Aufgabe 4: Shadow Maps

(Neue) Techniken:

Shadows, Depth Reconstruction, Light Space, ViewSpace

Beschreibung:

In dieser Aufgabe soll die Technik Shadow Maps implementiert werden. Dabei sollt Ihr mit Hilfe des aus Ausgabe 3 bekannten G-Buffers die bereits gerenderten Szene so verarbeiten, dass eine Textur entsteht pro Lichtquelle, die als zur Schatten Rekonstruktion verwendet werden kann.

Für die Bearbeitung der Aufgabe sollte OpenGL 4.3 und GLSL 4.30 verwendet werden.

Kernanforderung:

- Erstellung mind. einer korrekten Shadow Map.
- Anwenden der Shadow Map bei der Beleuchtungsberechnung und vergleich der Tiefe

Weitere Funktionalitätsanforderungen:

- Erstellung einer Shadow Map für Richtungslichtquellen
- Erstellung von Shadow Maps für die die Punktlichtquellen
    - Ggf. kann die Schattenberechnung für wenig intensive Lichtquellen ausgeschaltet werden
- Pre Rendering der Shadow Maps in der Initialisierungsphase (da die Szene statisch ist), für dynamische Objekte, kann die vorgerenderte Shadow Map als „Clear Buffer“ genutzt werden
- Effizientes Rendering der Schatten Map durch Geometry Shader Switch
- Das Anwenden der Schatten-Map soll per Tastendruck umschaltbar sein
- Percentage-Closer Filtering soll verwendet werden, um die zackigen Kanten des Schattens zu vermeiden. Dies soll per Tastendruck umschaltbar sein.
- Zum Auslesen der Positionswerte, wird nun nicht mehr eine eigene Positionstextur im GBuffer verwendet, sondern eine Rücktransformation des Tiefenwertes durchgeführt. Dadurch kann die Position anhand einer Komponente (GL_DEPTH) gespeichert werden (anstelle von dreien (GL_RGB)). (Zu Debug-Zwecken, kann die Position weiterhin im GBuffer-Pass auf eine eigene Textur geschrieben werden (Anzeige des GBuffers). Diese soll aber im weiteren Verlauf nicht mehr in den weiteren Shadern (Ermittlung: Fragment-Position; z.B. DirLightShader, PointLightShader) benutzt werden. ACHTUNG: in diesem Fall muss für den Tiefen-(/Stencil-Buffer) eine Textur und nicht mehr ein RenderBuffer (A3) verwendet werden.

Tastatur-Belegung (Vorschlag):

Taste	Funktion
Esc	Beenden des Programms
F1	Toggle Fullscreen
F2 [Optional]	Toggle Tangetspace
F3 [Optional]	Toggle Wireframe
F10	Reload Shader

1	Toggle Normalmapping
2	Toggle Parallaxmapping
3	Toggle Tessellation
4	Toggle SSAO
5	Toggle Schatten
H/h	Toggle Hilfe
F/f	Goggle FPS
G/g	Toggle Grid/Szene
D/d	Toggle Directional Light Quelle
L/l	Anzeige der Lichtquellen
N/n	Grid-Auflösung erhöhen
M/m	Grid-Auflösung verringern
Q/q	Gamma-Wert erhöhen
W/w	Gamma-Wert verringern
E/e	Exposure erhöhen
R/r	Exposure verringern
B/b	Toggle Percentage-Closer Filtering
U/u	Der Versatz der Vertices entlang der Normale wird erhöht.
I/i	Der Versatz der Vertices entlang der Normale wird verringert.
P/p	Zeitabhängige Animation pausieren
S/s	Umschalten zwischen (Debug) Kachelansicht oder Finaler Szene
STRG + LMB	Kamera Rotation
STRG + MMB	Pan Kamera
STRG + RMB (ggf. Maus Wheel)	Zoom Kamera (ggf. Versetzung entlang Orientierung)

Shader (Vorschlag):

Shader (Komponenten)	Beschreibung
PointShadowShader *(Vertex & Geometry & Fragment)*	Berechnung der Shadow Map bei Punktlichtquellen
DirShadowShader *(Vertex & Fragment)*	Berechnung der Shadow Map bei Richtungslichtquellen
GBufferShader *(Vertex & TessellationControl & TessellationEvaluate & Fragment)* (Aufg 3)	Rendert das Modell bzw. das Grid (die Geometrie & Material Informationen) in den G-Buffer (MRT). Ggf. mit Tessellation & Displacement. *

DirLightShader *(Vertex & Fragment)* (Aufg 3)	Directional Light-Beleuchtung *(Mit Schattenmap)*
PointLightShader *(Vertex & Fragment) (Aufg 3)*	Point Light-Beleuchtung *(Mit Schattenmap)*
NullShader *(Vertex & Fragment)* (Aufg 3)	Für den Stencil Test
SkyShader *(Vertex & Fragment)* (Aufg 1)	Zeichnet die Skymap im Hintergrund
BlurShader *(Vertex & Fragment)* (Aufg 3)	Shader, für den Blureffekt (mit advanced 2-Pass Filter; es reicht ein Shader mit einer entsprechenden Abzweigung)
PostProcessShader *(Vertex & Fragment)* (Aufg 3)	Shader, dient zum Zusammenführen der Ausgabe-Textur mit der Textur der Punktlichtquellen, die mittels Blur (mehrfach) gefiltert wurde sowie der Gamma-Korrektur und dem Tone-Mapping
TangentShader *(Vertex & Geometrie & Fragment)* (Aufg 2) *[optional]*	Zeichnet die 3 Achsen des Tangent Spaces, es ist darauf zu beachten, dieselbe Berechnung wie beim ModelShader zu nutzen.
TextShader *(Vertex & Fragment)* (Aufg 1)	Kann Text auf dem Display ausgeben
lightVisShader *(Vertex & Fragment)* (Aufg 3) *[optional]*	Zeichnet die Punkt-Lichtquelle als Primitive Geometrie (Cube oder Sphere)

* Hier findet das Normalmapping und Parallaxmapping statt.

Tipps:

1) Geht nach Bearbeitung der Aufgabe noch einmal alle Fehlerfälle durch und stellt sicher, dass diese bei Euch nicht auftreten.
2) Zum Testen der Positionsrekonstruktion aus dem Tiefenbuffer: gebt euch erst die Tiefe aus (ggf.) Linarisiert. Zum Testen könnt ihr dann die rekonstruierte Position ausgeben und mit dem Positionsbuffer (aus dem GBuffer-Pass) vergleichen. (Achtung: Spaces --> World-World / View-View)
3) Die Schattenmap der Punktlichtquellen kann getestet werden, in dem diese als Skymap gerendert wird.

Weitere (freiwillige) Funktionalitäten:

1) Es kann auch SSAO benutzt werden.