Bericht Aufgabe 3

Nathalie Junker, Jannik Portz, Dennis Ritter

8. Dezember 2015

Inhaltsverzeichnis

1 Aufgabenstellung

Der Raytracer soll um verschiedene Beleuchtungsmöglichkeiten und Materialien erweitert werden.

1.1 Licht

Die abstrakte Basisklasse *Light* soll die Farbe des Lichts speichern und die beiden Methoden *illuminates* und *directionFrom* deklarieren. Die Methode *illuminates* ermittelt ob der übergebene Punkt von diesem Licht angestrahlt wird. Was in dieser Aufgabe ausschließlich für die Winkelbegrenzung des *Spotlights* relevant ist. Die Methode *directionFrom* gibt für den übergebenen Punkt den Vektor l zurück, der zur Lichtquelle zeigt. Die Klasse *PointLight* repräsentiert eine Punktlichtquelle, die gleichmäßig in alle Richtungen strahlt. Die Klasse *SpotLight* ist eine Lichtquelle die von einem bestimmten Punkt aus in eine gegebene Richtung innerhalb eines festgelegten Winkels abstrahlt. Die Klasse *DirectionalLight* repräsentiert das Sonnenlicht, welches überall in der Szene aus der gleichen Richtung kommt. Es soll darüber hinaus Ihre Klasse World um ein Attribut vom Typ *Color*, welches das *ambiente Licht* repräsentiert. Darüber hinaus muss Ihre Klasse *World* eine Liste aller Lichtquellen beinhalten.

1.2 Material

Die abstrakte Basisklasse Material hat eine Methode *colorFor*, dessen Implementierungen die Farbe für ein *Hit-Objekt* zurückgibt. Die *World* wird für die Ermittlung der Lichter benötigt. Das *SingleColorMaterial* enspricht dem bisherigen Verhalten. Die bei der Konstruktion übergebene Farbe wird einfach von *colorFor* zurück gegeben, unabhängig von Lichtquellen oder der Normalen. Das *LambertMaterial* stellt das Material für einen perfekt diffus reflektierenden Körper dar. Es ist folgende Gleichung zur implementierung des *LambertMaterials* zu verwenden:

$$c = c_d * c_a + \sum_{i=1}^{i_{max}} [c_d * c_l * max(0, \vec{n} \cdot \vec{l})]$$

c_a repräsentiert hierbei die Farbe des ambienten Lichts und c_d die Farbe für die diffuse Reflektion.

Das *PhongMaterial* stellt das Material für einen perfekt diffus reflektierenden Körper mit Glanzpunkt dar. Zur Implementierung ist die folgende Gleichung zu verwenden:

$$c = c_d * c_a + \sum_{i=1}^{i_{max}} [c_d * c_l * max(0, \vec{n} \cdot \vec{l}) + c_s * c_l * max(0, \vec{e} \cdot \vec{r}_l)^p]$$

c_s ist hierbei dir Farbe für die spekulare Reflektion des Glanzpunkts.

1.3 Änderungen am Hit-Objekt

Das Hit-Objekt muss um die Normale des Schnittpunktes ergänzt werden.

1.4 Änderungen an der Geometry-Klasse und am Dreieck

Anstatt einer Farbe soll die *Geometry-Klasse* nun ein Material besitzen. Der Raytracer fragt also nicht mehr direkt die Farbe ab, sondern ruft bei der Materialklasse die Methode *colorFor* auf.
Das Dreieck muss um Normalen für jeden Eckpunkt erweitert werden. Die Normalen werden auf der Fläche interpoliert.

1.5 Änderungen an der World-Klasse

Die *World-Klasse* benötigt eine Liste aller Lichtquellen und die Farbe für das ambiente Licht.

2 Lösungsstrategie

Wie in der letzten Übung haben wir bereits das Grundgerüst der Aufgabe bereits während der Übung implementieren können. Dabei haben wir wieder eine grobe Einteilung unter den Teammitgliedern vorgenommen und uns bei der Reihenfolge größtenteils an die empfohlene Vorgehensweise gehalten. Danach konnte jedes Mitglied wieder alle Aufgabenpunkte ohne eine feste Einteilung bearbeiten.

