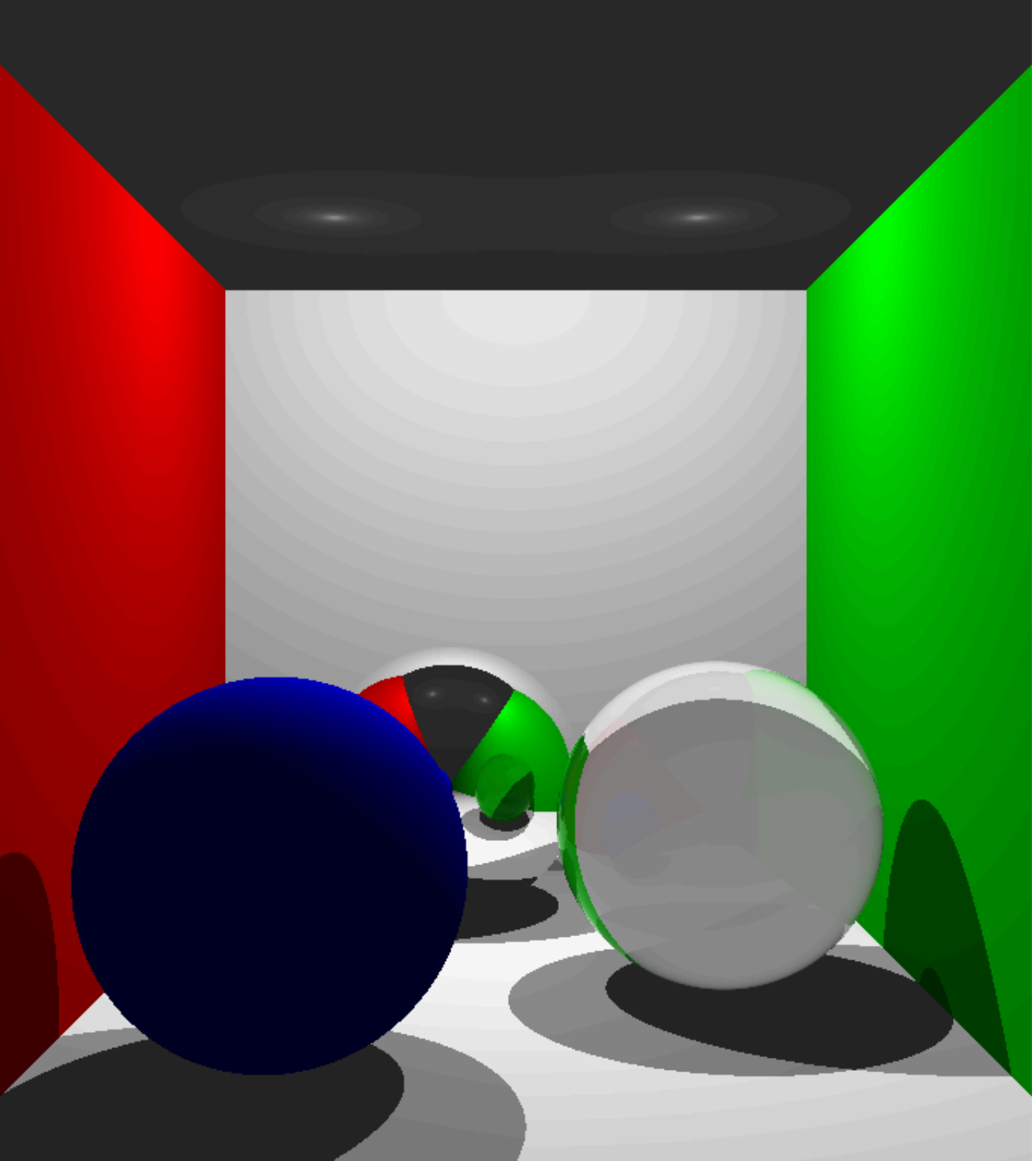
Computergrafik Labor

Vorstellung Raytracer

WS 24/25

Niklas Oesterle (niklas.oesterle@h-ka.de)

