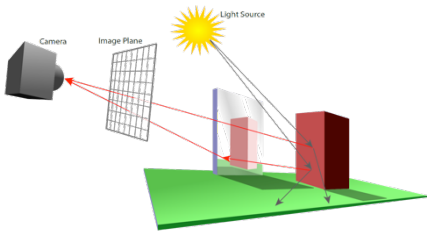
BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Computergrafik II
Raytracing-Implementierung



Bachelor Medieninformatik
Wintersemester 2011

Prof. Dr.-Ing. Hartmut Schirmacher
http://schirmacher.beuth-hochschule.de
hschirmacher@beuth-hoschule.de

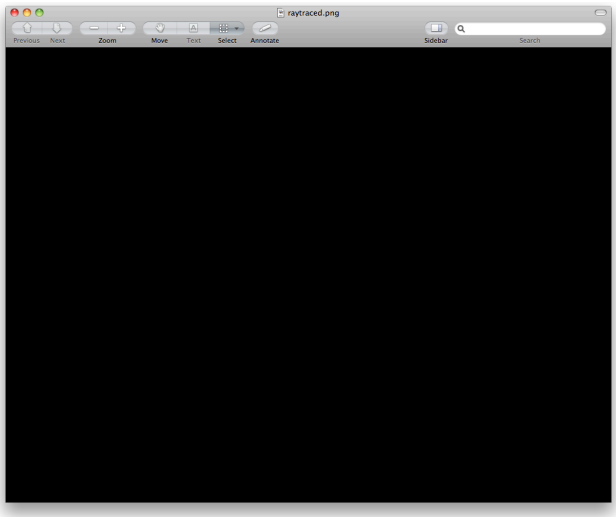
Gliederung

- Rückblick Raytracing Teil I
  - Wiederholung Raytracing-Algorithmus
  - Hinweis: Fehler auf vorherigen Folien (Kamermodell)
- Wie schreibe ich einen Raytracer?
  - Klassenentwurf
  - Sonstige Implementierungshinweise

Pseudocode des Raytracing Main Loops

```
// calculate color by tracing rays from the camera into the scene
raytrace(scene, camera, image)
{
  // loop over all pixels in image
  for all (i,j) from (0,0) to (image.width-1, image.height-1) do
  {
    // generate ray from eye point through pixel center
    ray ← generate_ray(i, j, image.width, image.height, camera);

    // find first intersection point with scene objects
    hit ← intersect(ray, scene);

    // calculate light intensity/color at hit point
    color ← shade(hit, ray, scene);

    // set corresponding pixel color in image
    image.pixel(i,j) ← color;
  }
}
```

Das Kameramodell aus der letzten Vorlesung enthielt einen Fehler!



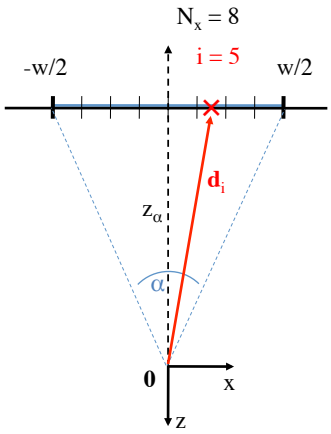
Ergebnisbild unter Verwendung der Kamera aus der Vorlesung

Eine einfache virtuelle Kamera (2)

- Pixelkoordinaten in der Bildebene
  - Bild umfaßt (N_x , N_y) Pixel
  - Achtung: quadratische Pixel gewünscht → $N_x : N_y = w : h$
  - (x,y)-Koordinaten des Mittelpunkts von Pixel (i,j):

$$x(i) = -\frac{w}{2} + (i + 0.5)\frac{w}{N_x}$$

$$y(j) = -\frac{h}{2} + (j + 0.5)\frac{h}{N_y}$$



- Vektor Auge → Pixel (i,j)

$$\mathbf{d}_{i,j} = \begin{pmatrix} x(i) \\ y(i) \\ -z_\alpha \end{pmatrix}$$

Klassenentwurf für einen Raytracer

Vorbemerkungen / Disclaimer

- Die folgenden Hinweise sind nur eine von unzähligen Möglichkeiten, einen Raytracer zu implementieren
- Die Hinweise sollen den Einstieg in die Übungsaufgabe erleichtern, nicht jedoch Ihre Kreativität einschränken!

