Übungen - Bildgenierung

Übung 11.

Jose Jimenez

Angewandte Informatik
Bergische Universität Wuppertal

January 25, 2023

Table of Contents

Raytracing nach Whitted

- Die Struktur Ray
- Die Struktur HitInfo
- Die Struktur VisibleInfo

Raytracing nach Whitted

- **Die Struktur Ray**
- Die Struktur HitInfo
- Die Struktur VisibleInfo

```
typedef struct {
 Vector3d origin;
 Vector3d direction;
   Object* insideObject {nullptr};   //Dieses Objekt wird Ignoriert.
 } Ray;
```

Die Struktur Ray stellt einen Strahl mit Ausgangspunkt und Richtung dar. Alle Informationen dazu finden Sie in der Datei ray.h.

Raytracing nach Whitted

- Die Struktur Ray
- **Die Struktur HitInfo**
- Die Struktur VisibleInfo

```
typedef struct {
  bool hit {false};
  // rayparam:   position = origin + rayparam*direction : linie(t)
  // -> für Entfernungsbestimmung.
double rayparam {std::numeric_limits<double>::max()};
Vector3d position {0,0,0};
Vector3d normal {0,0,0};
  Object *object {nullptr};
} HitInfo;
```

Die Struktur HitInfo bündelt die Schnittpunkt-Informationen eines Strahls mit einem Objekt. Alle Informationen dazu finden Sie in der Datei hitinfo.h.

Raytracing nach Whitted

- Die Struktur Ray
- Die Struktur HitInfo
- **Die Struktur VisibleInfo**

```
typedef struct {
   bool visible {false};
   Vector3d direction {0,0,0};  /* Richtung vom Punkt
                                   zur Lichtquelle, normiert*/
   double distance {0};
   Vector3d pos {0,0,0};
   Light *light {nullptr};
 } VisibleInfo;
```

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.

Raytracing nach Whitted

Zunächst müssen wir alle Eigenschaften der Szene und des Bildes abrufen.

Raytracing nach Whitted

Zunächst müssen wir alle Eigenschaften der Szene und des Bildes abrufen. d.h

- Ecke und spannende Vektoren der Projektionsebene bestimmen.

Raytracing nach Whitted

Zunächst müssen wir alle Eigenschaften der Szene und des Bildes abrufen. d.h

- Ecke und spannende Vektoren der Projektionsebene bestimmen.
  - Alle Kamerawerte.
  - Alle Ebenes.

Raytracing nach Whitted

Zunächst müssen wir alle Eigenschaften der Szene und des Bildes abrufen. d.h

- Ecke und spannende Vektoren der Projektionsebene bestimmen.
  - **Alle Kamerawerte.**
  - Alle Ebenes.

```
Vector3d campos = scene.getCamera().getPosition();
Vector3d camdir = scene.getCamera().getDirection();
Vector3d up     = scene.getCamera().getUp(); // VUP = Aufwärtsrichtung
double horangle = scene.getCamera().getHorAngle();
double aspect   = static_cast<double>(image.getWidth())
                                        /image.getHeight();
```

Lass uns sehen was wir machen, d.h. wie erzeugen wir die Ebenen?

Raytracing nach Whitted

Zunächst müssen wir alle Eigenschaften der Szene und des Bildes abrufen. d.h

- Ecke und spannende Vektoren der Projektionsebene bestimmen.
  - Alle Kamerawerte.
  - **Alle Ebenes.**

```
Vector3d campos = scene.getCamera().getPosition();
Vector3d camdir = scene.getCamera().getDirection();
Vector3d up     = scene.getCamera().getUp();
double horangle = scene.getCamera().getHorAngle();
double aspect   = static_cast<double>(image.getWidth())/image.getHeight(

Vector3d planeright = camdir.cross(up);
Vector3d planedown  = camdir.cross(planeright);

camdir     /= camdir.norm();
planeright /= planeright.norm();
planedown  /= planedown.norm();
```

Raytracing nach Whitted

```
Vector3d campos = scene.getCamera().getPosition();
Vector3d camdir = scene.getCamera().getDirection();
Vector3d up     = scene.getCamera().getUp();
double horangle = scene.getCamera().getHorAngle();
double aspect   = static_cast<double>(image.getWidth())/image.getHeight(

Vector3d planeright = camdir.cross(up);
Vector3d planedown  = camdir.cross(planeright);

camdir     /= camdir.norm();
planeright /= planeright.norm();
planedown  /= planedown.norm();
```

Wir versuchen, die Koordinaten der Ecke zu berechnen. **Wie?**

Und damit können wir alle Pixel durchgehen.

Raytracing nach Whitted

```
Vector3d campos = scene.getCamera().getPosition();
Vector3d camdir = scene.getCamera().getDirection();
Vector3d up     = scene.getCamera().getUp();
double horangle = scene.getCamera().getHorAngle();
double aspect   = static_cast<double>(image.getWidth())/image.getHeight(

Vector3d planeright = camdir.cross(up);
Vector3d planedown  = camdir.cross(planeright);

camdir     /= camdir.norm();
planeright /= planeright.norm();
planedown  /= planedown.norm();
```

Wir versuchen, die Koordinaten der Ecke zu berechnen. **Wie?**
Und damit können wir alle Pixel durchgehen.
Wir haben den **Horizontalwinkel**.
Lass uns es auf der Tafel sehen.

Raytracing nach Whitted

Wir versuchen, die Koordinaten der Ecke zu berechnen. **Wie?**
Und damit können wir alle Pixel durchgehen.
Wir haben den **Horizontalwinkel**.

```
double horangle = scene.getCamera().getHorAngle();
double aspect   = static_cast<double>(image.getWidth())/image.getHeight(

Vector3d planeright = camdir.cross(up);
Vector3d planedown  = camdir.cross(planeright);

camdir     /= camdir.norm();
planeright /= planeright.norm();
planedown  /= planedown.norm();

double pi = M_PI;
planeright *= tan(horangle * pi / 360.0
planedown  *= tan(horangle * pi / 360.0) / aspect;

Vector3d topleft = campos + camdir - planeright - planedown;
```

Raytracing nach Whitted

Ok, wir haben die Ecke.
Wir mussen dann alle Punkte auf der Projektionsebene erzeugen und Strahlen verfolgen.

Raytracing nach Whitted

Ok, wir haben die Ecke.
Wir mussen dann alle Punkte auf der Projektionsebene erzeugen und Strahlen verfolgen.
Wir...

- initialisieren ein "leeres" Bild, (2d vector von Vector3d),
- machen Raytracing für jedes Pixel.

Raytracing nach Whitted

Ok, wir haben die Ecke.
Wir mussen dann alle Punkte auf der Projektionsebene erzeugen und Strahlen verfolgen.
Wir...

- **initialisieren ein "leeres" Bild,** (2d vector von Vector3d),
- machen Raytracing für jedes Pixel.

```
vector<vector<Vector3d>> tmpImage;
tmpImage.resize( image.getHeight() ); //???
```

Raytracing nach Whitted

Ok, wir haben die Ecke.
Wir mussen dann alle Punkte auf der Projektionsebene erzeugen und Strahlen verfolgen.
Wir...

- **initialisieren ein "leeres" Bild,** (2d vector von Vector3d),
- machen Raytracing für jedes Pixel.

```
int w, h;

w = image.getWidth();
h = image.getHeight();

vector<vector<Vector3d>> tmpImage;
tmpImage.resize( h );
for (int y = 0; y < h; y++) {
  tmpImage[y].resize( w, Vector3d(0, 0, 0) );
}
```

Done

Raytracing nach Whitted

Wir...

- initialisieren ein "leeres" Bild, (2d vector von Vector3d)
- **machen Raytracing für jedes Pixel**.
  1. dafür brauchen wir die Maße der Pixels.
  2. wir wollen die Mitte jedes Pixels bekommen
  3. alle Punkte auf der Projektionsebene erzeugen und Strahlen verfolgen

Raytracing nach Whitted

Wir...

- initialisieren ein "leeres" Bild, (2d vector von Vector3d),
- **machen Raytracing für jedes Pixel**.
  1. **dafür brauchen wir die Maße der Pixels.**
  2. **wir wollen die Mitte jedes Pixels bekommen**
  3. alle Punkte auf der Projektionsebene erzeugen und Strahlen verfolgen

```
// Pixelmaße
double w1 = 1.0 / w;
double h1 = 1.0 / h;
double w1_halbe = 0.5 * w1;
double h1_halbe = 0.5 * h1;
```

Raytracing nach Whitted

Wir...

