Aufgabe 4: Screen Space Ambient Occlusion

(Neue) Techniken:

Ambient Occlusion & SSAO, Depth Reconstruction, ViewSpace

Beschreibung:

In dieser Aufgabe soll die Technik Screen Space Ambient Occlusion (SSAO) implementiert werden. Dabei sollt Ihr mit Hilfe des aus Ausgabe 3 bekannten G-Buffers die bereits gerenderten Szene so verarbeiten, dass eine Textur entsteht, die eine Approximation der allgemeinen Sichtbarkeit eines Punktes speichert. Diese Werte sollen anschließend beim erneuten Rendern der Szene dazu verwendet werden, den ambienten Anteil der Belechtung abzuschwächen. Dadurch wird die optische Qualität deutlich erhöht.

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

Kernanforderung:

- Im Fragment Shader des SSAO Rendering Passes wird ein Tangent Space für den korrespondierenden Oberflächenpunkt x berechnet.
- Durch zufällige Vektoren v im Tangential-Koordinatensystem werden Sample-Punkte s = x + v bestimmt, die auf den entsprechenden Buffer projiziert werden.
- Die Tiefe der Sample-Punkte s wird mit dem Tiefen-Wert an der projizierten Position verglichen und für jeden Sample-Punkt s, der durch den GBuffer Tiefenwert verdeckt ist, wird der Ambient Occlusion-Wert verändert.
- Der berechnete Ambient Occlusion-Wert wird genutzt, um die ambiente Beleuchtung abzuschwächen.

Weitere Funktionalitätsanforderungen:

- **Shaderprogramm für SSAO**
  Für die SSAO existiert ein Shaderprogramm, mit dem ein Full Screen Quad auf den Framebuffer für die SSAO gerendert wird. Ausschließlich mit diesem Shaderprogramm werden die Informationen des Pre-Pass ausgelesen und verarbeitet.
- **Tangent Space berechnen**
  Das Shaderprogramm für die SSAO erzeugt pro Pixel einen Tangent Space auf Grundlage der Oberflächennormale. Achtung: Da der Pre-Pass die Normale im Weltkoordinatensystem speichert, befinden sich auch die Tangenten und Bitangenten im Weltkoordinatensystem. Die Normale wird an der entsprechenden Position aus den Texturen des Pre-Pass ausgelesen. Die Tangente und Bitangente bilden zusammen mit der Normale ein orthogonales Koordinatensystem. Die Orientierung der Tangente und Bitangente in der Oberfläche ist beliebig und muss nicht, wie in Aufgabe 2, nach den Texturkoordinaten ausgerichtet sein. Diese werden vom Pre-Pass ohnehin nicht bereitgestellt. In keinem Fall wird einer der Vektoren zum Nullvektor.
- **Position des Samples**
  Die Position des Samples im entsprechenden Koordinatensystem wird berechnet. Dazu wird die Position der Oberfläche aus GBuffer des Pre-Pass ermittelt und um einen pseudo-zufälligen Vektor in der Hemisphäre des Oberflächenpunktes verschoben. Für die Erzeugung des Vektors werden die Funktionen hammersley und sampleHemisphereCos genutzt. Diese erzeugen normierte Vektoren in der Hemisphäre um die Z-Achse (Tangen-Space), deren Verteilung dem $\cos(\theta)$ entspricht. Mit dem Tangent Space werden die erzeugten Vektoren in das entsprechende Koordinantensystem transformiert.
- **Abstand der Samples**
  Der Abstand der Samples vom Ursprung (dem Oberflächenpunkt) ist pseudo-zufällig und wird mit einer Potenzfunktion so angepasst, dass nahe dem Ursprung mehr Samples erzeugt werden. Alle Samples werden um einen kleinen Wert entlang der Normale verschoben. So wird verhindert, dass Samples, die in einem 90°-Winkel zur Normale erzeugt werden, durch Gleitkommaungenauigkeiten als "verdeckt" gewertet werden. Somit findet auf einer ebenen Fläche keine Verdeckung statt. Der Radius der Halbkugel soll per Tastatur einstellbar sein, sollten hierbei Artefakte bei Randwerten entstehen ist dies für diese Übung zu vernachlässigen.
- **Projektion und Verdeckungstest**
  Die Position jedes Samples wird projiziert und der Tiefenwert mit dem Tiefenwertes an der korrespondierenden Position verglichen. Wenn der Tiefenwert hinter dem des GBuffer Samples liegt, wird das Sample als "verdeckt" gewertet. Sonst als "sichtbar". Der SSAO-Wert wird entsprechend angepasst.
- **Verwerfen von Samples mit zu großem Abstand**
  Samples, die einen zu großen Abstand zu der korrespondierenden Oberfläche haben, werden nicht mit in die Berechnung einbezogen. Hierzu ist ein Schwellwert definiert, der im Verhältnis zur Größe der Szene sinnvoll ist. Nutzt hierfür die Funktion smoothstep, damit keine harten Kanten entstehen.
- **Blur der SSAO Textur/des SSAO Buffers**
  Mittels Tastendruck kann ein Blurfilter auf die SSAO Textur angewendet werden. Dadurch kann die Anzahl der Samples im SSAO-Pass verringert werden.
- **Anwenden auf Beleuchtung**
  Die Ambient Occlusion wird beim erneuten Zeichnen der Szene ausschließlich mit dem ambienten Anteil der Beleuchtung multipliziert. Mit der Tastatur kann dies ein- und ausgeschaltet werden.

