UNIVERSITÄT BASEL

Prof. Dr. Thomas Vetter
Departement Mathematik und Informatik

Spiegelgasse 1
CH – 4051 Basel

Patrick Kahr (patrick.kahr@unibas.ch)
Clemens Büchner (clemens.büchner@unibas.ch)

Computer Grafik 2019 - Übungsblatt 5

Ausgabe in Woche **11** (**2**.**5**.2019).

Vorführung der laufenden Programme im Tutorium Woche **12** (Abgabe **9**.**5**.2019).

Maximal zu erreichende Punktzahl: 6

Wir sind nun in der Lage Objekte auf die Bildebene zu projizieren und diese mit verschiedenen Beleuchtungsmodellen zu rendern.

In diesem Übungsblatt implementieren Sie ein Verfahren um Schatten zu berechnen und Sie werden mit Texturen arbeiten.

Aufgabe 1 - Schatten (3 Punkte)

Es gibt verschiedene Algorithmen zur Berechnung von Schatten. Wir schlagen vor, dass Sie Shadow Maps verwenden.

Die Grundüberlegung der Shadow Maps: Rendert man die Szene von der Lichtquelle aus, sind Flächen welche aus der Perspektive der Lichtquelle nicht sichtbar sind im Schatten.

Um die Viewmatrix aus der Sicht der Lichtquelle zu erzeugen, gehen Sie wie folgt vor.

$$M = \left(\begin{array}{c|c} \mathbf{Q} & \mathbf{d} \\ \hline \mathbf{0}^T & 1 \end{array}\right)$$

Wobei **Q** eine 3x3 Matrix ist, welche das Koordinatensystem aus Richtung der Lichtquelle aufspannt:

$$Q = \begin{pmatrix} \vec{x} \\ \vec{y} \\ \vec{z} \end{pmatrix}$$

Wenn die Position der Lichtquelle durch L gegeben ist, können wir das Koordinatensystem aus der Richtung der Lichtquelle wie folgt generieren:

$$\vec{z} = \frac{L}{|L|}$$
$$\vec{y} = \frac{\vec{z} \times \vec{x_0}}{|\vec{z} \times \vec{x_0}|} \mid \vec{x_0} = (1, 0, 0)^T$$
$$\vec{x} = \frac{\vec{y} \times \vec{z}}{|\vec{y} \times \vec{z}|}$$

a) View Matrix (1 Punkt)

Implementieren Sie die Methode `getViewMatrixOfLightSource` der Klasse `PinholeProjection`, welche die eben beschriebene ViewMatrix erzeugt.

Benötigte Dateien: projection.PinholeProjection.java
Mögliche Tests: Phong: Direction of LightSource, Reflectance: Direction of LightSource

(b) Shadow Map (1 Punkt)

Implementieren Sie die Methode `generateShadowMap` der Klasse `Occlusion`, die Viewmatrix aus der Richtung der Lichtquelle in `shadowProjection` und das damit erzeugte Korrespondenzfeld ind `shadowMap` speichert.

Benötigte Dateien: occlusion.Occlusion.java
Mögliche Tests: Phong: ShadowMap, Reflectance: ShadowMap

(c) Shadows (1 Punkt)

Ergänzen sie die Methode `inShadow` der Klasse `Occlusion`, welche einen double zwischen 0 und 1 zurückgibt. Der Rückgabewert soll 0 sein, falls der zu prüfende Punkt von der Lichtquelle nicht beleuchtet wird, also im Schatten liegt. Es soll 1 zurückgegeben werden, falls der Punkt von der Lichtquelle "gesehen" wird.

Ändern Sie ihren `PhongMeshRenderer`, so dass die Berechnung der Farbe berücksichtigt ob ein Schatten gezeichnet werden soll oder nicht. Falls Schatten gerendert werden sollen (`shadows` wahr ist), rufen Sie die Methode `inShadow` der Member-Variable `shadowSystem` auf und verwenden den Rückgabewert entsprechend. Verändern Sie nun ebenfalls die Klasse `ReflectanceMehsRenderer` um Schatten rendern zu können.

Hinweis 1: Wir blicken in die negative z-Richtung.
Hinweis 2: Beziehen sie die Member-Variable `shadows` im Renderer mit ein. Greifen Sie nur auf das shadowSystem zu, falls Schatten aktiviert sind und shadowSystem initialisiert ist.
Hinweis 3: Seien Sie vorsichtig, wo dass sie die ShadowMap generieren lassen. Für jede Lichtquelle wird bestenfalls nur einmal eine ShadowMap erzeugt.
Hinweis 4: Beziehen Sie im Code die Membervariable `shadowBias` mit ein. Diese kann durch den Graphischen Test verändert werden.

Benötigte Dateien: occlusion.Occlusion, renderer.PhongMeshRenderer.java, renderer.ReflectanceMeshRenderer
Mögliche Tests: Phong: Shadow Renderer, Reflectance: Shadow Renderer

Aufgabe 2 - Weiche Schatten (3 Punkte)

Betrachten Sie die von Ihnen gerechneten Schatten aus der vorigen Aufgabe. Dabei wird Ihnen auffallen, dass der Übergang zwischen Schatten und beleuchteten Bereichen unnatürlich hart ist. Überlegen Sie sich, weshalb dies in der Natur nicht geschieht. (1 Punkt)

Ziel dieser Aufgabe ist es weiche Schatten zu berechnen.

Eine Methode dafür ist Percentage Closer Filtering (PCF, siehe Vorlesung): Nachdem man die Objektkoordinaten auf die Shadow Map projiziert hat, testet man den z-Buffer auch für die umgebenden Pixel (z.B. 3 x 3). Diese Bool Werte können dann miteinander verrechnet werden. Wenn Sie wollen, können Sie auch eine andere Methode implementieren, um Soft Shadows zu rendern (z.B. Light Jittering).

Benötigte Dateien: occlusion.Occlusion.java
Mögliche Tests:Soft Shadow: Phong Renderer, Soft Shadow: Reflectance Renderer