Aufgabe 1: Getting Ready

(Neue) Techniken:

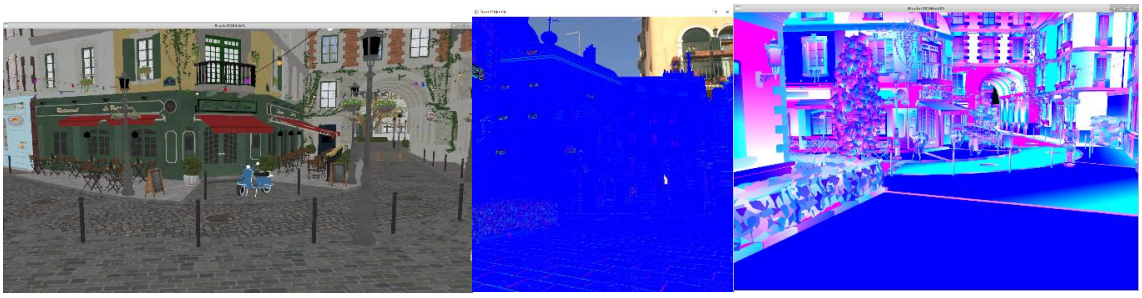
Modelloading, Texturkompression, Phong-Lightning, Text

Beschreibung:

In dieser Aufgabe sollen die Shader spezifischen Konzepte in OpenGL wiederholt werden und es soll ein grundlegendes Programm bzw. Struktur erstellt werden, die in folgenden Übungen wiederverwendet wird.
In dieser Übung soll die Amazon Lumberyard Bistro (Exterior) Szene [1] genutzt werden. Hierbei handelt es sich um eine komplexe Szene (ca. 3 Mio. Dreiecke), die zum Testen von Render-Engins verwendet werden kann.

Szene mit Richtungslichtquelle

ACHTUNG: Im Gegensatz zu bisherigen CG Übungen sind die Kernanforderung weiterhin Pflicht zur weiteren Teilnahme an der Übung, jedoch sind (fast) sämtliche Funktionalitätsanforderungen Basis für die folgenden Aufgaben und müssen u.U. zu späteren Zeitpunkten nachgeholt werden!



Von Links nach Rechts: Diffuse (Albedo) Texture, Normal Textur (Tangent Space), UV-Space

Kernanforderung:

- Laden & Anzeigen des Modells
- Laden & Anzeigen von DDS Texturen
- Es soll eine bewegbare Kamera geben

Weitere Funktionalitätsanforderungen:

- Es soll die Exterior FBX Szene des Amazon Lumberyard Bistros geladen und texturiert angezeigt werden. (Incl. Wireframe Toggle)
- Es soll für Testzwecke zwischen dem Modell und einem einfachen Grid mit veränderbarer Auflösung umgeschaltet werden können. (Nutzung Element Arrays)
- Per Tastendruck sollen die Shader neu kompiliert und geladen werden. Dies dient dazu, dass das Testen der Shader zur Laufzeit durchgeführt werden kann und nicht ständig die komplette Szene neu geladen werden muss.
  Es ist darauf zu achten, dass das Programm sich nicht schließt/abstürzt, sollte ein Fehler im GLSL Code vorliegen.
- Eine Kamera soll per Maus bewegt werden können, dafür soll die CTRL Taste dienen um die Maussteuerung zu aktivieren. (Alternativ kann auch ein Maus/Tastaturlayout ähnlich herkömmlicher 3D Applikationen gewählt werden z.B. Blender, Maya).
  Ein einfaches Tastatur Layout ist nicht ausreichend. Zur Vermeidung von Gimbal Lock können Rotations-Limits gesetzt werden. Die Kamera soll Rotiert und Verschoben werden können sowie Zoom besitzen.
- Passt beim einlesen der Texturen darauf auf, dass diese nicht Mehrfach eingelesen werden und es redundante OpenGL Textur-Objekte gibt. (In dem „naiven“ Demo Programm würden 4872 Texturen anstelle der ~400 Texturen eingelesen werden)
- Es soll eine Richtungslichtquelle geben, die die Szene beleuchtet. Die Lichtberechnung soll mit Hilfe des Phong Beleuchtungsmodells durchgeführt werden.
  Die Lichtquellen-Eigenschaften sollen im Host-Programm definiert sein und mittels Uniform-Variablen an den Shader übergeben werden.
- Die Hilfe soll als Text auf dem Bildschirm ausgegeben werden (Hierzu hilft die Text Demo) zusätzlich soll die Framerate auf dem Bildschirm ausgegeben werden. Beides soll per Tastendruck aktiviert/deaktiviert werden können.
- Der vorgegebene Model-Loader ist Node-Basiert bzgl. der Transformationen.
  Die Implementation in dem Demo-Programm sieht jedoch nur lokale Positionen vor und vernachlässigt Transformationen (oder Szenen Hierarchien).
  Der Model-Loader soll so angepasst werden, dass auch die Transformationen in der Szene durchgeführt werden. Ggf. Konkatenationen über Mehrere Ebenen.
  Ob dafür pro Mesh eine Modelmatrix gespeichert wird oder beim einlesen die Vertices (und Normalen!) verändert werden ist euch überlassen. Begründet die Entscheidung!
- Die Diffusen Texturen besitzen einen Alpha-Kanal. In dieser Aufgabe muss kein Blending umgesetzt werden! Es sollen jedoch Fragmente verworfen werden, die einen Alpha-Wert kleiner als .1 besitzen.
- Das Shading soll mittels Tastendruckes umschaltbar sein.
  Es soll folgende Möglichkeiten geben:
  - Phong-Shading mit Richtungslichtquelle (Beachtung Diffuse- & Specular-Map)
  - Geometrie-Normalen (als Farbe)

