Media Informatics /
Graphics & Geometric Computing

Prof. Dr. Marcel Campen
Steffen Hinderink

UNIVERSITÄT OSNABRÜCK

Computergrafik (SoSe 2023)

Blatt 4

fällig Sonntag **21. Mai**, 24:00

Verspätet eingereichte Abgaben können nicht berücksichtigt werden.

Aufgabe 1 (Theorie: Lighting & Shading)

(a) (5P) Gegeben sei eine Oberfläche im Raum. Der Punkt $(0,0,-32)^\mathsf{T}$ liege auf dieser Oberfläche. An diesem Punkt habe die Oberfläche die Normale $(0,0,1)^\mathsf{T}$. Werten Sie das Phong-Beleuchtungsmodell an diesem Punkt aus. Verwenden Sie dabei *nicht* die Blinn-Approximation. Die folgenden Parameter sind gegeben:

- Projektionszentrum an der Stelle $(0,0,0)^\mathsf{T}$
- Punktlichtquelle 1 an der Stelle $(0,8,-24)^\mathsf{T}$ mit Farbe $(0.5,0,0)^\mathsf{T}$
- Punktlichtquelle 2 an der Stelle $(-16,0,-16)^\mathsf{T}$ mit Farbe $(0,0.6,0)^\mathsf{T}$
- $C_a = \mathrm{diag}(0.2,0.2,0.2)$
- $C_d = \mathrm{diag}(0.8,0.8,0.8)$
- $C_s = \mathrm{diag}(0.4,0.4,0.4)$
- Shininess $s = 10$

(b) (2P) Werten Sie das Phong-Beleuchtungsmodell für die gleiche Situation, allerdings nun *mit* Blinn-Approximation und Shininess $s = 20$, aus.

(c) (3P) Gegeben ist das Dreieck mit den Vertices $(1,5,-7)^\mathsf{T}$, $(-4,4,-7)^\mathsf{T}$, $(0,0,-7)^\mathsf{T}$ und den Vertex-Normalen $(0,0,1)^\mathsf{T}$, $(-1,0,0)^\mathsf{T}$, $(0,-1,0)^\mathsf{T}$. Die Beleuchtung des Punktes $(0,3,-7)^\mathsf{T}$, den das Dreieck enthält, soll mittels Phong Shading berechnet werden. Für welchen Punkt und welche Normale oder welche Punkte und welche Normalen muss dazu das Phong-Beleuchtungsmodell ausgewertet werden?

Hinweis: Wundern Sie sich nicht, wenn sich bei diesen Aufgaben "unschöne" Werte ergeben; es reicht dann aus, einige Nachkommastellen anzugeben.

Media Informatics /
Graphics & Geometric Computing

Prof. Dr. Marcel Campen
Steffen Hinderink

UNIVERSITÄT OSNABRÜCK

Aufgabe 2 (Praxis: Phong Lighting + Flat Shading)

Die vorgegebene Methode `createTriangles()` erzeugt eine Menge von Dreiecken, die zusammen eine Kugel annähern. Jedes Dreieck wird dann in der Methode `update()` ModelView-transformiert, beleuchtet, projiziert, und rasterisiert (also die bedeckten Pixel bestimmt und eingefärbt).

(a) (10P) Vervollständigen Sie die Funktion `light`, so dass diese für ein gegebenes Dreieck (in Kamera-Koordinaten) das Phong-Beleuchtungsmodell mit Blinn-Approximation auswertet und die resultierenden Farbe (als RGB-Tripel) zurückgibt. Es soll Flat Shading mit Per-Triangle-Lighting verwendet werden; die Auswertung ist also am Mittelpunkt (Schwerpunkt) des Dreiecks vorzunehmen.

Es ist eine Punktlichtquelle vorgesehen. Die Position dieser Punktlichtquelle (`lightPos`) (in Kamera-Koordinaten) können Sie mit den Slidern variieren. Die Farbe des Lichts der Lichtquelle ist als `lightColor` gegeben. Das Projektionszentrum befindet sich im Ursprung. Die Normale n brauchen (und sollen) Sie hier nicht aus dem Dreieck selbst berechnen; verwenden Sie einfach die Definition $n(p) = (p + (0,0,2)^\mathsf{T})/\|p + (0,0,2)^\mathsf{T}\|$ um eine Normale für den Punkt p zu berechnen. Dies funktioniert hier, weil das Objekt eine um -2 in z-Richtung verschobene Einheitskugel ist.

Im Array `inputs` befinden sich alle vom Benutzer eingestellten Parameter, die Sie im Phong-Modell benötigen. Auf diese kann mit dem Namen als Index zugegriffen werden, z.B. ist der Wert des Shininess-Sliders `inputs["shininess"]`. Die Ambient- und Diffuse-Koeffizienten C_a und C_d erhalten Sie in dieser Aufgabe wie folgt: Die Reflektivität (umgangssprachlich "Farbe") jedes Dreiecks ist in dem Attribut `color` der Klasse `Triangle` ersichtlich (und steht in der Funktion `light` bereits als `triColor` zur Verfügung); diese müssen Sie noch mit den skalaren Werten der jeweiligen Slider multiplizieren. Den Specular-Koeffizienten C_s erhalten Sie analog, jedoch nehmen wir an, dass die Reflektivität für diesen Koeffizienten grundsätzlich $(1,1,1)^\mathsf{T}$ ist, so dass das auf spekulare Weise reflektierte Licht die Farbe der Lichtquelle beibehält.

Hinweis: Beachten Sie die Methoden `dot`, `cross`, `norm`, `normalize`, `add`, `sub` und `mul` von `Vector` bzw. `Point`; diese können hier nützlich sein.

Hinweis: Wir haben C_a, etc., als Matrix kennengelernt. Da alle Nichtdiagonaleinträge hier 0 sind, lässt es sich auch als Vektor darstellen. Für die Multiplikation mit einem anderen Vektor ist dann die komponentenweise Vektormultiplikation nötig; dazu steht die Funktion `mulVec` zur Verfügung.

Hinweis: Sie können die Größe des Canvas mit der Konstanten `size` anpassen, um ggf. das Rendering zu beschleunigen.

Aufgabe 3 (Bonus: Phong Shading)

Verwenden Sie Phong Shading für das Rendern der Dreiecke, wenn die dazugehörige Option ausgewählt ist (Boolescher Wert in `inputs["phongShading"]`). Dafür muss in der Funktion `draw` das Beleuchtungsmodell für jeden Pixel ausgewertet werden. Beachten Sie allerdings, dass die Beleuchtung im Kamerakoordinatensystem erfolgen muss, und nicht etwa nach der (nicht winkelerhaltenden) Projektion. Deshalb muss zusätzlich zum projizierten Dreieck (`projected[i]`) das nur durch die ModelView-Matrix transformierte Dreieck (`transformed[indices[i]]`) übergeben werden.