TH KÖLN
Prof. Dr. H. Stenzel
Advanced Media Institute

Technology
Arts Sciences
TH Köln

23. März 2021
Abgabe: spätestens am 09.07.2021

Computergrafik und Animation SoSe2021
Blatt 4 - Lights

In dem letzten Aufgabenblatt dieses Semesters werden Sie sich mit der Beleuchtung einer Szene anhand des Phong-Modells beschäftigen. Dies wird in der Kategorie *Materials and Light* des THe-Learning-Systems thematisiert.
Testen Sie Ihre Implementierung nach jedem Schritt, die zusammengesetzten Teilergebnisse sollen letztlich ein Renderergebnis ähnlich der Abbildung 3 ergeben.

4.1 Phong-Modell

$$L_O = \underbrace{M_e}_{A} + \underbrace{M_d \cdot L_a}_{B} + \sum_{i=1}^{n} (\underbrace{M_d \cdot \max(\cos\alpha, 0)}_{C} + \underbrace{M_S \cdot \max(\cos\beta, 0)^k}_{D}) \cdot L_i$$

4.1.1 Phong-Modell Terme

Die obengenannte Gleichung stellt das erweiterte Phong-Modell dar. Bitte machen Sie sich mit der Theorie vertraut und spezifizieren Sie die Buchstaben A - D bzgl. ihres Aufgabenbereiches. In Abbildung 1 sind die verwendeten Winkel visualisiert.

A -

B -

C -

D -

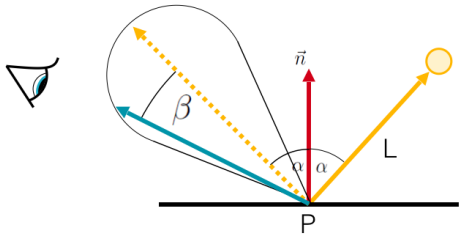

Abbildung 1: Schematische Darstellung des Phong-Modells

4.1.2 Cosinus-Terme

Überlegen Sie warum das $max(cos, 0)$ bei den Cosinus-Termen notwendig ist.

4.2 Integration einer Punktlichtquelle

Mit dem Aufgabenzettel haben Sie wieder zwei Interfaces erhalten, welche für die Beleuchtung genutzt werden sollen. Eines ist für die Implementierung einer Punktlichtquelle und eines für ein Spotlight.
Denken Sie daran, dass im Folgenden alle Berechnungen im Viewspace durchgeführt werden sollen. Daher müssen auch alle benötigten Vektoren in dieses Koordinatensystem transformiert werden.

4.2.1 `PointLight`

Erstellen Sie bitte die Klasse `cga.exercise.components.light.PointLight`, welche von der Klasse `Transformable` erbt und das Interface `cga.exercise.components.light.IPointLight` nutzt. Das `PointLight` hält als Eigenschaften die Position und die Lichtfarbe, mit der das Licht erstrahlen soll. Die übergebene Position muss das PointLight im Weltkoordinatensystem platzieren. Nutzen Sie hierfür die Vererbungshierarchie, um die globale Translation im Init-Block der Klasse durchzuführen und sie somit in die Modelmatrix des Objektes einfließen zu lassen.
Außerdem muss die Methode `bind(...)` implementiert werden, die das Interface definiert. In der Methode sollen nach gewohnter Art die Parameter der Lichtquelle (Farbe und Position) an die Shader übertragen werden.

4.2.2 `Scene`

Im `Init`-Block der `Scene` muss das PointLight angelegt und mit den entsprechenden Parametern versehen werden. Bedenken Sie, dass ein Vektor(0,0,0) für die Lichtfarbe eher unpassend ist. Zudem setzen Sie die Lichtquelle als Child des Motorrads, sodass es sich mit dem Motorrad mitbewegt. Die Lichtquelle befindet sich allerdings immer noch auf gleicher Höhe mit der Ebene. Heben Sie sie etwas an, damit die Lichtquelle den Boden anstrahlen kann.
Abschließend muss das Punktlicht in der `render()`-Methode gebunden werden. Machen Sie sich die Reihenfolge innerhalb der Methode bewusst. Nicht jede Reihenfolge ist sinnvoll.