    - Textur-Koordinaten (als Farbe)
    - Diffuse Textur
    - Spekulare Textur
    - Normal-Textur (Tangent-Space)
- Um die Szene soll eine Skymap gerendert werden. (z.B. die san_giuseppe_bridge_4k.hdr Datei beim Modell) dafür soll ein zweites Shader Programm erstellt werden. Bei der Skymap ist es euch freigestellt, ob es sich um eine Cubemap oder eine Plattkarte handelt. (für spätere Aufgabe empfiehlt sich jedoch eine HDR Plattkarte) [4]

Tastatur-Belegung (Vorschlag):

Taste	Funktion
Esc	Beenden des Programms
F1	Toggle Fullscreen
F3	Toggle Wireframe
F10	Reload Shader
H/h	Toggle Hilfe
G/g	Toggle Grid/Szene
N/n	Grid-Auflösung erhöhen
M/m	Grid-Auflösung verringern
S/s	Shading Umschalten
STRG + LMB	Kamera Rotation
STRG + MMB	Pan Kamera
STRG + RMB (ggf. Maus Wheel)	Zoom Kamera (ggf. Verstzung entlang Orientierung)

Tipps:

1) Wir liefern eine Mathe-Bibliothek mit (cglm), die genutzt werden darf.
Die Dokumentation dieser Bibliothek kann hier gefunden werden:
https://cglm.readthedocs.io/en/latest/api.html
Die Funktionen in der Bibliothek sind durch SSE (Straming SIMD Extension) intern optimiert, wodurch die Nutzung extrem effizient ist.
Alternativ kann auch eine eigene Mathe-Bibliothek angelegt werden, bei der die Verantwortung aller internen Datenstruktur, Effizienz und Mathematische-Korrektheit dem Programmier obliegt.
Das nutzen einer Bibliothek entlastet nicht, sich mit der Mathematik (Matrix-Multiplikation…) auseinander zu setzen, in dem theoretischen Teil wird auf diesen Punkt eingegangen.

> 1.1) Die Bibliothek stellt Datentypen wie mat3, mat4, vec3 .. bereit, aufgrund des internen Speicherlayouts kann ein cast (float*) ausgeführt werden, um ein Pointer auf das 1. Element zu erhalten.

1.2) Folgende Funktionen der Bibliothek können Hilfreich sein: glm_scale, glm_translate, glm_rotate, glm_perspective (In der Abnahme wird erwartet, dass bei Verwendung dieser Funktionen, diese genau erklärt werden können (auf Mathematischer Ebene))

2) Der Assimp [2] kann genutzt werden zum Laden von Modeldatei Formaten.
Die Verwendung dieses wird in dem entsprechenden Model-Demo-Programm gezeigt.
Es werden jedoch Hierarchische Transformationen vernachlässigt. Die Node-Transformationen können über node->mTransformation abgerufen werden. Um diese in cglm mat4 Speicherlayout zu bringen muss die Matrix Transponiert werden!

3) DirectDraw Surface File Formate (.dds) ist ein von Microsoft eingeführtes Dateiformat welches zur Optimierung und Speicherungen von insb. Texturen vorgesehen ist. Insbesondere sind Mipmaps, Cubemaps und Volume Maps unterstützt. Weitere Infos können hier gefunden werden [3]

Die Texturen der Szene besitzen folgende Formate:
DXGI_FORMAT_BC1_UNORM (Base_Color)
DXGI_FORMAT_BC5_UNORM (Normal)
DXGI_FORMAT_BC3_UNORM (Specular)
OpenGL benötigt Extension(s) zum Laden der Dateien.
GL_EXT_texture
GL_EXT_texture_compression_latc
GL_EXT_texture_compression_rgtc
GL_EXT_texture_compression_s3tc
GL_EXT_texture_sRGB

[1]

```
@misc{ORCAAmazonBistro,
  title = {Amazon Lumberyard Bistro, Open Research Content Archive (ORCA)},
  author = {Amazon Lumberyard},
  year = {2017},
  month = {July},
  note = {\small \texttt{http://developer.nvidia.com/orca/amazon-lumberyard-bistro}},
  url = {http://developer.nvidia.com/orca/amazon-lumberyard-bistro}
}
```

[2] http://www.assimp.org/

[3] https://docs.microsoft.com/de-de/windows/win32/direct3ddds/dds-header
https://www.oldunreal.com/editing/s3tc/ARB_texture_compression.pdf

[4] https://hdrihaven.com/