Raytracing Aufgabe 02 - Schnittberechnung

Benjamin Schurian, Clemens Pade, Robert Dimitrov
Computergraphik I

WS2015/16

1 Einführung

In dieser Aufgabe sollen erste Versuche mit der Implementierung eines Raytracers gemacht werden, der gegebene Szenen und geometrische Objekte in ein Fenster rendert. Dafür sollten ein paar wichtige Klassen geschrieben werden, die auch für das spätere Ray-Tracing hilfreich sein werden. Die Klassen aus der ersten Übung werden ebenso benutzt, womit klar gezeigt wird, wie die verschiedenen Übungen im Laufe des Semesters aufeinanderbauen.

Bei der Aufgabenverteilung haben wir zuerst beachtet, welche Klassen von welchen anderen abhängig sind und wer mit welchem Themengebiet am besten klar kommt. Am Ende haben wir beschlossen, die Übung in drei Teilaufgaben zu verteilen, jede davon aus zwei kleineren Problemen bestehend.

2 Teilaufgabe 1

In der ersten Teilaufgabe werden zwei wichtige Komponenten des Raytracers entwickelt - Klassen, die einen Strahl und eine Kamera darstellen. Die Vorbereitung dafür hat nicht viel Zeit in Anspruch genommen, da diese Themen ausführlich in der Vorlesung behandelt wurden und gut strukturiert in den Folien des Dozenten zu finden sind.

Die `Ray`-Klasse verfügt über zwei öffentliche Parameter - `Point3 o` - den Ursprungspunkt des Strahls, `Vector3 d` - die Richtung des Strahls, einen Konstruktor, sowie zwei Methoden - `Point3 at(double t)`, die den Punkt zurückliefert, der auf dem Strahl liegt und den Abstand `t` zum Ursprung hat und `double tOf(Point3 p)`, welche einen Punkt als Parameter annimmt und den entsprechenden Wert `t` zurückgibt.

Der zweite Teil dieser Aufgabe hat mehr Zeit gekostet. Ausgehend von einer abstrakten Basisklasse `Camera` sollte man zwei andere implementieren, die jeweils eine perspektivische und eine orthographische Kamera repräsentieren. Zu den Attributen der Basisklasse zählen die Position `e` der Kamera, ihre Blickrichtung `g`, der Up-Vektor `t`. Darüber hinaus verfügen die erbenden Klassen über ein weiteres Attribut - bei der `OrthographicCamera` ist das der Skalierungsfaktor `s`, bei der `PerspectiveCamera` - der Öffnungswinkel. Außerdem werden in der abstrakten Klasse die Achsen des Kamera-Koordinatensystem anhand der im Konstruktor übergebenen Werte berechnet. Die Programmierung dieser Klassen war an sich nichts besonders Schwieriges, aber später (in der Testklasse) tauchten einige Probleme auf, manche von denen leider nicht behoben werden konnten. Die Versuche, diese Probleme zu beseitigen, haben eigentlich die meiste Zeit gekostet.

- **Teammitglied:** Robert
- **Einlesen:** 30 Minuten
- **Programmieren:** 1-2 Stunden
- **Fehler beheben:** 4 Stunden

3 Teilaufgabe 2

Zu dieser Teilaufgabe gehören die Implementierung einer Farb-Klasse und die Schnittberechnung eines Strahls mit verschiedenen Geometrieformen (`Plane, Triangle, Sphere, AxisAlignedBox`). Der Kommilitone, der diese Aufgabe übernommen hat, begann als erster mit der Bearbeitung. In den zu implementiernden Klassen werden Objekte aus der ersten Teilaufgabe benötigt, die noch nicht vollständig geschrieben waren, was aber kein Problem verursachte, da die Namen der Klassen und deren Methoden schon bekannt waren und "aufgerufen" werden konnten, ohne dass es nötig war, auf die Fertigstellung der ersten Teilaufgabe zu warten. So zum Beispiel konnte in der `hit()`-Methode der `Triangle`-Klasse den Übergabeparameter `Ray r` verwenden, noch bevor dieser geschrieben war. Das erinnert einigermaßen an die Bequemlichkeit eines Interfaces.

Die `Color`-Klasse war einfach zu implementieren, aber die Multiplikation mit einer anderen Farbe hatten wir in der Vorlesung nicht richtig behandelt und online konnte auch nicht viel zum Thema gefunden werden. Vielleicht wird die Methode nochmal verändert werden müssen, wenn wir wissen, wofür genau man sie verwenden wird.

