

# **7. Computergraphik Übung: Raytracing**

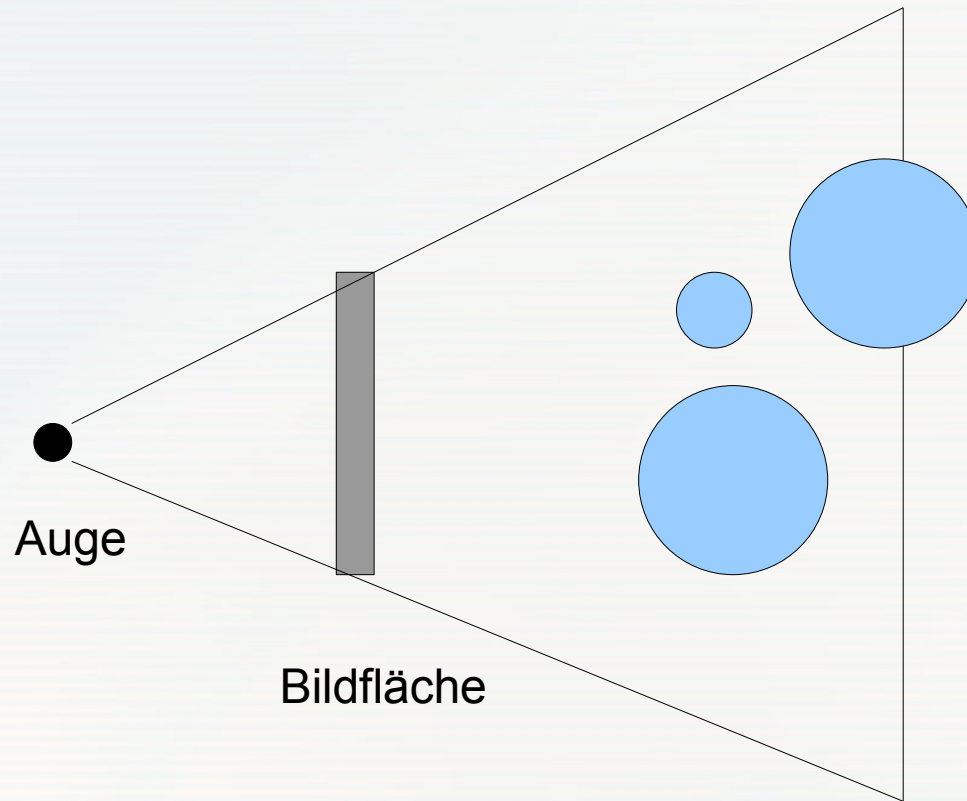
**Heni Ben Amor**

**Arbeitsgruppe Virtuelle Realität und Multimedia**

**TU Bergakademie Freiberg**

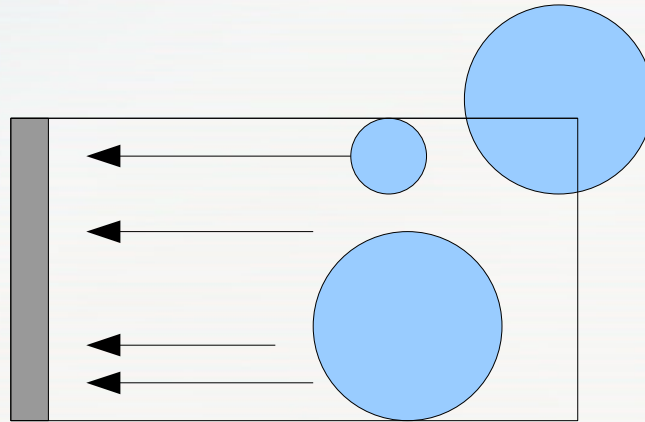
# Projektionen

- **Idee: Abbildung der 3D Szene auf 2D Bildfläche**



# Projektionen

- **Orthographische Projektion:**
  - eine Komponente (Z) wird abgeschnitten
  - Viewvolume wird Bildschirmgröße angepasst



Bildfläche

# Projektionen

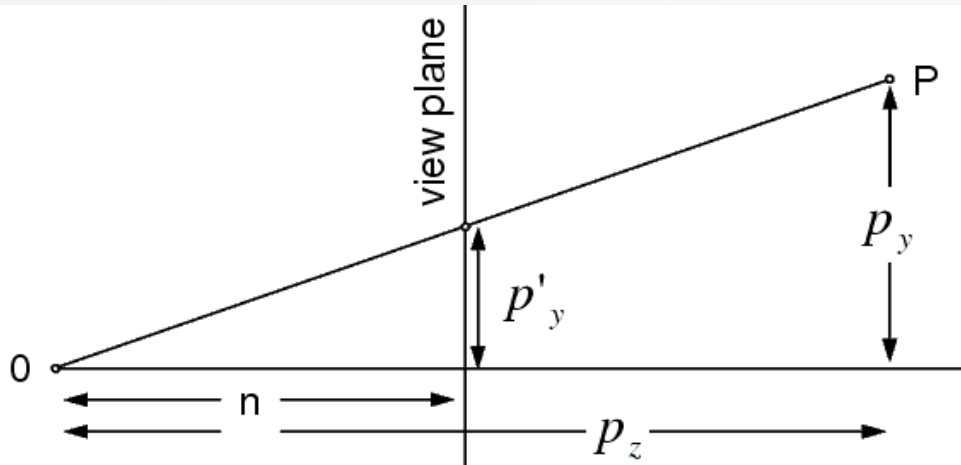
- **Orthographische Projektion:**
  - eine Komponente (Z) wird abgeschnitten
  - Viewvolume wird Bildschirmgröße angepasst
  - OpenGL Bildschirmbereich [-1..1] in x,y,z