Im Raytracer verwendete Objekte

```
// calculate color by tracing rays from the camera into the scene
raytrace(scene, camera, image)
{
  // loop over all pixels in image
  for all (i,j) from (0,0) to (image.width-1, image.height-1) do
  {
    // generate ray from eye point through pixel center
    ray ← generate_ray(i, j, image.width, image.height, camera);

    // find first intersection point with scene objects
    hit ← intersect(ray, scene);

    // calculate light intensity/color at hit point
    color ← shade(hit, ray, scene);

    // set corresponding pixel color in image
    image.pixel(i,j) ← color;
  }
}
```

Kern des eigentlichen Raytracers

Objekte:

Camera

Scene

Ray

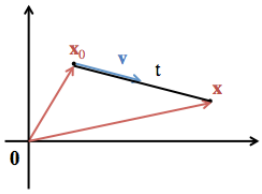
Hit

Klasse Ray

Ray

origin: Vector
direction: Vector

getOrigin(): Vector
getNormDirection (): Vector
getPoint(float): Vector



- getOrigin()
  - liefert Ausgangspunkt $\mathbf{x}_0$ zurück
- getNormalizedDirection()
  - liefert Richtungsvektor $\mathbf{v}$ zurück
  - garantiert, dass dieser normiert ist
- getPoint(float t)
  - liefert für ein gegebenes t die 3D-Koordinaten $\mathbf{x}(t)$ des entsprechenden Punktes auf dem Strahl

$$\mathbf{x}(t) = \mathbf{x}_0 + t\mathbf{v}$$

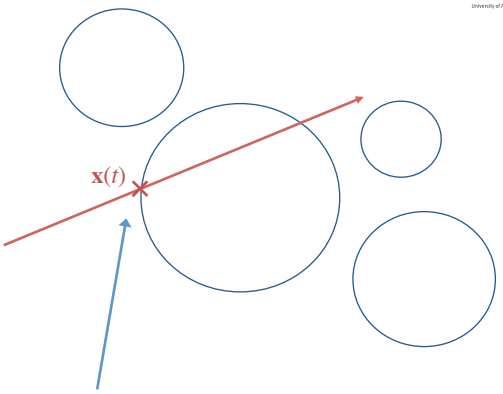
- Ausgangspunkt $\mathbf{x}_0$
- Richtung $\mathbf{v}$
  - Wird oft normalisiert: $|\mathbf{v}| = 1$
- Parameter $t \in [-\infty, \infty]$

Klasse **Hit**

Hit

ray: Ray
shape: Shape
rayParam: float;

getRay(): Ray
getShape(): Shape
getRayParam(): float



- getRay()
  - der Strahl, der das Objekt getroffen hat
- getShape()
  - das vorderste getroffene Objekt
- getRayParam()
  - Der Strahl-Parameter t für den vordersten Schnittpunkt

Ein Treffer, beschrieben durch:
- Welcher Strahl hat welches Objekt geschnitten?
- Wo auf dem Strahl liegt der Schnittpunkt?

