BERGISCHE UNIVERSITÄT WUPPERTAL
FAKULTÄT FÜR MATHEMATIK
UND NATURWISSENSCHAFTEN

BERGISCHE
UNIVERSITÄT
WUPPERTAL

Angewandte Informatik
Dr. Martin Galgon
M.Sc. Jose Jimenez

Bildgenerierung

Wintersemester 2023 / 2024

Übungsblatt 11

Aufgabe 34 (Raytracing nach Whitted)

Implementieren Sie rekursives Raytracing nach Algorithmus 10.5 und 10.6 des Skripts und verwenden Sie für das Auffinden der vom Strahl getroffenen Objekte den „naiven Ansatz“ von Seite 10-10.

Im Verzeichnis `/home/bildgen/Aufgaben/raytracing-1` finden Sie ein Archiv mit einem Rahmenprogramm, in dem Sie die Implementierung vornehmen können. Sie müssen nur Änderungen an der Datei `raytracer.cc` vornehmen. Compiliert wird das Rahmenprogramm durch Eingabe von `make`.

Folgende Methoden, Objekte und Strukturen werden für die Implementierung benötigt:

- Die Klasse `Vector3d` stellt einen Vektor mit drei `double`-Komponenten dar und besitzt unter anderem die Methoden
  - `cross(const Vector3d&)` zur Berechnung des Kreuzproduktes mit einem weiteren Vektor,
  - `dot(const Vector3d&)` zur Berechnung des Innenproduktes mit einem weiteren Vektor,
  - `norm()` liefert die euklidische Norm des Vektors und
  - `cwiseProduct(const Vector3d&)` kann zur komponentenweisen Multiplikation mit einem weiteren Vektor verwendet werden.

  Außerdem sind die Operatoren `+`, `-`, `+=` sowie `-=` zur Addition und Subtraktion von Vektoren, die Operatoren `*` und `*=` zur Multiplikation mit Skalaren und die Operatoren `/` und `/=` zur Division von Vektoren durch Skalare überladen.
- Die Klasse `Image` enthält das erzeugte Bild und stellt die Methoden
  - `getWidth()` und `getHeight()`
  - sowie `setPixel(int x, int y, const Vector3d &rgb)`

  bereit. Verwenden Sie die Methode `Raytracer::clampCol(Vector3d&)` vor dem Aufruf von `setPixel`, damit die Farbwerte auf das Intervall [0; 1] beschränkt werden.
- Die Klasse `Scene` enthält sämtliche die Szene betreffenden Informationen. Sie benötigen folgende Methoden und Members:
  - `getCamera()` liefert ein Objekt der Klasse `Camera`, die weiter unten erläutert wird.
  - `objects` ist ein Vektor, der Pointer auf alle in der Szene vorhanden Objekte vorhält.
  - `getBackground()` liefert die Hintergrundfarbe der Szene.
  - `lights` ist ein Vektor, der Pointer auf alle in der Szene vorhanden Lichtquellen vorhält.

- Sie benötigen folgende Methoden der Klasse `Camera`:
  - `getPosition()` liefert die Position der Kamera als `Vector3d`.
  - `getDirection()` liefert die Blickrichtung der Kamera als `Vector3d`.
  - `getUp()` liefert den Up-Vektor der Kamera als `Vector3d`.
  - `getHorAngle()` liefert den horizontalen Blickwinkel der Kamera als `double`.
- Die verschiedenen Objekt-Typen sind von der Klasse `Object` abgeleitet und Sie benötigen:
  - `isHitBy(const Ray&)` gibt eine Instanz der Struktur `HitInfo` zurück.
- Alle Lichtquellen sind Instanzen der Klasse `Light` und Sie benötigen:
  - `visibleFrom(const Scene&, const Vector3d&)` bestimmt, ob die Lichtquelle vom übergebenen Punkt aus sichtbar ist und gibt die Information in einer Instanz der Struktur `VisibleInfo` zurück.
- Die Struktur `Ray` stellt einen Strahl mit Ausgangspunkt und Richtung dar. Alle Informationen dazu finden Sie in der Datei `ray.h`.
- Die Struktur `HitInfo` bündelt die Schnittpunkt-Informationen eines Strahls mit einem Objekt. Alle Informationen dazu finden Sie in der Datei `hitinfo.h`.
- Die Struktur `VisibleInfo` bündelt die Informationen, ob eine Lichtquelle von einem Punkt aus sichtbar ist. Alle Informationen dazu finden Sie in der Datei `visibleinfo.h`.

Aufgabe 35 (Raytracing nach Whitted - Teil 2)

Erweitern Sie die Implementierung zum rekursiven Raytracing aus Aufgabe 34 um Lichtbrechung und die Darstellung von Tetraedern.

Im Verzeichnis `/home/bildgen/Aufgaben/raytracing-2` finden Sie ein ein Rahmenprogramm, in dem Sie die Implementierung vornehmen können. Sie müssen nur Änderungen an den markierten Stellen in den Dateien `raytracer.cc`, `triangle.cc` und `tetraeder.cc` vornehmen. Orientieren Sie sich für die Implementierung der Klassen `Tetraeder` und `Triangle` an den Klassen `Box` und `Parallelogram`. In der Datei `README` wird beschrieben, wie Sie das Rahmenprogramm compilieren können.

Zusätzlich zu den erläuterten Methoden, Objekten und Strukturen aus Aufgabe 34 benötigen Sie die Methoden

- `Object::getRefraction()`, die den Brechungsindex für das Medium des Objekts als `double` liefert, und
- `Object::getPigment()`, die einen `Vector3d` zurückgibt und die Durchlässigkeit des Objekts für die drei Farbkomponenten darstellt.

Abgabe: Do., 24.01.2024, 13:15 Uhr

Senden Sie Ihre Lösungen der Theorie-Aufgaben und Ihre Programme per E-Mail an `bildgen@studs.math.uni-wuppertal.de` .