HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

Übungen zur Vorlesung Computergrafik 1
- Wiederholung -

Soommersemester 2018

Prof. Dr. Ingo Ginkel

Aufgabenblatt 6

13.07.2018

Aufgabe 13: Material, Farbe und Licht

Hinweis: Aufgabe gekürzt

Implementieren Sie eine Klasse `CgAppearance` um Material-Eigenschaften und andere Attribute für Objekte im Szenegraphen zu verwalten.

a) Implementieren Sie eine Klasse `CgAppearance`, die die Farb- und Material-Eigenschaften eines Objektes verwaltet. Als Material-Parameter sollen alle Parameter verwaltet werden, die beim Phong-Beleuchtungsmodell benötigt werden. Diese sollen beim Rendering jeweils ausgelesen und für das aktuell zu zeichnende Objekt verwendet werden. Übertragen Sie diese beim Zeichnen mit den Funktionen `setUniformValue(...)`, so dass diese im Shader-Programm genutzt werden können (vgl. nächste Aufgabe).

b) Implementieren Sie zusätzlich eine RGB-Farbe pro Objekt, die dann verwendet wird, wenn die Beleuchtung ausgeschaltet ist (ggf. eigenen Event dafür implementieren).

c) Implementieren Sie entsprechende GUI-Elemente und Events, so dass Material aus einer vordefinierten Liste (ca 5 Materialien aus den VL-Folien) gewählt werden kann. Die Material-Events sollen von Ihrem Szenegraphen verarbeitet werden und auf das jeweils gerade selektierte Objekt angewendet werden (analog für die Farbe ohne Beleuchtung).

d) Implementieren Sie eine Klasse zur Repräsentation einer Punktlichtquelle (Instanz z.B. als Teil des Szenegraphen in Welt-Koordinaten) und übertragen Sie Position und Eigenschaften (alle die für Phong-Beleuchtung benötigt werden) wieder mit `setUniformValue(...)` an die Shader-Programme. Die Position und Lichteigenschaften können initial gewählt werden und müssen nicht veränderbar sein.

Aufgabe 14: Shader Programm zur Beleuchtungsberechnung

Hinweis: Aufgabe unverändert

Implementieren Sie eine Funktionalität um Objekte zu beleuchten. Schreiben Sie dazu zwei eigene Shader-Programme in GLSL (Version 130, deswegen die ältere Version weil das in Qt und auch in VMs noch funktioniert).
Sie können entweder die Dateien `simple.vert` und `simple.frag` verändern oder eigene Text-Dateien anlegen und diese in `CgQtGlRenderWidget.cpp` Zeile 238 einlesen lassen. Wahlweise können Sie auch eigene Events implementieren, die die Shader-Source-Files umschalten können (mit `void CgBaseRenderer::setShaderSourceFiles(std::string filename_vert, std::string filename_frag)`).

a) Machen Sie sich mit der Shader-Sprache GLSL vertraut. Viele Dinge sind analog zur glm-Library gestaltet (etwa Vektoren, Matrizen und mathematische Operationen). Informationen finden Sie z.B unter

`https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.1.30.pdf`.

b) Schreiben Sie einen Vertex-Shader zur Berechnung von Gouraud-Shading. Berechnen Sie also die Beleuchtung pro Vertex im Vertex-Shader (in Kamera-Koordinaten) und geben Sie die berechnete Intensität als out-Variable an den Fragment-Shader weiter.

c) Schreiben Sie einen Fragment-Shader zur Berechnung von Phong-Shading. Geben Sie im Vertex-Shader die benötigten Werte als out-Variable an den Fragment-Shader weiter und berechnen Sie die Beleuchtung dort pro Pixel in NDCs.

d) Implementieren Sie eine Funktionalität bei der sich per GUI pro Objekt die verschiedenen Shading-Arten (Gouraud/Phong) umschalten lassen und analog dazu die Art der Interpolation für die Normalen (flat/smooth).

Sie können diese Einstellungen bei den GUI-Elementen für Farbe und Material aus der vorherigen Aufgabe ergänzen.

Hinweis 1: Als Standard ist smooth (d.h. Interpolation für jeden Pixel) in der Systematik der out-Variablen beim Vertex Shader vorgesehen. Für flat-Shading wird hier hinter `out` noch `flat` geschrieben (und zusätzlich auch analog bei der entsprechenden Fragment-Shader `in`-Variable) um zu verhindern, dass die Normalen interpoliert werden.

Hinweis 2: Mit dieser Funktionalität kann sowohl Flat-Shading als auch Flat-Phong-Shading realisiert werden.

Aufgabe 15: Beleuchtungsberechnung mit Schatten

Hinweis: Aufgabe neu

Implementieren Sie eine Funktionalität, die ein gerendertes Bild mit Phong Beleuchtungsmodell und Schatten in eine Datei abspeichert.

a) Implementieren Sie eine Hilfs-Klasse zur Repräsentation eines Bildes. Leiten Sie diese von `CgBaseImage` ab. Sie können den Inhalt ihres Bildes dann mit
`CgBaseRenderer::writeImageToFile(CgBaseImage* image, std::string filename)`
in eine Datei abspeichern (z.B. mit der Endung .jpg wird automatisch eine jpg-Datei erstellt).

b) Führen Sie auf Knopfdruck analog zum Picking einen Schnittpunkt-Test mit je einem Picking-Strahl durch jeden einzelnen Pixel des Fensters durch und berechnen Sie jeweils den nächstgelegenen Schnittpunkt mit den Objekten der Szene. Berechnen Sie an diesem Punkt das Phong Beleuchtungsmodell (sie müssen sich eine smooth-Normale selbst berechnen) und schreiben sie die berechnete Farbe in eine Instanz ihrer Bild-Klasse. Speichern Sie das Ergebnis als Datei ab.

c) Erweitern Sie die Funktionalität aus b) so, dass auch Schatten erzeugt werden. Schicken Sie dazu vor der Auswertung des Beleuchtungsmodells an einem getroffenen Punkt einen

Shadow-Ray zur Lichtquelle und prüfen Sie ob der Punkt im Schatten liegt. Falls ja berechnen Sie nur den ambienten Teil des Beleuchtungsmodells um den Schatten zu erzeugen.

Hinweis: Das Verfahren hier heißt Ray-Casting. Es ist quasi der erste Schritt eines Raytracers (schieße einen Strahl durch jeden Pixel) ohne die rekursive Weiterverfolgung der Strahlen.