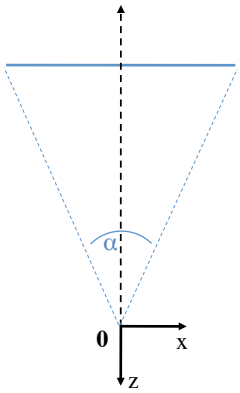
Klasse **Camera**

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Camera

eyePoint: Vector
viewingDirection: Vector
fieldOfViewX: float

- Kamera ist definiert durch
  - Augpunkt
  - Blickrichtung
  - Öffnungswinkel (Kamera-Ojektiv)



Klasse **Camera**

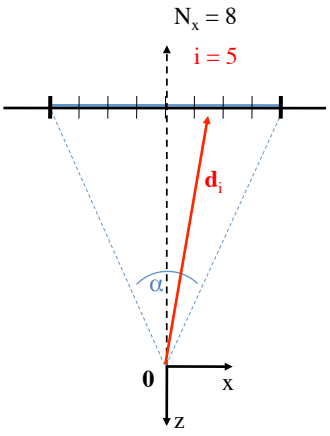
BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Camera

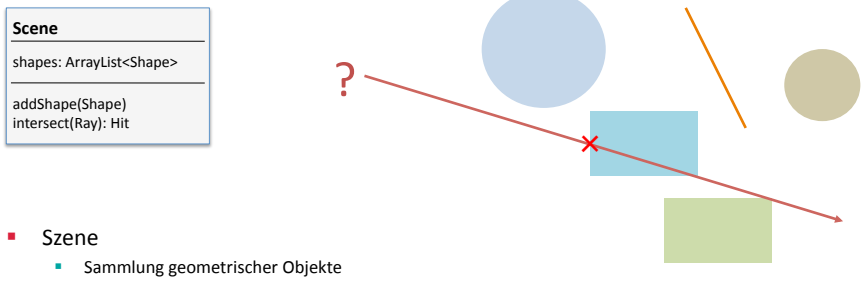
eyePoint: Vector
viewingDirection: Vector
fieldOfViewX: float

generateRay (...): Ray

- Kamera ist definiert durch
  - Augpunkt
  - Blickrichtung
  - Öffnungswinkel (Kamera-Ojektiv)
- Die Kamera kann das Berechnen der Primärstrahlen übernehmen
  - generateRay(i,i,N_x,N_y) liefert einen Strahl durch den Pixel (i,j) für ein Bild der Auflösung (Nx.Ny)



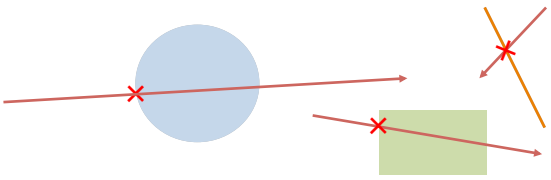
Klasse **Scene**



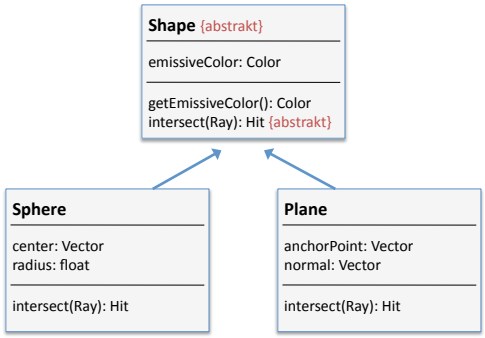
- Szene
  - Sammlung geometrischer Objekte
  - Objekte können der Szene hinzugefügt werden
  - Ein Strahl kann in die Szene „gesendet“ werden, um zu ermitteln, welches der vorderste Treffer mit einem in der Szene enthaltenen Objekt ist

Geometrische Objekte (Shapes)

- Objekttypen
  - Sphere
  - Plane
  - Axis Aligned Box
  - ... (erweiterbar!!!)



- Gemeinsamkeiten:
  - Besitzen eine eigene Farbe
  - Man möchte den Schnittpunkt des Objekts mit einem Strahl ermitteln
- Unterschiede
  - Daten zur Repräsentation der Geometrie
  - Wie berechne ich denn tatsächlich den Schnittpunkt?