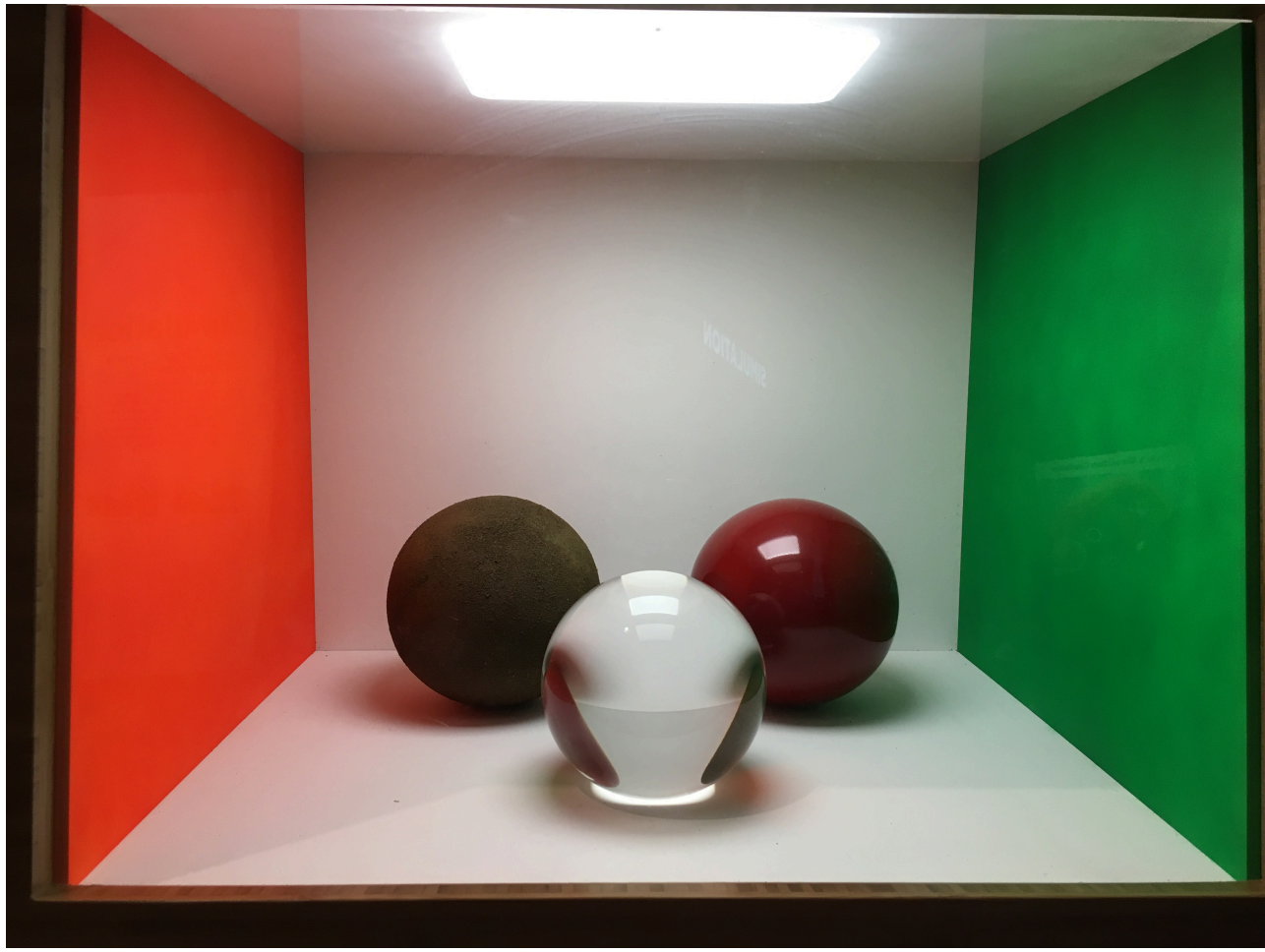


Raytracer (`04_raytracer`)