3 Implementierung

3.1 Licht

Die abstrakte Klasse *Light* gibt vor, dass jedes Licht ein *Color* Attribut und die Methoden *Boolean illuminates* und *Vector3 directionFrom* implementieren muss. Beiden Methoden muss ein *Point3* Punkt übergeben werden Alle Lichter des Raytracers erben von *Light*.

Die Klasse *PointLight* besitzt ein Attribut position vom Typ *Point3* welches die Position der Lichtquelle angibt. die Methode *illuminates* gibt derzeit immer true zurück, das diese Methode erst bei der Berechnung von Schatten eine Rolle spielt. *directionFrom* berechnet $\vec{l}$ der zur Lichtquelle zeigt. Es wird die Differenz aus der Lichtposition und dem übergebenen Punkt ermittelt und anschließend normalisiert.

Die Klasse *DirectionalLight* besitzt ein Attribut direction vom Typ *Vector3* die Richtung angibt in die das Licht scheint. die Methode *illuminates* gibt derzeit immer true zurück, das diese Methode erst bei der Berechnung von Schatten eine Rolle spielt. *directionFrom* berechnet $\vec{l}$ der zur Lichtquelle zeigt. Hierfür wird der direction-Vector Attribut mit -1 multipliziert und anschließend normalisiert.

Die Klasse SpotLight besitzt die Attribute *position:Point3*, *direction:Vector3* und *halfAngle:double*, wobei *positon* für die Position der Lichtquelle, *direction* für die Richtung in die das Licht scheint und halfAngle für den halben Öffnungswinkel des Lichts stehen. Die Methode *illuminates* berechnet den Winkel zwischen dem umgedrehten Vector $\vec{l}$ und dem direction Attribut und vergleicht den Winkel mit dem angegebenen Öffnungswinkel *halfAngle*. Ist der Winkel größer als *halfAngle* gibt die Methode *false* zurück, andernfalls *true*.

3.2 Material

Alle Materialien erben von der abstrakten Klasse *Material*, welche die Methode colorFor für jede erbende klasse erzwingt.

Das *SingleColorMaterial* addiert die übergebene Farbe mit der Lichtfarbe, wenn das Objekt angestrahlt wird.
Das *LambertMaterial* und *PhoneMaterial* Berechnen die diffuse Reflektion des Lichtes mit den in der Aufgabenstellung mitgelieferten Formeln, wenn das Objekt angestrahlt wird. außerdem werden alle Farbkomponenten, sofern sie nach der Berechnung größer als 1.0 sein sollten wieder auf 1.0 reduziert.

4 Probleme / Schwierigkeiten bei der Bearbeitung

- Sowohl beim LambertMaterial als auch beim PhongMaterial entstand im Inneren des eigentlich hellen Bereiches eine schwarz-Färbung. Dieses Problem trat erst auf als ambientes Licht mit eingesetzt wurde und deshalb

nicht sofort erkannt. Die Suche nach der Ursache des Problems gestaltete sich schwieriger als zunächst gedacht und nahm viel Zeit in Anspruch. Letztendlich konnten wir herausfinden, dass das Problem immer auftritt wenn einzelne Farbkomponenten größer als 1.0 waren und haben darauf die limitColorComponentsTo1 Methode in der Klasse Material implementiert.

- Die Farbe der AxisAlignedBox hat sich nicht korrekt verhalten. Zum Beispiel wurden keine Schattierungen angezeigt. Es stellte sich heraus, dass die Schnittpunktberechnung der Box fehlerhaft war. Nach der Korrektur funktionierte alles wie erwartet.

- Aufgrund eines Klammerfehlers bei der Normalenberechnung des Dreiecks kam es zu eigenartien Schatten auf dem Dreieck.