Die äußere Schleife des Raytracing-Algorithmus

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

```
// calculate color by tracing rays from the camera into the scene
raytrace(scene, camera, image)
{
  // loop over all pixels in image
  for all (i,j) from (0,0) to (image.width-1, image.height-1) do
  {
    // generate ray from eye point through pixel center
    ray ← generate_ray(i, j, image.width, image.height, camera);

    // find first intersection point with scene objects
    hit ← intersect(ray, scene);

    // calculate light intensity/color at hit point
    color ← shade(hit, ray, scene);

    // set corresponding pixel color in image
    image.pixel(i,j) ← color;
  }
}
```

Bekannte Komponente aus dem Warmup!

`class ImageGenerator`

Pseudocode des Raytracing Main Loops

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

```
// calculate color by tracing rays from the camera into the scene
raytrace(scene, camera, image)
{
  // loop over all pixels in image
  for all (i,j) from (0,0) to (image.width-1, image.height-1) do
  {
    // generate ray from eye point through pixel center
    ray ← generate_ray(i, j, image.width, image.height, camera);

    // find first intersection point with scene objects
    hit ← intersect(ray, scene);

    // calculate light intensity/color at hit point
    color ← shade(hit, ray, scene);

    // set corresponding pixel color in image
    image.pixel(i,j) ← color;
  }
}
```

Kern des eigentlichen Raytracers

`class Raytracer`

→ *Welches Interface muß der* ***`Raytracer`*** *haben, damit er zusammen mit dem* ***`ImageGenerator`*** *funktioniert?*

ImageGenerator, Painter und Raytracer

ImageGenerator

generate(Painter, ...)

```
// Set a color for each pixel.
for (int i = 0; i < width; i++)
  for (int j = 0; j < height; j++)
    image.setRGB( i, j, painter.pixelColorAt(i, j, width, height) );
```

Painter {abstrakt}

pixelColorAt(): Color {abstrakt}

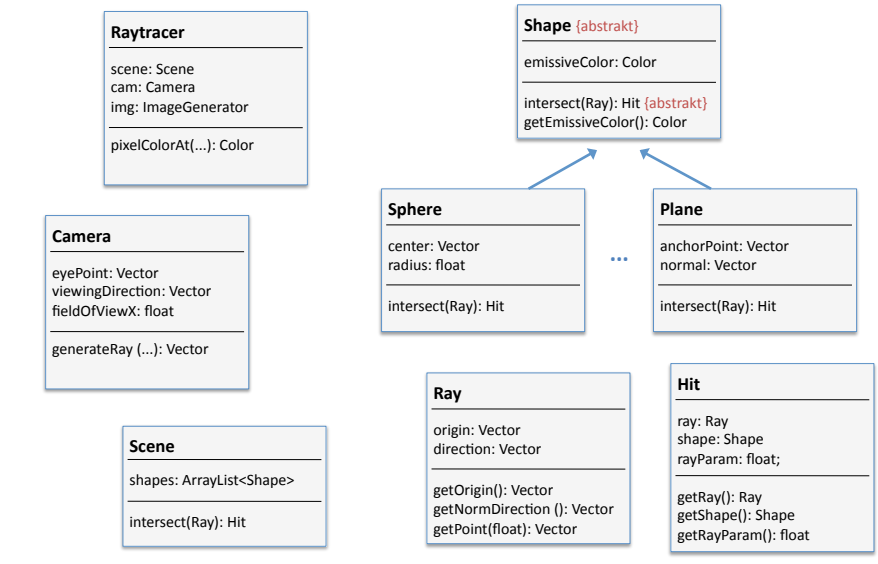
→ *In dieser Aufgabe muß der* ***`Raytracer`*** *als* ***`Painter`*** *fungieren. D.h. er berechnet für jedes Pixel (i,j) dessen Farbe.*