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Orthographische Projektionsmatrix

# Projektionen

- **Prespektivische Projektion:**
  - Verzerrung entlang der z-Achse
  - Viewvolume wird Bildschirmgröße angepasst



$$\frac{p'_y}{p_y} = \frac{n}{p_z}$$

$$p'_y = \frac{n}{p_z} \cdot p_y$$

$p_z$  kritischer Punkt

# Projektionen

- **Prespektivische Projektion:**
  - Verzerrung entlang der z-Achse
  - Viewvolume wird Bildschirmgröße angepasst

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & ? & ? \\ 0 & 0 & 1/n & 0 \end{pmatrix}$$

Perspektivische ProjektionsMatrix

# Projektionen

- **Prespektivische Projektion:**
  - Verzerrung entlang der z-Achse
  - Viewvolume wird Bildschirmgröße angepasst

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & 1/n & 0 \end{pmatrix}$$

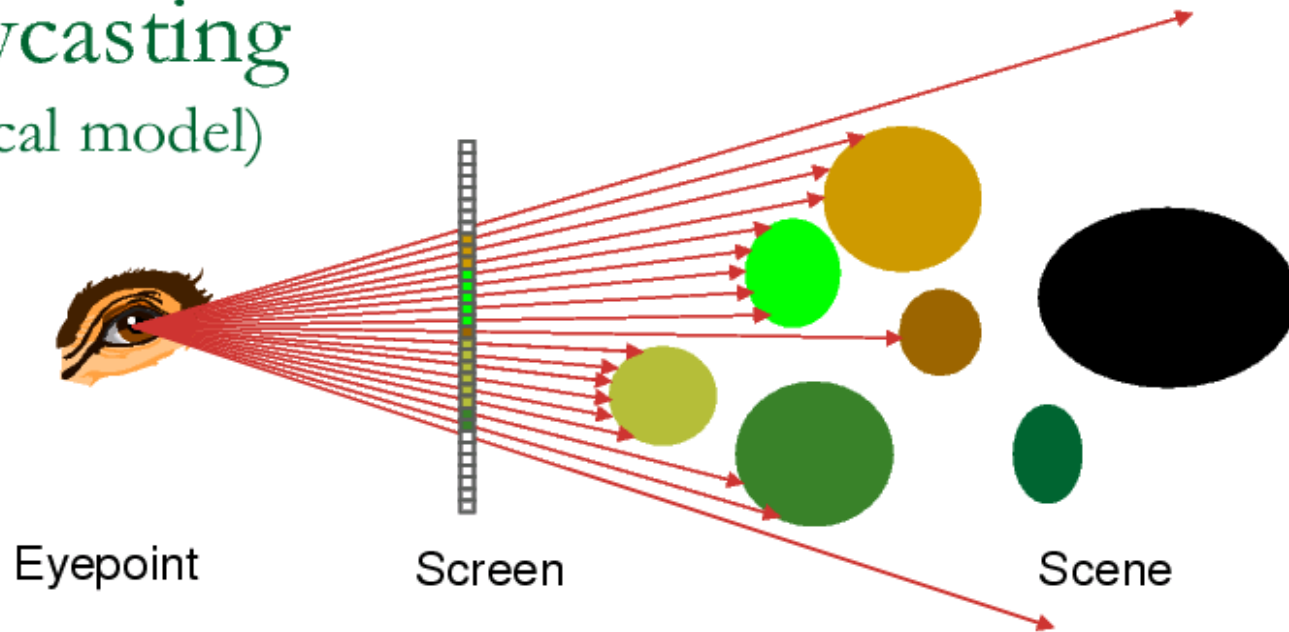
Perspektivische ProjektionsMatrix (OpenGL)

# Shadow Map

1. Kamera in Position und Ausrichtung der Lichtquelle setzen
2. Z-Buffer rendern
3. Z-Buffer in Textur speichern
4. Kamera wieder zurück in Ausgangsposition
5. Szene rendern
6. Bei Rasterisierung:
  - (a) Punkt  $p$  in Koordinatensystem der Lichtquelle transformieren  $\rightarrow p'$
  - (b) if  $p'.z \geq \text{texture}.z$  then beleuchte Punkte
  - (c) else setze Punkt Schwarz