- Implementierung eines Raytracers
  (Verwendung der bisherigen `math` - und `geometry` - Implementierung)
- "Einfacher" *whitted style* Raytracer
- Lambertian shading

Szenenaufbau (Cornell-Box)

- Verfahren, welches einen Vergleich zwischen einem synthetisierten und einem echten Bild erlaubt
- nach vorne geöffnetes *Zimmer*
- Gegenstände werden in Box platziert
- Konkrete Vorstellung wichtig für "Nachbau"
  - Kamera-Geometrie
  - Objekt-Positionierung



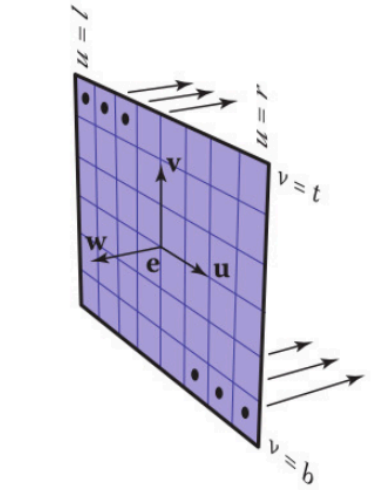
Kameraaufbau

Orthografische Projektion

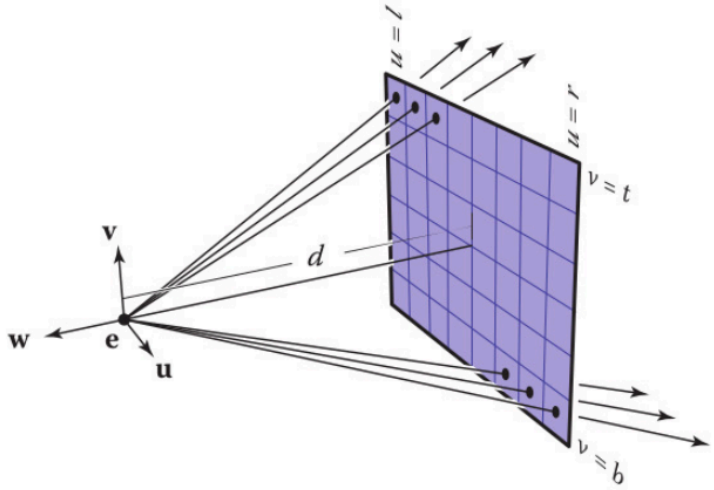
- Bildschirm (l, r, b, t)
- lokales, orthogonales Koordinatensystem $(\vec{u}, \vec{v}, \vec{w})$
- Auflösung (n_x, n_y)

Zusätzlich bei perspektivischer Projektion

- Augpunkt (e)
- Abstand zum Bildschirm (d)

