

CGI: BERICHT ZU AUFGABE 2, RAYTRACER

VON SEBASTIAN DASSÉ, MAX NOVICHKOV, SIMON LISCHKA

1. AUFGABENSTELLUNG

- 1.1. Implementierung der beiden Kameras `OrthographicCamera` und `PerspectiveCamera`.
- 1.2. Schreiben einer `Color`-Klasse.
- 1.3. Entwerfen einer abstrakten Geometrie-Superklasse `Geometry` und der entsprechenden Unterklassen `Plane`, `AxisAlignedBox`, `Triangle` und `Sphere` sowie eines `Hit`-Objekts.
- 1.4. Entwerfen einer `Welt`-Klasse, die alle Objekte der Szene enthält
- 1.5. Entwerfen der `Raytracer`-Klasse, die über die Objekte der `Welt` iteriert.
- 1.6. Testen der `Geometry`-Objekte durch entsprechende Beispielkonfigurationen.

2. LÖSUNGSSTRATEGIEN

- 2.1. Die Klassen wurden zunächst als leere Vorlagen generiert.
- 2.2. Die entsprechenden Klassen wurden zunächst auf Papier gerechnet und dann gemeinsam im linearen Zeitablauf implementiert.
- 2.3. `Raytracer` und `View` Objekte wurden möglichst generisch modularisiert.

3. BESONDERE PROBLEME

- 3.1. Einsetzen der Werte aus der selbsterstellten `Color`-Klasse in das `WriteableRaster`.
- 3.2. Implementierung der `AxisAlignedBox`.

4. IMPLEMENTIERUNG

- 4.1. Die Kameras, die Klassen `Color` und `Hit` sowie die Geometrie-Klassen wurden anhand der im Unterricht vorgegebenen Methodik und dem Buch *Ray Tracing From Ground Up* von Kevin Suffern umgesetzt.
- 4.2. Der `Raytracer` gibt mit der Methode `trace()` ein `BufferedImage`-Objekt aus, das mit der UI-Klasse `ShowImage` auf einem `Canvas` dargestellt wird. Zur Bequemen Erzeugung der Graphikobjekte wurde die Factoryklasse `raytracer.tests.graphical.Factory` erstellt.

5. ZEITBEDARF

- 5.1. Der Zeitbedarf wurde bei allen Teammitgliedern mit jeweils 25 Stunden angegeben. Dabei wurde viel Zeit für mathematische Überlegungen aufgewandt.

Date: 24.11.2013.