Prof. G. Zachmann
P. Meißenhelter
R. Fischer
A. Mühlenbrock

Universität Bremen
Fachbereich 3
Arbeitsgruppe CGVR
12. Januar 2022

Wintersemester 2021/22

Übungen zu Computergraphik - Blatt 10

Abgabe am 24.01.2022, 23:59 Uhr

In diesem Aufgabenblatt beschäftigen wir uns mit der Beleuchtung von 3D-Modellen, die im Wesentlichen in Shadern implementiert wird. Laden Sie sich dazu das *DeferredLightingFramework* herunter, welches auf der CGVR-Seite zum Download bereitsteht.

Aufgabe 1 (Blinn-Phong Modell, *3 + 1 Punkte*)

a) Implementieren Sie das Blinn-Phong Beleuchtungsmodell an markierter Stelle in der Shader-Datei `forward.frag`. Die entgültige Farbe eines Fragments soll sich über die folgende Formel ergeben:

$$I_{out} = k_a \odot I_{amb} + \sum_{j=1}^{n} k_\mathrm{d} \odot I_{\mathrm{diff}_j}(\mathbf{n}\cdot\mathbf{l_j}) + k_\mathrm{s} \odot I_{\mathrm{spec}_j}(\mathbf{h_j}\cdot\mathbf{n})^p$$

Für die Material-Komponenten k_a, k_d und k_s kann die vorgegebene Variable `kD` verwendet werden. Für p kann die uniform-Variable `shininess` verwendet werden. Bei I_amb handelt es sich um eine globale Konstante, die die Farbe einer ambienten Lichtquelle beschreibt (siehe gleichnamige Variable im Shader). Mit I_{diff_j} und I_{spec_j} sind die Farben der jeweiligen Lichtquelle gemeint, die durch die Attribute `diffuse` und `specular` der `Light`-Struct gegeben sind, welche zudem die Position der Lichtquelle aus Sicht der Kamera enthält. $\mathbf{l_j}$ beschreibt die Richtung der Lichtquelle ausgehend vom Fragment und mit $\mathbf{h_j}$ ist der "half-vector" gemeint. $\mathbf{n}$ ist die Oberflächennormale (siehe `normal` im Shader). Zur Berechnung des "half-vector" ist die vorhandene in-Variable `position` von Bedeutung - sie gibt die Position des Fragments aus Sicht der Kamera an. Stellen Sie bei Ihrer Implementierung sicher, dass die entsprechenden Vektoren normiert sind und das Fragment nur beleuchtet wird, wenn der Winkel zum Licht nicht größer als 90 Grad ist.

b) Erweitern Sie Ihren Code aus Ausgabenteil (a), sodass das Licht in Abhängigkeit von der Distanz d abgeschwächt wird. Die hier zu implementierende benutzerdefinierte Abschwächungsfunktion ist wie folgt definiert:

$$i(d) = \begin{cases} (1 - \frac{d}{d_\mathrm{max}})^2 & d < d_\mathrm{max} \\ 0 & \text{sonst} \end{cases}$$

Bei der Distanz d soll es sich um die Distanz zwischen Lichtquelle und Fragment handeln. Für d_max soll das `range`-Attribut der Light-Struct des jeweiligen Lichtes benutzt werden.

Aufgabe 2 (Deferred-Shading, *1 + 1 + 1 + 3 Punkte*)

In dieser Aufgabe beschäftigen wir uns mit dem Deferred-Shading

a) Beschreiben Sie ein Szenario, in welchem sich das herkömliche sofortige Shading besser eignen dürfte und ein weiteres Szenario, wo sich das Deferred-Shading besser eignen dürfte, jeweils mit einer kurzen Begründung.

b) Sehen Sie sich im Quellcode der Datei `glwidget.cpp` den Abschnitt A (Z.412 – Z.431) und den Abschnitt B (Z.436 – Z.472) an. Können Sie lediglich am Quellcode der entsprechenden Zeilen erkennen, welcher Abschnitt Render-Pass 1 und welcher Abschnitt Render-Pass 2 ist? Geben Sie Ihre Einschätzung inklusive einer kurzen Begründung an, die berücksichtigt, was in den jeweiligen Zeilen gemacht wird.

c) Im ersten Pass rendern wir nicht direkt auf den Viewport, sondern auf mehreren 2D-Texturen, die genauso groß sind wie der Viewport der Applikation. Die Texturen werden bereits im Programm angelegt und beim Skalieren des Fensters mitskaliert. Ihre Aufgabe ist es, das Herausschreiben der zur Beleuchtung benötigten Attribute *Color*, *Normal* und *Position* im Fragment-Shader `deferred_pass1.frag` zu implementieren, die folglich jeweils in eine Textur gespeichert werden.

d) Im zweiten Pass wird lediglich ein großes Quad über den gesamten Viewport gerendert, um für jeden Pixel des Viewports den Fragment-Shader genau einmal zu aktivieren. Sehen Sie sich den Fragment-Shader `deferredpass2.frag` an. Implementieren Sie hier ebenfalls das Blinn-Phong-Modell aus Aufgabe 1 inklusive Abschwächung (1b). Nutzen Sie die Attribute aus den Texturen.

Tipp: Die vorliegende Implementierung in `deferred_pass2.frag` ließt bereits die Color-Textur aus und zeigt diese auf dem Viewport an. Dies ist auch der Grund, warum Sie nach Implementierung der Aufgabe 2c die Color-Textur in der Anwendung angezeigt bekommen haben. Bevor Sie mit der Implementierung des Blinn-Phong Modells starten, können sie austesten, ob auch die Normal-Textur und die WorldPosition-Textur sinnvolle Werte enthalten - sehen sie sich die in den Vorlesungsfolien zu erwarteten Farbdarstellungen an.

Hnweis: Die Texturen, auf denen im ersten Pass gerendert wird, werden im Kontext des Deferred-Shading oft auch als G-Buffer bezeichnet. Grundsätzlich lassen sich 2D-Texturen nicht nur zur Darstellung von unbeweglichen 2D-Grafiken nutzen, sondern eben auch als allgemeine 2D-Buffer (≈ 2D-Arrays), die in jedem Frame änderbar sind und in ihren Pixeln z.B. int-, float-, vec2-, vec3- oder vec4-Werte speichern.