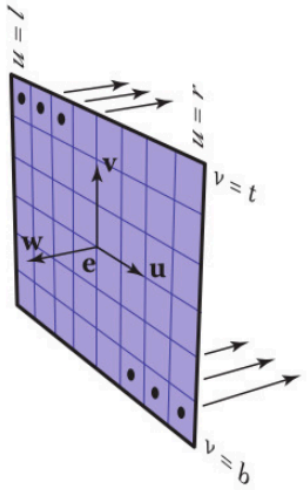


Parallel projection
same direction, different origins

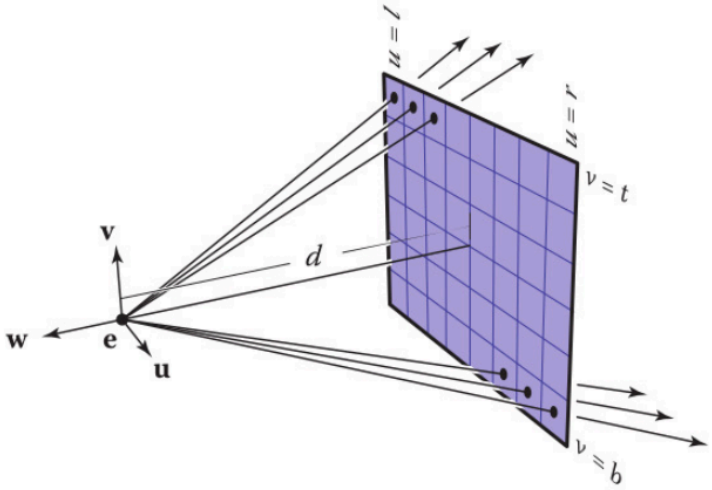


Perspective projection
same origin, different directions

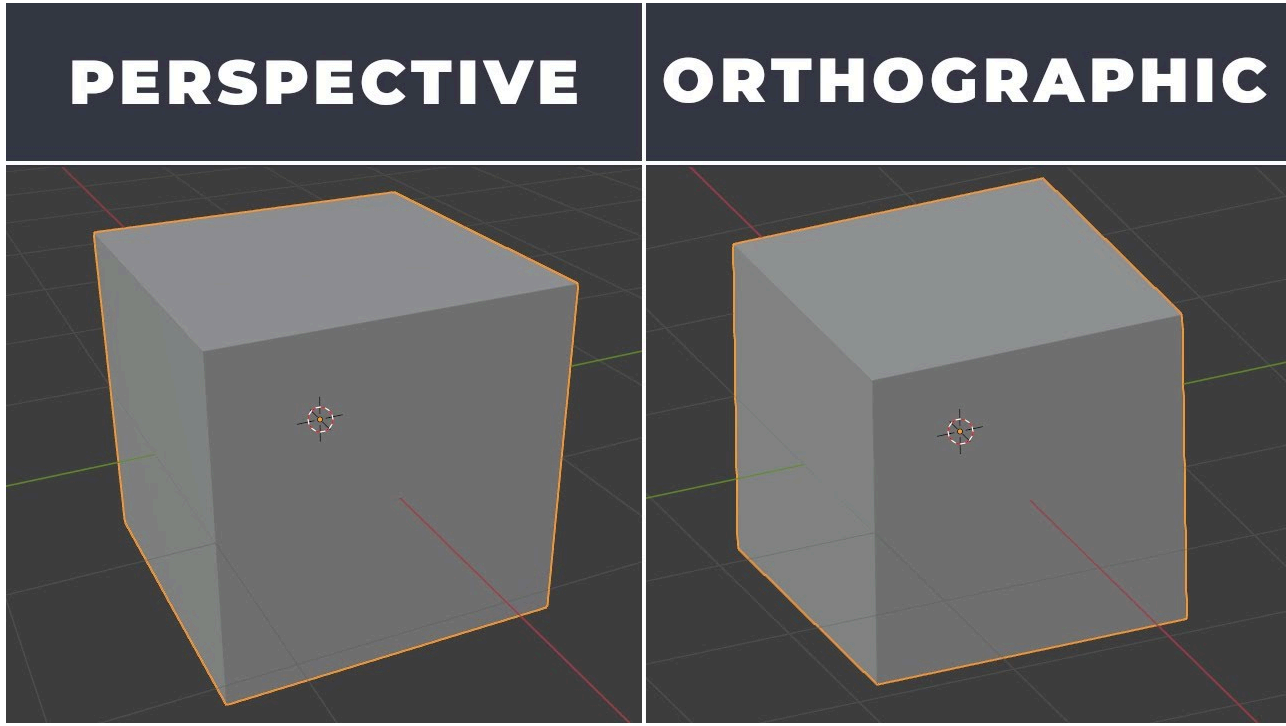
Perspektivische und Orthografische Sicht



