Dritte Übung

Anton Krebs 786645
Johann Hofmann 785841
Gregor Rosenbaum 785815

November 26, 2012

1 Aufgabenstellung

In der dritten Übung soll nun der grundlegende Raytracer implementiert werden. Das Ergebnis sind erste Bilder von geometrischen Strukturen, die mittels Raytracing ermittelt werden. Um dies zu erreichen, werden die ersten beiden Übungen übernommen und einige neue Klassen hinzugefügt.

- Eine abstrakte Camera-Klasse mit den Subklassen Orthographic- und PerspectiveCamera.
- Eine Klasse Ray, der Strahl, der von der Kamera geworfen wird.
- Eine Color-Klasse zum Speichern und Verarbeiten von Farbinformationen
- Einige Klassen, die Objekte repräsentieren, die vom Raytracer "getroffen" werden können.
- Eine Hit-Klasse, die einen "Treffer" mit einem Objekt symbolisiert.
- Die World-Klasse, die alle Objekte enthält.

2 Lösung

2.1 Ansatz

Als Rahmen für den Raytracer benutzen wir unsere Klasse ImageSaver aus der ersten Übung. Damit lässt sich das erzeugte Bild praktischerweise auch gleich speichern. Statt ImageCanvas nennen wir die Canvas-Klasse nun RayTracer. Sie malt nun keinen schrägen Strich mehr, aber ihre Grundfunktion bleibt gleich. Sie erzeugt ein Bild und geht es Pixel für Pixel ab. An jedem der Pixel wird von der Kamera ein Strahl berechnet und auf Kollision mit allen Objekten, die in der World enthalten sind, geprüft. Die World-Klasse ruft ihrerseits die hit-Methode der Objekte auf und gibt den Strahl zur Kollisionsprüfung weiter.

Die Formeln zur Berechnung der Strahlen und Kollisionen wurden bereits größtenteils in der Vorlesung behandelt und sind dadurch gegeben. Dies verringert den Schwierigkeitsgrad zumindest auf mathematischer Ebene.

2.2 Durchführung

Auch der Raytracer erbt von Canvas und kann damit problemlos in den JFrame von ImageSaver eingebaut werden. Statt auf rot muss der Pixel nun auf einen variablen Farb-Wert gesetzt werden. Dieser muss vom Typ Integer sein und gleich groß sein wie die äquivalente Hexadezimalzahl. Dazu wird der Klasse Color noch die Methode toInt() hinzugefügt.

Des weiteren müssen wir dem Raytracer noch eine World und eine Kamera geben, damit das Raytracing stattfinden kann. Die World-Klasse hat ein Array, das Objekte mit der Superklasse Geometry enthält. Eine List wäre zum einfacheren hinzufügen von Objekten während der Laufzeit praktischer, aber wir haben uns aus Performancegründen dagegen entschieden.

Jede Kamera auch immer ein eigenes Koordinatensystem mit den Achsen u,v und w. Diese können also gleich in der Superklasse Camera berechnet werden. Die Berechnung des Rays mit der Methode rayFor gestaltet sich durch die bereits vorhandenen Formeln einfacher als gedacht. Probleme machen hier aber Unachtsamkeiten bei Klammersetzung und Float- bzw. Integerdivison sowie kleine Fehler in der Vektorenbibliothek, die lange und mühsam durch Debugging entdeckt werden müssen. Bei der perspektivischen Kamera muss zudem darauf geachtet werden, dass Winkel in Radian umgerechnet oder am besten nur in Radian eingetragen werden.

Die einzelnen Objekte gestalten sich teilweise als schwieriger, hier müssen teilweise zusätzliche Informationen recherchiert werden, um alle Bedingungen zum Abschluss der Methode hit() herauszufinden. Auch hier plagen oft Fehler in der Vektorenbibliothek aus Übung 2 oder kleine Unachtsamkeiten.

Die Herausforderung, Klassen ohne Möglichkeit zum direkten Feedback (der Raytracer kann natürlich erst Bilder produzieren, wenn alle Elemente fertig sind) zu schreiben haben wir durch die Hilfe von (sehr) kleinen UnitTests besser bewältigt. Mit der Hilfe dieser kleinen Testklassen konnten wir z.B. unsere Kameras noch vor dem ersten Bild auf mathematische Korrektheit überprüfen. Auch die Objekte wurden mit Hilfe einer Testklasse mathematisch bestätigt.

3 Aufwand

Diese Übung war mit Abstand die schwierigste und zeitaufwändigste. Mit dem Zeitaufwand der zweiten Übung wären wir dieser nicht annähernd gerecht geworden. Trotz des hohen Schwierigkeitsgrads war die Übung trotzdem nicht unfair oder überfordernd. Die Gesamtarbeitszeit einzuschätzen ist sehr schwierig. Sie ist, wie gesagt, sehr sehr hoch. Wir können von mindenstens 12 Stunden individueller Arbeit pro Person und etlichen weiteren Stunden Teamarbeit ausgehen.