- **Depth Buffer**
  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):

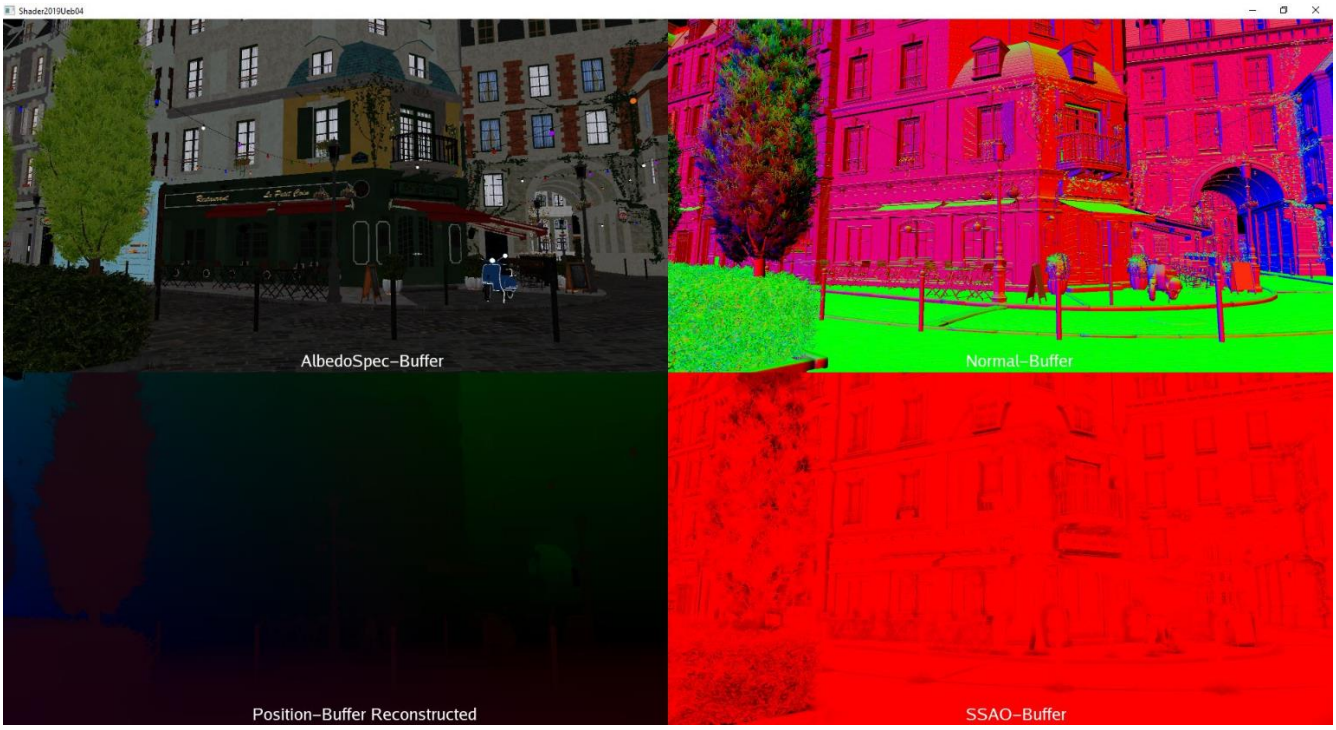
Taste	Funktion
Esc	Beenden des Programms
F1	Toggle Fullscreen
F2 [Optional]	Toggle Tangetspace
F3 [Optional]	Toggle Wireframe
F10	Reload Shader
1	Toggle Normalmapping
2	Toggle Parallaxmapping
3	Toggle Tessellation
4	Toggle SSAO
H/h	Toggle Hilfe
F/f	Goggle FPS
G/g	Toggle Grid/Szene
D/d	Toggle Directional 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
J/j	AO Radius erhöhen
K/k	AO Radius verringern
B/b	Toggle SSAO Blur

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
SSAOShader *(Vertex & Fragment)*	Berechnung der Screen Space Ambienten Verdeckung
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 SSAO Abschwächung)*
PointLightShader *(Vertex & Fragment) (Aufg 3)*	Point Light-Beleuchtung *(Mit SSAO Abschwächung)*
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) lightVisShader *(Vertex & Fragment)* (Aufg 3) *[optional]*	Kann Text auf dem Display ausgeben Zeichnet die Punkt-Lichtquelle als Primitive Geometrie (Cube oder Sphere)

* Hier findet das Normalmapping und Parallaxmapping statt.



Tipps:

1) Schaut Euch die Speicherung der Normalen im Pre-Pass an, um sie im Shaderprogramm für die SSAO korrekt zu dekodieren.
2) Wählt für das Berechnen der Tangenten und Bitangenten einen zusätzlichen Vektor zur Normale und berechnet ein Kreuzprodukt. Achtet darauf, dass dieses Kreuzprodukt immer einen orthogonalen Vektor zurückliefert und behandelt den Fall, indem dies nicht geschieht. Lasst Euch nicht von Aufgabe 2 beeinflussen. Obwohl es sich um einen Tangent Space handelt, ist die Berechnung deutlich einfacher.
3) Geht nach Bearbeitung der Aufgabe noch einmal alle Fehlerfälle durch und stellt sicher, dass diese bei Euch nicht auftreten.
4) Die Funktion sampleHemisphereCos erzeugt Einheitsvektoren, die Ihr anschließend zufällig skalieren sollt. Es kann durchaus Sinn machen, Samples auch in einem Abstand > 1 zu positionieren.
5) Achtet bei Vektor- & Matrix-Multiplikationen auf die Koordinatensysteme in denen sich die Vektoren befinden.
6) 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)

Weitere (freiwillige) Funktionalitäten:

1) Da hier Verdeckung behandelt wird, können wir auch einen Schritt weiter gehen und Schatten implementieren. Diese könnt Ihr z.B. mit Shadow-Maps erstellen. Informationen dazu findet ihr im Zusatzmaterial.