Parallel projection
same direction, different origins

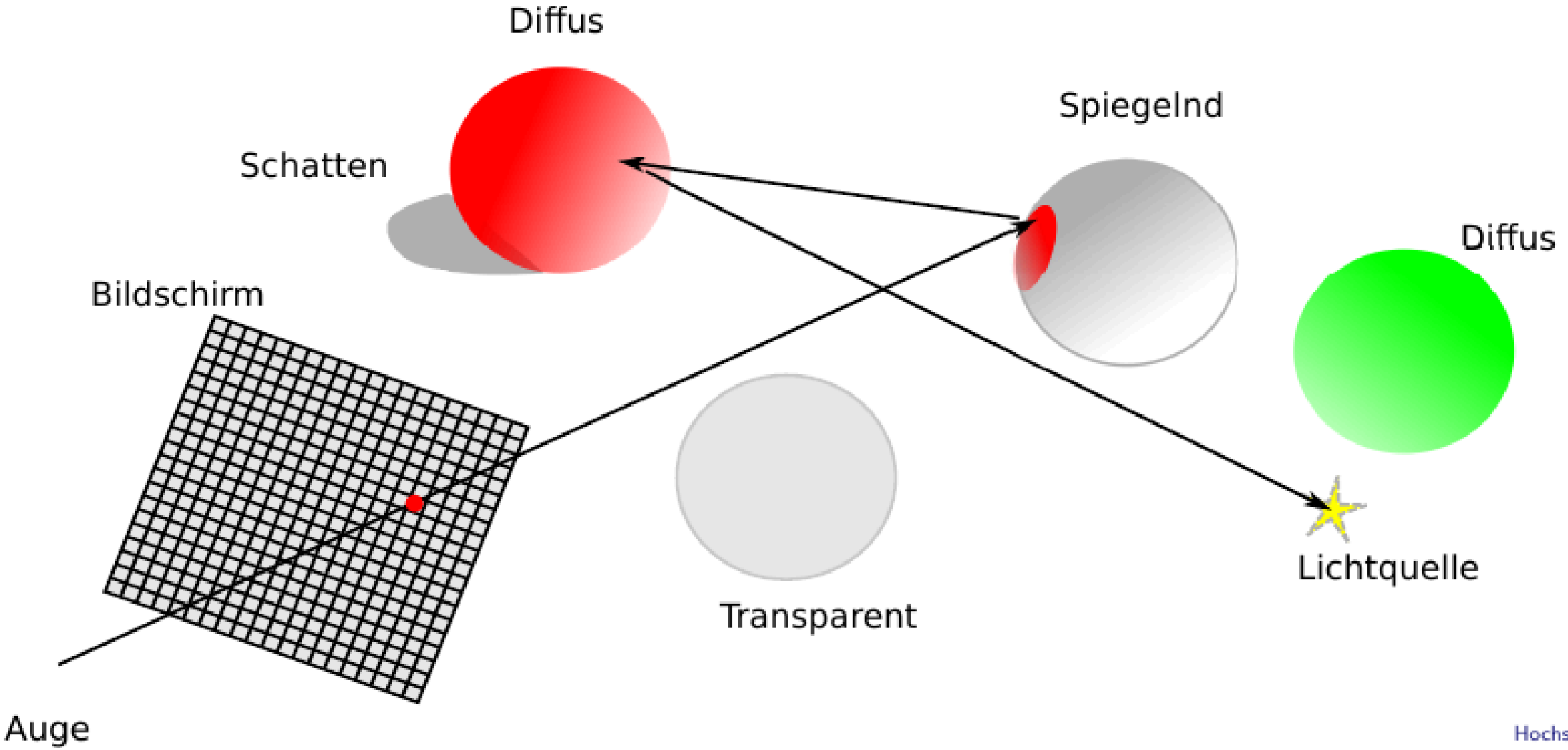


Perspective projection
same origin, different directions



Raytracer - Step-By-Step

1. **Ray Generation**
    i. Bildung eines (Seh-)Strahls (abhängig von Kamera-Geometrie)
2. **Ray Intersection**
    i. Intersection mit anderen Objekten in der Szene prüfen
    ii. Das zur Kamera nächste Objekt herausfinden
3. **Shading**
    i. Pixelfarbe anhand der Intersection berechnen, abhängig von Lichtquellen