# Raytracing

## Raycasting (a local model)



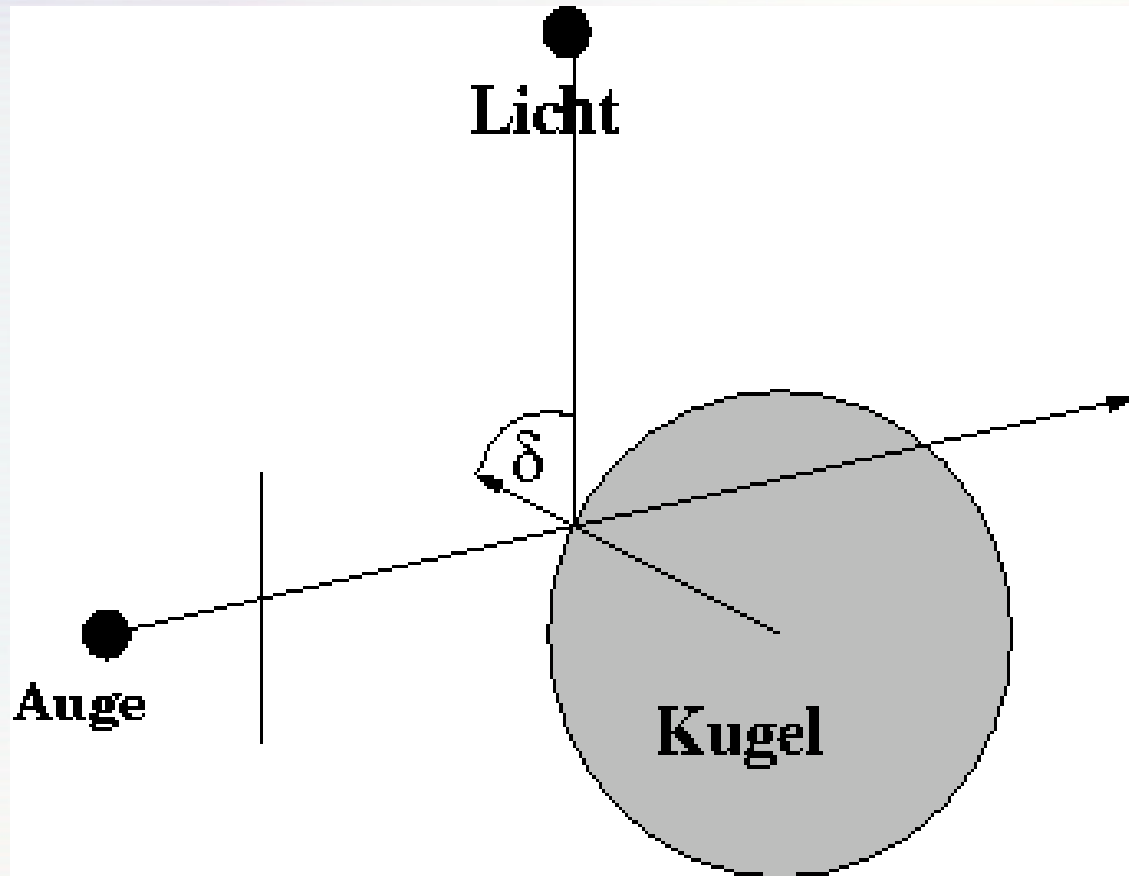
```
for each pixel in the image
  cast a ray from the eye through the pixel
  determine the first intersection with a scene object
  apply the lighting model (e.g. Phong) at the intersection
  paint the pixel in the computed color
```

Note: no clipping, no perspective transformation, no scan conversion

# Simpler Raytracer

- **Komponenten:**
  - **Mathematische Berechnungen:** Vektoren, Strahlen, Geometrien, Schnittpunkt Berechnung
  - **Szenenbeschreibung:** Objekte, Materialien, Lichter
  - **Raytracing/Raycasting Algorithmus**
  - **Ausgabe System**

# Beleuchtung



**Einfache Beleuchtung ergibt sich aus:**

$\cos(\delta) * \text{material.diffus} * \text{material.farbe} * \text{lichtquelle.farbe}$

**aufsummiert über alle lichtquellen**

# Ausgabe

- Ausgabe des Raytracers erfolgt in Bilddatei
- Dateityp: portable pixel map
- ppm ist einfaches unkomprimiertes Bildformat
- Beispiel:

```
P3
# feep.ppm
4 4
15
0 0 0 0 0 0 0 0 0 15 0 15
0 0 0 0 15 7 0 0 0 0 0 0
0 0 0 0 0 0 0 15 7 0 0 0
15 0 15 0 0 0 0 0 0 0 0 0
```

# Ausgabe

- Ausgabe des Raytracers erfolgt in Bilddatei
- Dateityp: portable pixel map
- ppm ist einfaches unkomprimiertes Bildformat
- Beispiel:

```
P3  
# feep.ppm  
4 4  
15  
0 0 0 0 0 0 0 0 0 15 0 15  
0 0 0 0 15 7 0 0 0 0 0 0  
0 0 0 0 0 0 0 15 7 0 0 0  
15 0 15 0 0 0 0 0 0 0 0 0
```

← Magic Number

← Kommentar

← Dimensionen

max. Farbwert



# Ergebnis