- initialisieren ein "leeres" Bild, (2d vector von Vector3d),
- **machen Raytracing für jedes Pixel**.
  1. dafür brauchen wir die Maße der Pixels.
  2. wir wollen die Mitte jedes Pixels bekommen
  3. **alle Punkte auf der Projektionsebene erzeugen und Strahlen verfolgen**

```
Ray currentray;
currentray.origin = campos;
for (int y = 0; y < h; y++) {
  for (int x = 0; x < w; x++) {
  currentray.direction = ...
  }
}
```

Raytracing nach Whitted

Wir...

- initialisieren ein "leeres" Bild, (2d vector von Vector3d),
- **machen Raytracing für jedes Pixel**.
  1. dafür brauchen wir die Maße der Pixels.
  2. wir wollen die Mitte jedes Pixels bekommen
  3. **alle Punkte auf der Projektionsebene erzeugen und Strahlen verfolgen**

```
Ray currentray;
currentray.origin = campos;
for (int y = 0; y < h; y++) {
  for (int x = 0; x < w; x++) {
  currentray.direction =  topleft - campos
                        + 2 * planeright * ( x * w1 + w1_halbe )
                        + 2 * planedown  * ( y * h1 + h1_halbe );
  }
}
```

Raytracing nach Whitted

Wir...

- initialisieren ein "leeres" Bild, (2d vector von Vector3d),
- **machen Raytracing für jedes Pixel**.

```
Ray currentray;
currentray.origin = campos;
for (int y = 0; y < h; y++) {
  for (int x = 0; x < w; x++) {
  currentray.direction =  topleft - campos
                        + 2 * planeright * ( x * w1 + w1_halbe )
                        + 2 * planedown  * ( y * h1 + h1_halbe );

  tmpImage[y][x] +=  raytrace( scene, currentray, 1 );
  }
}
```

Done? Fast... Die Farbe-Werte muss zwischen 0 und 1.

Raytracing nach Whitted

Wir...

- initialisieren ein "leeres" Bild, (2d vector von Vector3d),
- **machen Raytracing für jedes Pixel**.

```
Ray currentray;
currentray.origin = campos;
for (int y = 0; y < h; y++) {
  for (int x = 0; x < w; x++) {
  currentray.direction =  topleft - campos
                        + 2 * planeright * ( x * w1 + w1_halbe )
                        + 2 * planedown  * ( y * h1 + h1_halbe );

  tmpImage[y][x] +=  raytrace( scene, currentray, 1 );

  Vector3d col;

  col = tmpImage[y][x] ;
  clampCol(col);
  image.setPixel( x, y, col );
  }
}
```

/27

Raytracing nach Whitted

Jetz, wir implementieren Algorithmus 10.6.

```
Vector3d Raytracer::raytrace( const Scene &scene, const Ray &ray, int dep,
                              Object *origin ){
  Vector3d rgbCol;
```

Raytracing nach Whitted

Jetz, wir implementieren Algorithmus 10.6.

```
Vector3d Raytracer::raytrace( const Scene &scene, const Ray &ray, int dep,
                              Object *origin ){
  Vector3d rgbCol;
```

Wir tun es in drei Schritten.

0 Wir Kriegen das HitInfo des Strahls.

1. Rechnnen diffuser und winkelabhängiger "direkter" Anteil.
2. Spiegelung
3. Lichtberechnung

Raytracing nach Whitted

```
Vector3d Raytracer::raytrace( const Scene &scene, const Ray &ray, int dep,
                              Object *origin ){
   HitInfo hi;
    for (auto o: scene.objects) {
      if (o.get() == origin)
        continue;

      HitInfo hitest = o->isHitBy( ray );
      if ( hitest.hit
           && (hitest.rayparam < hi.rayparam)
           && (hitest.rayparam >= 0) ) {
        hi = hitest;
      }
    }
```

Ok, wir haben alle hit Information. Dann, konnen wir mit Algorithmus 10.6 arbeiten.