Diffus
Schatten
SpiegeInd
Diffus
Bildschirm
Transparent
Lichtquelle
Auge

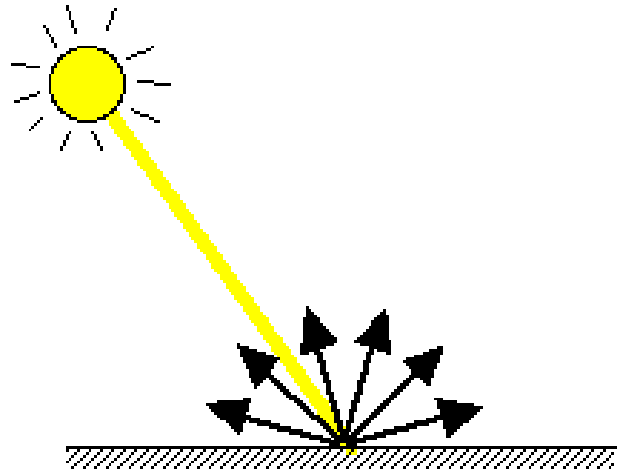
Raytracer - Pseudocode

```
Screen screen(WIDTH, HEIGHT);
for (y = 0; y < HEIGHT; y++)
  for (x = 0; x < WIDTH; x++)
    Ray ray = ...                                    // Ray Generation
    Object object = find_nearest_object( ray )       // Ray Intersection
    Color color = shade(object, ... )                // Shading
    screen.set_pixel(x,y, color)                     // Bildausgabe
```

Lambertian Shading

- Punktförmige Lichtquellen, (x, y, z) genügt
- Für alle nicht-blockierten Lichtquellen Aufhellung berechnen
- Oberflächennormale ($\vec{n}$), Richtung zur Lichtquelle ($\vec{I}$)
- Anteil ambienten Lichts k_a
- Anteil diffusen Lichts k_d

$$L = k_a + k_d \frac{1}{n} \sum_{1}^{n} \cdot \vec{I_i'} \max\{0, \vec{n} \cdot \vec{I_i'}\}$$