4.2.3 Shader

Für die Implementierung des Phong-Beleuchtungsmodells müssen die Shader entsprechend angepasst werden. Dort findet die eigentliche Beleuchtungsberechnung statt. Eine gute Informationsquelle ist hierfür `https://learnopengl.com/Lighting/Basic-Lighting`.

VertexShader(VS)
Im VS nehmen Sie die Position der Punktlichtquelle entgegen. Berechnen Sie die Vektoren `toCamera` und den `toLight`-Vektor für die Punktlichtquelle im Viewspace und übergeben Sie diese an den Fragment-Shader.
Hinweis: Diese beiden Vektoren dürfen erst im Fragment Shader normalisiert werden. Warum?

FragmentShader(FS)
Führen Sie die Berechnung der Phong-Komponenten im FS durch. Hierzu verwenden Sie die Daten die vom Vertexshader übergeben werden sowie Material-Uniforms usw. Beachten Sie dabei, dass die Winkel nicht größer als 90° werden dürfen und alle Vektoren vor der Benutzung normalisiert werden müssen.

4.3 Scheinwerfer

Ein Motorrad besitzt einen Frontscheinwerfer, welcher in diesem Projekt mittels eines Spotlights simuliert werden soll.

4.3.1 `SpotLight`

Hierfür erstellen Sie eine neue Klasse `SpotLight`, welche von der Klasse `PointLight` erbt und das Interface `ISpotLight` implementiert.
Charakteristisch für ein Spotlight ist, dass es gerichtet ist und somit in der Ausbreitung begrenzt. Es entstehen zwei Kegel, welche durch die abgebildeten Winkel definiert werden. Der Winkel γ definiert einen inneren Kegel, in dem die Fragmente voll beleuchtet werden. Für Fragmente außerhalb des inneren, aber innerhalb des äußeren Kegels schwächt sich die Beleuchtung linear ab. Dieser äußere Kegel wird durch den Winkel ϕ spezifiziert. Der abgebildete Winkel θ definiert den Winkel zwischen der Lichtrichtung der Lichtquelle und des Vektors zwischen dem betrachteten Punkt und der Lichtquellenposition. Nähere Informationen dazu finden Sie unter `https://learnopengl.com/Lighting/Light-casters`.

Der Konstruktor des Spotlights beinhaltet neben den vom PointLight-Konstruktor benötigten Parametern Lichtfarbe und Position zusätzlich die Winkeldefinitionen zum inneren und äußeren Kegel. Diese Winkel sind private Eigenschaften des `SpotLights`. Das Interface definiert die `bind(...)`-Methode, welche von Ihnen zu implementieren ist. Die Klasse muss die Lichtfarbe, Position, die Kegeldefinition und den Richtungsvektor an die Shader übermitteln. Hierbei können Sie die Vererbung zu Hilfe nehmen. Die Winkel sollten bereits als Radiant-Werte vorliegen, um sich die Umrechnung im Fragment-Shader zu ersparen.

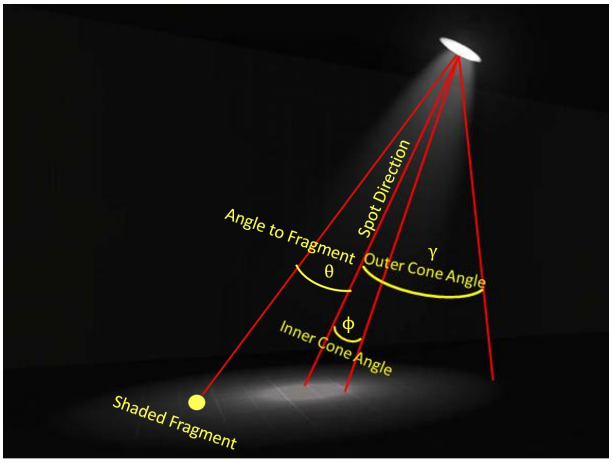

Abbildung 2: Aufbau eines Spotlights

4.3.2 `Scene`

Von der Instanzierung bis zum RenderCall gehen Sie wie in Aufgabe 4.2.2 vor. Das Spotlight sollte sich vorne am Motorrad befinden und leicht nach unten geneigt sein, damit der Boden durch den Scheinwerfer beleuchtet wird.

