus verläuft ähnlich zu traditionellem ShadowMapping. Die Depthmap wird aus der Perspektive des Lichts erzeugt. Basierend auf der aktuellen Fragmentposition und dem gespeicherten Tiefenwert wird entschieden, ob sich das Fragment im Schatten befindet oder nicht. Die Depthmap ist dabei jedoch eine CubeMap wo die Szene aus allen umgebenden Richtungen eines Punktlichts gerendert wird.

Anstelle von 6 Renderpasses wird über einen GeometryShader und den entsprechenden Matrizen die Geometrie in den jeweiligen Lightspace zu bringen.

Zur Verminderung von Akne, Peter Panning, etc. wurde percentage closer filtering verwendet. Die Schatten sind an den Wänden gut sichtbar an den Säulen oder auch den Podesten.


Refraction -

Verwendete Referenzen:

https://developer.nvidia.com/gpugems/GPUGems2/gpugems2_chapter19.html

Dieser Effekt ist am ersten Podest links auf eine flaschenförmige Figur angewandt, sowie an der dreieckigen Figur gegenüber des Brunnens.

Für diesen Effekt wird die Szenengeometrie ohne refractive models in eine Textur gerendert. Diese wird anschließend per glsl refract Funktion im shader so verzerrt, dass die Projektion der Szenentextur pro face verzerrt wird. Wie bei der reflect funktion nimmt die refract funktion einen Normalvektor als 2ten Richtungsvektor an. Dieser wird nicht von den face



Normals genommen, sondern wie bei Normalmapping einen Vektor aus einer normalmap. Dies führt statt einer flachen Verzerrung pro face zu einer strukturellen Verzerrung.

WaterShading

Watershading ist angelehnt an Refraction, mit ein bisschen Eigenerfindung um die Textur und das Mesh zu animieren. Es wird im Prinzip jeweils im vertexshader die Position.y und die UVs mit einem zeitabhängigen Offset verschoben. Dieser Effekt ist am Ende der automatischen Kamerafahrt direkt im Brunnen schön zu sehen.

Libraries

- GLEW - <http://glew.sourceforge.net/>
- GLFW - <http://www.glfw.org/>
- GLM - <http://glm.g-truc.net/0.9.8/index.html>
- ASSIMP - <http://assimp.sourceforge.net/>
- FreeImage - <http://freeimage.sourceforge.net/>

Verwendete Tutorials / Papers / Bücher (Referencelist)

Shaderloader

<https://learnopengl.com/#!Getting-started/Shaders>

Objectloader

<https://learnopengl.com/#!Model-Loading/Mesh>

<https://learnopengl.com/#!Model-Loading/Model>

Camera

<https://learnopengl.com/#!Getting-started/Camera>