Bildausgabe

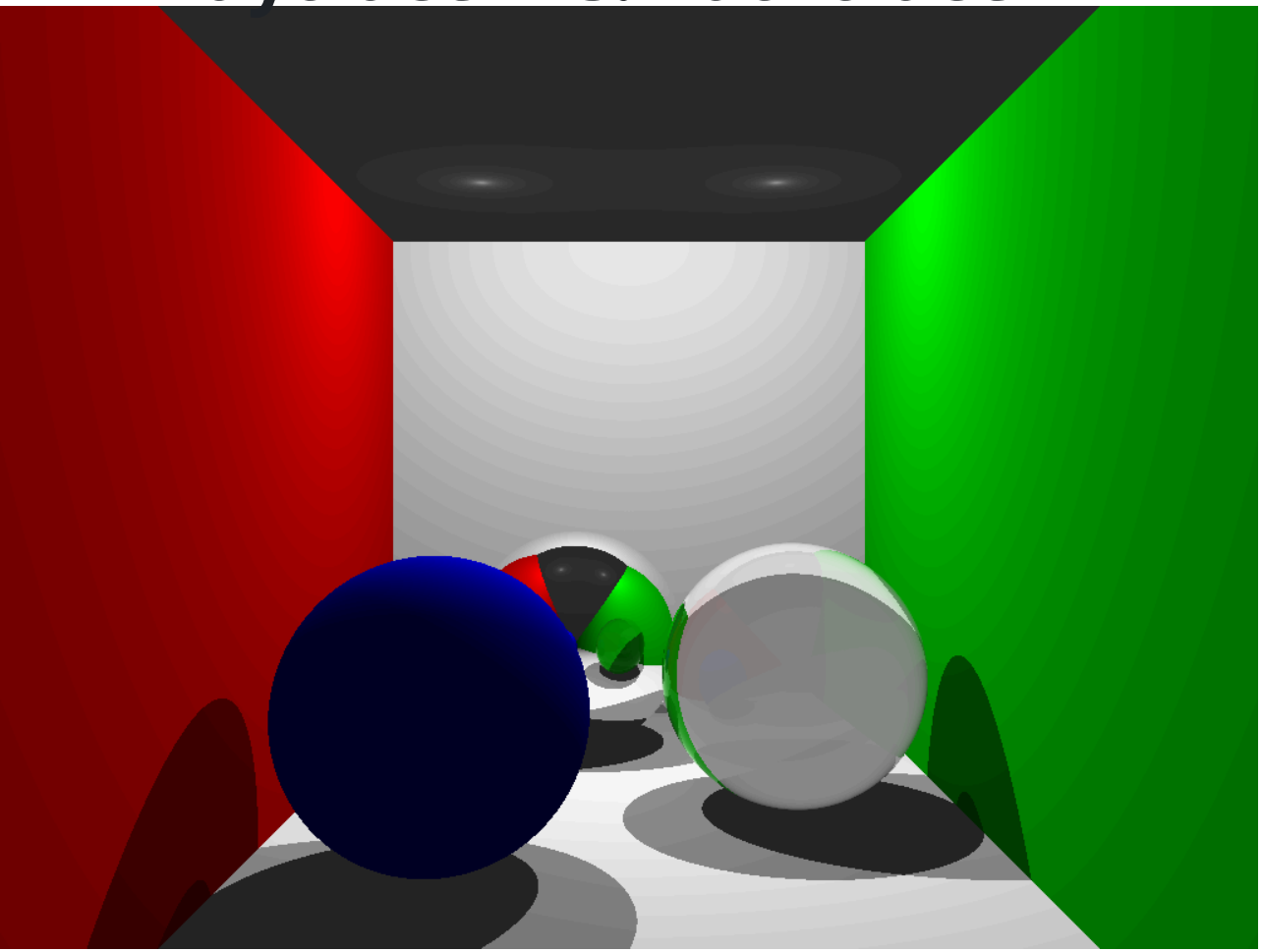
PPM

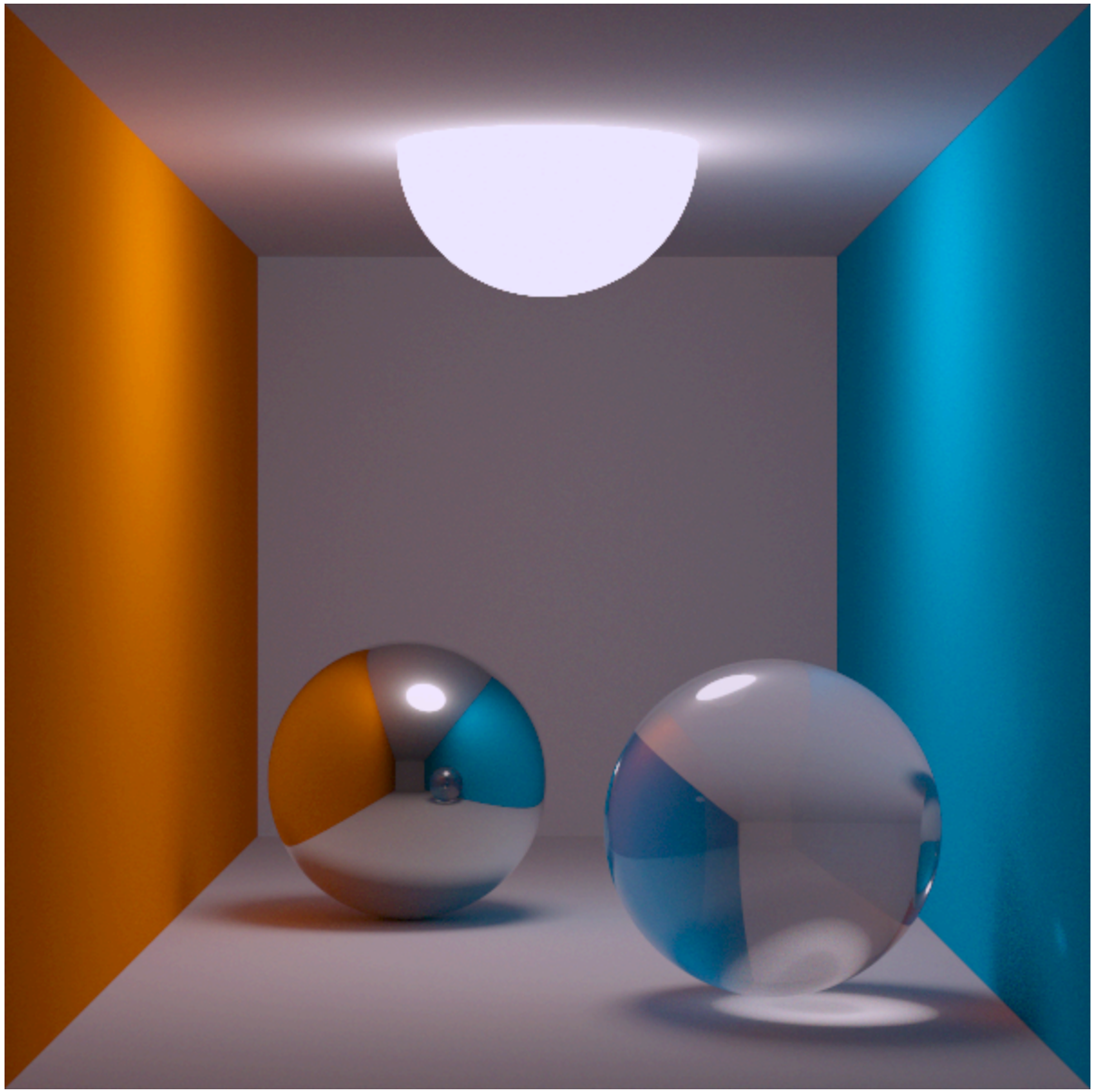
- Pro Pixel: Drei mit Leerzeichen getrennte Farbwerte (siehe Vorlesung)
- Ausgabe in ppm-Datei
- Clion kann ppm-Dateien nicht interpretieren

SDL2

- Bereits in `sdl2renderer.cc` verwendet
  $\rightarrow$ Fensteraufbau kann übernommen werden
- Farbe lässt sich wie zuvor setzen `SDL_SetRenderDrawColor(renderer,r,g,b,a)`
- Pixel des Fensters lassen sich färben `SDL_RenderDrawPoint(renderer,x,y)`

Raytracer vs. Pathtracer





Tipps

- Pseudocode auf Folien und Hinweise auf dem Übungsblatt beachten

Literatur

- **Raytracing in One Weekend**: Erste Kapitel bis Kameraaufbau (vor allem 2. & 4.)
- **Fundamentals of Computer Graphics** (Raytracing ab S. 69)

Häufige Fehler

- Bei Gleitkommazahlen oder numerischen Ungenauigkeiten der Schnittpunktberechnung, kann Schnittpunkt p hinter der Oberfläche sein
- Schattenakne (Strahl mit $p + \epsilon * \vec{n}$ bilden)
- Kamera-Geometrie muss verstanden werden
    - Positionierung der Objekte im Raum (Sichtbarkeit sicherstellen!)