Raytracer

pixelColorAt(...): Color

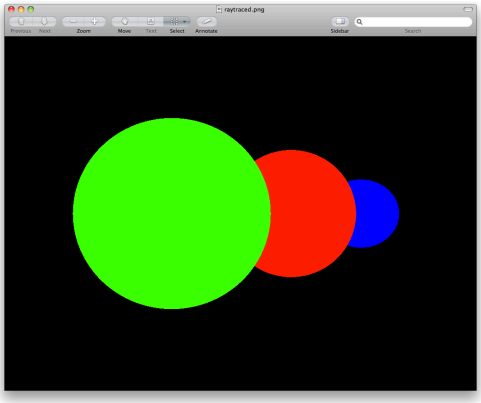
```
public class Raytracer implements Painter {

    // this is the main method of the raytracer:
    // for a given pixel (i,j) calculate the pixel's color
    @Override
    public Color pixelColorAt(int i, int j, int width, int height) {
```

Klassenentwurf für einen einfachen Raytracer



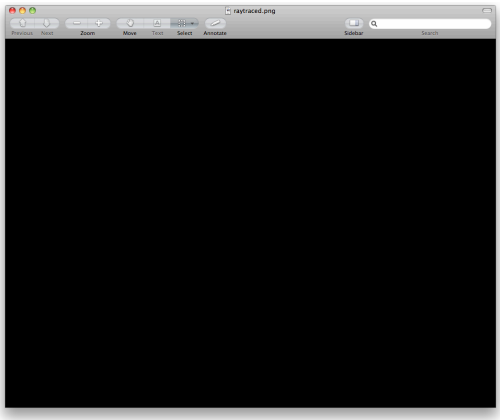
Sonstige Hinweise

Kugeln



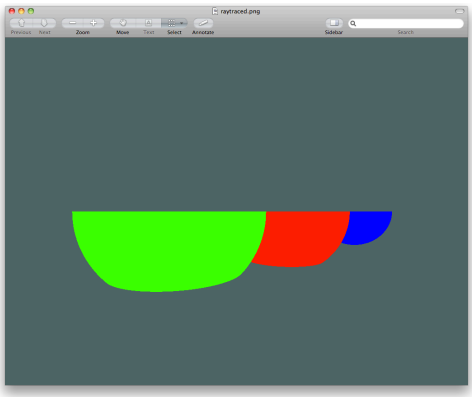
- Entwerfen Sie eine Szene mit einer Kugel
- Achten Sie darauf, dass die Kugel von der Kamera gesehen wird
- Wenn das klappt, fügen Sie weitere Kugeln hinzu

Schwarzes Bild...



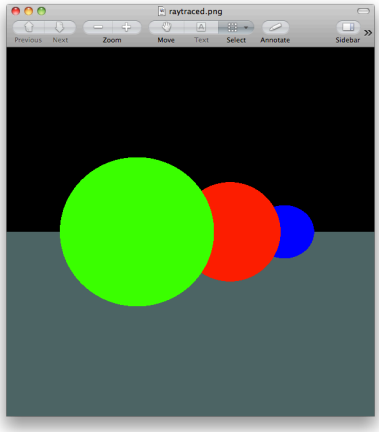
- Sind die Objekte vor dem Auge?
- Ist die Bildebene vor dem Auge? (Achtung –z_alpha nehmen!)
- Funktioniert die Schnittpunktberechnung?

Eine Ebene

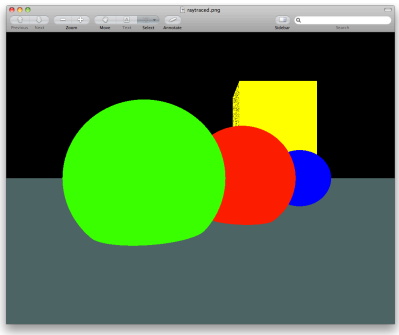


- Versuch, eine Ebene unterhalb der Kugeln zu plazieren
- Der untere Teil des Bildes stimmt, der obere nicht
- Was könnte das verursachen...?

Eine Ebene - korrekt



Eine Axis Aligned Box



- Im Prinzip schon ganz gut, aber...
- „Rauschen“ – manche Punkte sind schwarz anstatt gelb
- Was könnte das verursachen...?

Axis Aligned Box – korrekt.

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences