CGI: BERICHT ZU AUFGABE 6, RAYTRACER

VON SEBASTIAN DASSÉ, MAX NOVICHKOV, SIMON LISCHKA

1. Aufgabenstellung

Implementierung der Texturierung.

1.1. Dazu sollte zunächst ein UML-Diagramm des Raytracers mit allen geplanten Änderungen angefertigt werden.

1.2. Dann sollte im ersten Schritt eine Klasse `TexCoord2` für die Texturkoordinaten und die Berechnung der Texturkoordinaten für die Geometrieklassen implementiert werden.

1.3. Nun sollten drei verschiedene Texturklassen implementiert werden, die alle auf dem zu definierenden Interface `Texture` basieren sollten: `SingleColorTexture` für einfarbige Oberflächen, `ImageTexture` für eine Textur auf der Grundlage eines Bildes und `InterpolatedImageTexture` für eine Textur mit einem bilinear interpolierten, im Original gering aufgelösten Bild.

1.4. Zusätzlich sollte ein Material implementiert werden, dass in Anhängigkeit von der Stärke der Beleuchtung eines Punktes eine von zwei alternativen Texturen anzeigt.

1.5. Nach all diesen Modifikationen waren vier vorgegebene Szenen zu rendern. Außerdem sollte unter Verwendung der meisten implementierten Techniken eine freie Szene erstellt werden.

2. Lösungsstrategien

2.1. Zunächst wurden alle rein „formalen“ Änderungen der Methodenheader usw. umgesetzt und die neuen Klassen nach bewährtem Schema als leere Vorlagen generiert.

2.2. Die aus der Vorlesung bekannten Formeln wurden nach Möglichkeit an passender Stelle direkt in Code umgesetzt und das Ergebnis an kleinen Testszenen ausprobiert.

2.3. Das UML-Klassendiagramm erwies sich als hilfreiches Werkzeug zur Planung und um den Überblick über die Zusammenhänge zwischen den Klassen im Raytracer zu behalten.

Date: 10.02.2014.

3. Besondere Probleme

3.1. Bei der `ImageTexture` dauerte das Rendering bei der ersten Implementierung aussergewöhnlich lange. Das ließ sich dadurch lösen, dass in der `getColor`-Methode die Breite und Höhe des Bildes nicht mehr über Methodenaufrufe an das `BufferedImage` ermittelt wurde sondern, sich diese Werte in Instanzvariablen gemerkt werden. Außerdem wurden alle Variablen `final` gesetzt.

3.2. Eine weitere kleine Schwierigkeit bei der `ImageTexture` war es, die Konvertierung von Farben vom `BufferedImage` in unser eigenes `Color`-Format korrekt umzusetzen.

4. Implementierung

4.1. Texturkoordinaten und deren Berechnung. In der Klasse `TexCoord2` werden im Konstruktor die als Parameter übergebenen Werte u und v in positive Werte umgewandelt und Modulo 1 gerechnet. Auf diese Weise werden alle Texturen „gekachelt“.
Die Berechnung der Texturkoordinaten in den einzelnen Geometrien wurden nach den Formeln, die in der Aufgabe vorgegeben wurden, implementiert.

4.2. Texturen. Für die beiden Bild-Texturen wurde eine abstrakte Superklasse `AbstractImageTexture` definiert, in der das eigentliche Laden des Bildes implementiert ist und Instanzvariablen für das Bild, dessen Raster und dessen Breite und Höhe existieren.
In der `ImageTexture` wurde die Umrechnung der Texturkoordinaten auf die Bildkoordinaten durch einfache Multiplikation mit der Breite und Umwandlung in ganzzahlige Werte gelöst.
In der `InterpolatedImageTexture` wurde die Koordinatenumrechnung nach den Formeln, die in der Aufgabe vorgegeben wurden, implementiert.

4.3. Material für Tag und Nacht. Das `DayAndNightMaterial` wurde so implementiert, dass im Konstruktor zwei Materialien übergeben werden. In der `colorFor`-Methode wird zum Testen der Beleuchtungsintensität für einen Punkt ein `LambertMaterial` mit der Farbe weiß geprüft und der entprechende Farbwert in einen Helligkeitswert umgerechnet. Je nachdem ob dieser nun einen festgelegten Schwellenwert über- oder unterschreitet wird das Material für den Tag oder für die Nacht angezeigt.

5. Zeitbedarf

5.1. Der Zeitbedarf wurde bei allen Teammitgliedern mit jeweils 15 Stunden angegeben.