Bericht Aufgabe 2

Nathalie Junker, Jannik Portz, Dennis Ritter

23. November 2015

Inhaltsverzeichnis

1 Aufgabenstellung

Es soll ein kompletter Raytracer geschrieben werden, der unter Angabe der richtigen Parameter verschiedene Objekte in eine Szene setzt und diese in ein Fenster rendert. Dazu werden verschiedene Klassen benötigt.

1.1 Strahl

Die erste Klasse soll einen Strahl repräsentieren, der als beschreibende Parameter einen Ortsvektor und einen Richtungsvektor enthält, damit ein Objekt in einer Szene überhaupt sichtbar gemacht werden kann.

1.2 Kamera

Die Kamera des Raytracers unterteilt sich in drei Klassen. Die allgemeine Camera-Klasse, die die Vektoren für die Position der Kamera, die Blickrichtung und den sogenannten Up-Vector enthalt, soll automatisch die Vektoren für die x, y und z-Achse der Kamera berechnen. Außerdem stellt sie eine Methode zur Verfügung, mit deren Hilfe für einen bestimmten Pixel der Strahl zurückgegeben werden kann. Die Subklassen OrthographicCamera und PerspectiveCamera bauen auf der allgemeinen Camera-Klasse auf und stellen jeweils die perspektivische und orthographische Kamera dar.

1.3 Farben

Mit Hilfe dieser Klasse soll eine Farbe im RGB-Farbraum repräsentiert werden. Zusätzlich ist es möglich, eine Farbe mit einer anderen zu addieren, zu subtrahieren oder zu multiplizieren.

1.4 Geometrie

Die Basisklasse Geometry stellt die Basisklasse für alle geometrischen Elemente in einer Szene dar. Sie beinhaltet als Attribut eine Farbe und die abstrakte Methode hit(), die in den jeweiligen Subklassen für die Schnittberechnung ausimplementiert werden soll. Als Subklassen sind zunächst Plane, AxisAlignedBox, Triangle und Sphere vorgesehen.

1.5 Welt

In der Klasse Welt können Objekte erzeugt werden, die später in eine Szene dargestellt werden sollen. Sie beinhaltet eine Hintergrundfarbe und ebenfalls eine hit()-Methode, die den übergebenen Strahl gegen alle Objekte, die sich in der Szene befinden, überprüft.

1.6 Raytracer

Mit Hilfe der Raytracer-Klasse soll über alle Pixel des zu erzeugenden Bildes itieriert und für jeden Pixel ein Strahl aufgestellt werden. Wird dabei ein Schnittpunkt mit einem Objekt gefunden, wird die Farbe des Objekts abgebildet. Wird kein Schnittpunkt gefunden, wird der Pixel in der Hintergrundfarbe der Welt dargestellt.

2 Lösungsstrategie

Während der Übung hatten wir bereits die Möglichkeit, einen Großteil der Klassen gemeinsam auszuimplementieren. Dabei haben wir zunächst die Klassen unter den Teammitgliedern aufgeteilt, sodass zwar jeder sein eigenes Aufgabengebiet hatte, man sich jedoch bei der gemeinsamen Arbeit am gleichen Ort unterstützen und beraten konnte. Das Fertigstellen der Aufgabe wurde danach weniger streng aufgeteilt, sodass jeder an allen Teilen der Aufgabe arbeiten konnte, wann immer es zeitlich möglich war.

3 Implementierung

3.1 Strahl

Der Konstruktor der Strahl-Klasse benötigt ein Objekt o vom Typ Point3 und ein Objekt d vom Typ Vector3, um ein Ray-Objekt erzeugen zu können. Die in der Klasse enthaltene at()-Methode berechnet einen Punkt, an dem sich der Strahl befindet, in dem der Vektor d mit einem Objekt t vom Typ double multipliziert wird. Die Methode tOf() berechnet einen double-Wert, mit dem der Richtungsvektor d multipliziert werden kann, damit der Strahl den an die Methode als Parameter übergebenen Punkt p erreicht werden kann. Darüber hinaus werden die equals()- und die hashCode()-Methode der Superklasse überschrieben.

3.2 Kamera

Zusatzlich zu den oben genannten Berechnungen für die x, y und z-Achse der Kamera implementiert die abstrakte Klasse die equals()- und die hashCode()-Methode. Die Methode rayFor() wird hier als abstrakt gekennzeichnet und in den Subklassen OrthographicCamera und PerspectiveCamera ausimplementiert. Zusatzlich zur Überschreibung der hashCode()-Methode der Superklasse wird die toString()-Methode überschrieben, sodass alle Punkte, Vektoren und Winkel als String ausgegeben werden können.

3.3 Farben

Ein Color-Objekt besteht aus 3 double-Werten, die jeweils die Gewichtung des roten, grünen und blauen Lichts angeben sollen. Ein Color-Objekt kann außerdem mit einem anderen Color-Objekt addiert und subtrahiert werden. Eine Multiplikation ist sowohl mit einem Color-Objekt als auch mit einer Gleitkommazahl möglich. asInt() ermöglicht es darüber hinaus, ein Color-Objekt als Ganzzahl darzustellen.

3.4 Geometrie

Die einzelnen Geometrie-Klassen konnten ohne Probleme ausimplementiert werden, da in der Vorlesung die notwendigen Formeln zur Berechnung der wichtigsten Eckpunkte jeder einzelnen Geometrie besprochen und erklärt wurde. Diese Formeln haben wir so übersetzt, dass jede der Geometrie-Klassen ein entsprechendes Objekt erzeugt. Zusätzlich wird bei jeder hit()-Methode nicht nur auf einen Schnittpunkt hin überprüft, sondern es wird auch getestet, ob der Strahl überhaupt im entsprechenden Bereich der Szene liegt. Darüber hinaus wurden jeweils equals(), hashCode() und toString() überschrieben.

3.5 Welt

Diese Klasse erzeugt ein World-Objekt, der als Parameter ein Array mit Objekten der Klasse Geometry sowie eine Backgroundcolor übergeben wird. Außerdem berechnet die hit()-Methode dieser Klasse das kleinstmögliche t für einen bestimmten Ray und jede der übergebenen Geometrien des Arrays.

3.6 Raytracer

Die Raytracer-Klasse haben wir in drei Klassen Raytracer, RaytracerPanel und RaytracerTest aufgespalten. Letztere enthält eine main-Methode und erzeugt die zu rendernden Objekte und eine Kamera, die dann an ein RaytracerPanel übergeben werden. Das Raytracer-Panel sorgt mit Hilfe der Überschriebenen paintComponent()-Methode dafür, dass über jeden Pixel iteriert und mit HIlfe der traceRay()-Methode eines Raytracer-Objektes auf einen Schnittpunkt hin geprüft und mit einer Farbe versehen wird.

4 Probleme / Schwierigkeiten bei der Bearbeitung

- Einarbeitung in LaTex
- Die in der letzten Übung problematische Arbeitsaufteilung und damit einhergehenden merge-Konflikten konnten wir diesmal durch paralleles Arbeiten an unterschiedlichen Klassen gut vermeiden.
- Bei den ersten Testläufen zur Darstellung der angeforderten Geometrien ist uns aufgefallen, dass die Geometrien, trotz mit den in den Anforderungskriterien übereinstimmenden Daten, im Spiegelbild gerendert werden. Es stellte sich heraus, dass die Berechnungen der u, v und w Achsen des orthomalen Koordinatensystems in der abstrakten Camera-Klasse der Grund dafür waren.
- Rauscheffekt durch Rundungsfehler in der Axis-Aligned-Box