4.3.3 Shader

VertexShader(VS)
Im VS nehmen Sie die Position des SpotLights entgegen und berechnen einen weiteren `toLight`-Vektor für das SpotLight.

FragmentShader(FS)
Im Fragment-Shader muss die jeweilige Lichtintensität I berechnet werden, mit der die Lichtquelle auf das Fragment strahlt. Für Fragmente im inneren Kegel ist die Helligkeit noch maximal, die Intensität ist 1. Fragmente außerhalb des äußeren Kegels werden gar nicht angestrahlt, die Intensität ist 0.
Für die restlichen Fragmente lässt sich I nach folgender Formel berechnen:

$$I = \frac{cos(\theta) - cos(\gamma)}{cos(\phi) - cos(\gamma)}$$

Achten Sie darauf, dass I zwischen 0 und 1 liegen muss. Eine sinnvolle Methode in diesem Zusammenhang ist `clamp(...)`[1].
Die ermittelte Lichtintensität kann dann mit der Lichtfarbe der Lichtquelle multipliziert werden.
Methoden helfen im Übrigen dabei, den Code übersichtlich und strukturiert zu gestalten. Durch Auslagern der Bestandteile der Berechnung verbessert sich auch die Wiederverwendbarkeit.

4.3.4 Attenuation

Bei Punktlichtquellen gilt das *Inverse Square Law*, d.h. das Licht schwächt sich mit $\frac{1}{d^2}$ quadratisch ab. Um dieses Phänomen zu simulieren, bauen Sie in die Lichtberechnung eine Lichtabschwächung (engl. Light attenuation) ein. Um bei der Anpassung der Visualisierung etwas mehr Spiel zu haben[2], verwenden Sie hierfür bitte einen Attenuation Factor F_{att} bestehend aus einem konstanten (K_c), linearen (K_l) und quadratischen (K_q) Term:

$$F_{att} = \frac{1.0}{K_c + K_l * d + K_q * d^2}$$

Die Distanz d zu den Lichtquellen erhält man implizit durch die Länge der `toLight` Vektoren vom Vertexshader.

[1] `https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/clamp.xhtml`
[2] Ohne Gammakorrektur sind unsere Beleuchtungsberechnungen nichtlinear verzerrt, daher ist dieser "Hack" nötig. Weitere Informationen: `https://learnopengl.com/Advanced-Lighting/Gamma-Correction`

Die Light Attenuation ist unter `https://learnopengl.com/Lighting/Light-casters` genauer beschrieben. Bitte führen Sie diese Berechnung in dem FS durch und skalieren Sie die Lichtfarbe der Lichtquellen entsprechend.
Hierfür müssen allerdings die benannten Faktoren K_c, K_l und K_q im Vorfeld in der Klasse `PointLight` als Eigenschaft gesetzt, initialisiert und in `bind(...)` an den Shader übermittelt werden. Nutzen Sie hierfür sinnigerweise einen Vector3f. Als brauchbare Werte haben sich für die Lichtquellen folgende LightAttenuation-Parameter herausgestellt (Tabelle 1):

Lichtquelle	K_c	K_l	K_q
PointLight	1.0	0.5	0.1
SpotLight	0.5	0.05	0.01

Tabelle 1: Parameter zur Lichtabschwächung

4.4 Farb-Variationen

Um das Spiel ein wenig schöner und bunter zu gestalten, geben Sie beim RenderCall noch eine Farbinformation an den Shader, welche die Emission der Objekte beeinflussen soll. Für den Boden können Sie einen statischen Farbwert definieren und mit der emissiven Textur multiplizieren. Wählt man *grün (0, 1, 0)* als Farbe, färbt sich der Boden wie in Abbildung 3.
Anschließend soll das Motorrad abhängig von der Zeit (`t`) ebenfalls die Farbüberlagerung erhalten, dabei aber dieselbe uniform-Variable des Bodens nutzen. Dabei lassen sich verschiedene Sinus-Werte der Zeit als RGB-Komponenten verwenden. Die Lichtfarbe des Punktlichtes soll die gleiche Farbe erhalten.
Haben Sie alle Aufgaben korrekt implementiert, ergibt sich das folgende Bild.