Die abstrakte `Geometry`-Klasse, sowie `Hit` und `Plane` waren am einfachsten zu programmieren.

Bei `Triangle` musste man erst einmal das baryzentrische Kordinatensystem verstehen und es danach korrekt andwenden, um damit den Schnittpunkt zwischen dem Ray und dem Triangle zu berechnen.

Die `Sphere` war die Klasse mit der schwierigsten Berechnung. Glücklicherweise gab es dort nicht ganz so viel zu korregieren.

Die `AxisAlignedBox` hat sehr gut funktioniert. Die flexible Speicherstruktur List wurde (bei der Variable `facingSides`) vewendet, weil es vorkommen kann, das man mal weniger als drei Seiten sieht (zB. wenn man direkt davor steht).

Das Kommentieren der Variablen `lbf` und `run` war schwierig, weil wir wussten, was sie jeweils darstellen, aber nicht wofür jeder Buchstabe in der Abkürzung steht.

- **Teammitglied:** Benjamin
- **Einlesen:** 30 Minuten
- **Programmieren:** 5-6 Stunden
- **Fehler beheben:** 2-3 Stunden
- **Helfen:** 1 Stunde

4 Teilaufgabe 3

Am Anfang wurde die Klasse `World` erstellt. Sie enthält eine Menge mit geometrischen Objekten, die auf einer Fläche dargestellt werden. In der Methode `hit()` werden die `hit()`-Methoden der in der Menge enthaltenen Objekte ausgeführt und somit ein möglicher Schnittpunkt berechnet. Gibt es einen Schnittpunkt, gibt die `hit()`-Methode der Objekte ein `Hit`-Objekt zurück, welches in einem weiteren `HashSet` gespeichert wird. Aus diesem Set wird das `Hit`-Objekt mit der kleinsten positiven Variablen `t` ermittelt. Dieses Objekt wird von der

`hit()`-Methode zurückgegeben. Gab es keinen Hit wird null zurückgegeben. Gab es einen Hit wird an der Stelle das Pixel in der Farbe des Objektes gezeichnet, andernfalls wird die Hintergrundfarbe des `World`-Objektes verwendet.

Als Letztes wurde die Klasse `Raytracer` geschrieben, die das Zeichen der Pixel impelemntiert. Es wird über alle Pixel des Bildes iteriert. Dabei wird von der übergebenen Kamera für jeden Pixel ein Strahl erzeugt und getestet, ob sich dieser Strahl mit einer Geometrie schneidet(mit einem Aufruf der Methode `hit()` der Klasse `World`). In der Methode `paintComponent()` wird anschließend das Bild gezeichnet. In der Klasse `RaytracerTest` wird ein Fenster erzeugt und die Variablen initialisiert, die dem `Raytracer` übergeben werden. Anschließend wird in diesem Fenster der `Raytracer` geladen und das Bild gezeichnet.

Bei der Implementierung der Klassen gab es eigentlich keine Probleme. Lediglich in der Methode `hit()` der Klasse `World` gab es Schwierigkeiten bei der Berechnung des kleinsten positiven `t`. Da uns das Verständnis der theoretischen Grundlagen zur Schnittberechnung etwas schwerfällt, hat die Bearbeitung der Aufgabe doch länger gedauert.

- **Teammitglied:** Clemens
- **Einlesen:** 2 Stunden
- **Bearbeitung:** 6 Stunden

5 Probleme

Bei der `AxisAlignBox` ist ein komisches Rauschen entstanden, welches durch Nachkommastellenungenaugkeit verursacht worden war. Dieses Problem konnte durch die Hilfe eines Kommilitonen entfernt werden.

Bei der Darstellung der verschiedenen Geometrien sind wir auf Schwierigkeiten gestoßen, obwohl wir alle beteiligten Klassen mehrmals auf Fehler geprüft und verschiedene Lösungswege versucht hatten. Das Problem besteht nämlich darin, dass einige Objekte nicht wie in der Aufgabenstellung dargestellt werden konnten. Dieses Problem haben wir teilweise gelöst, jedoch ist bei dem Dreieck-Beispiel immer noch eine Ungenauigkeit zu beobachten.

6 Quellen und Literatur

- Computergraphik I Folien – Stephan Rehfeld
- Ray Tracing from the Ground Up – Kevin Suffern