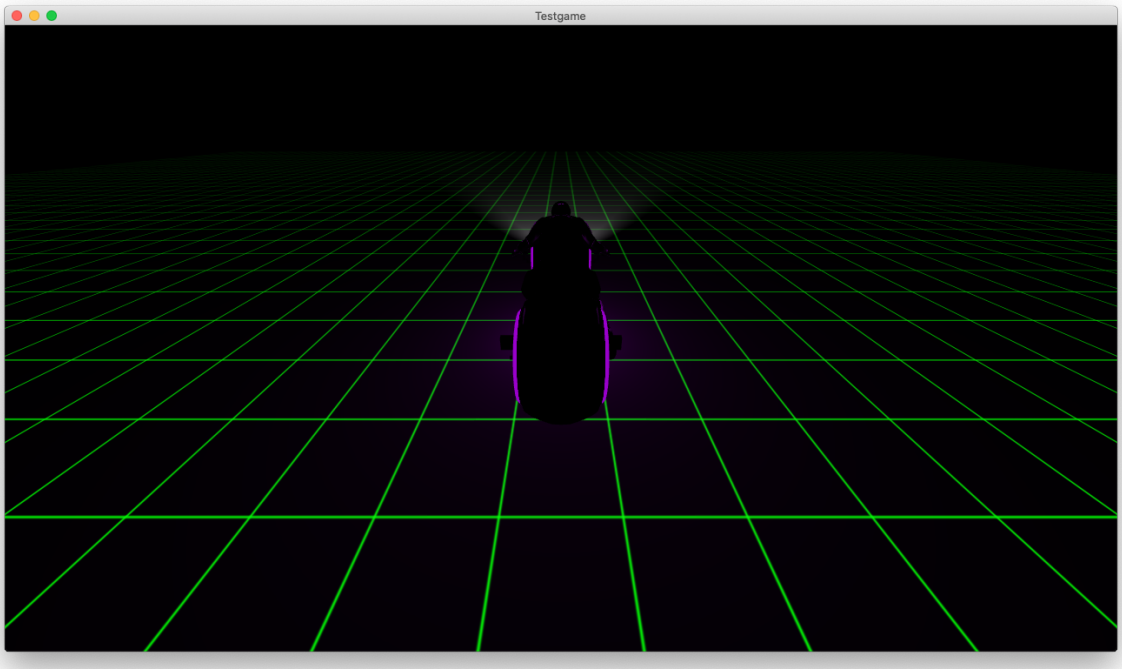

Abbildung 3: Finales Renderingergebnis

4.5 Kamera-Bewegung per Maus-Steuerung

Die Kamera soll nach dieser Aufgabe mit Hilfe der Maus um das Motorrad rotiert werden können, ohne es aus dem Fokus zu verlieren. Verwirklichen Sie diese Kamerabewegung in der Methode `Scene.onMouseMove(...)`.
Die Bewegung der Maus in x-Richtung (Differenz zwischen alter und neuer Position) stellt den Rotationswinkel um die y-Achse dar. Dieser sollte allerdings um den Faktor 0.002 skaliert werden, da die Cursor-Positionen im ScreenSpace gegeben sind.

4.6 Weitere Lichtquellen

Verteilen Sie einige weitere, verschiedenfarbige Punktlichtquellen in der Szene, beispielsweise in den Ecken der Ground Plane.

4.7 Erweiterung des Beleuchtungsmodells

Mit einer kleinen Änderung können wir den Realismus des einfachen Phong Beleuchtungsmodells deutlich verbessern. Implementieren Sie hierzu den Ansatz basierend auf den Half-Vectors (Lektion "Specular Reflection - Halfvector" 04:10). Dieses Modell wird auch "Blinn-Phong"-Modell genannt. Weitere Informationen kann man unter `https://learnopengl.com/Advanced-Lighting/Advanced-Lighting` finden.