Vorlesung

# Programmiersprachen

im

# Modul Software I

# *Ray Tracing*

Vorlesung im Sommersemester 2014

Bernd Fröhlich

Professur Systeme der Virtuellen Realität

Fakultät Medien

Bauhaus-Universität Weimar

# Ray Tracing

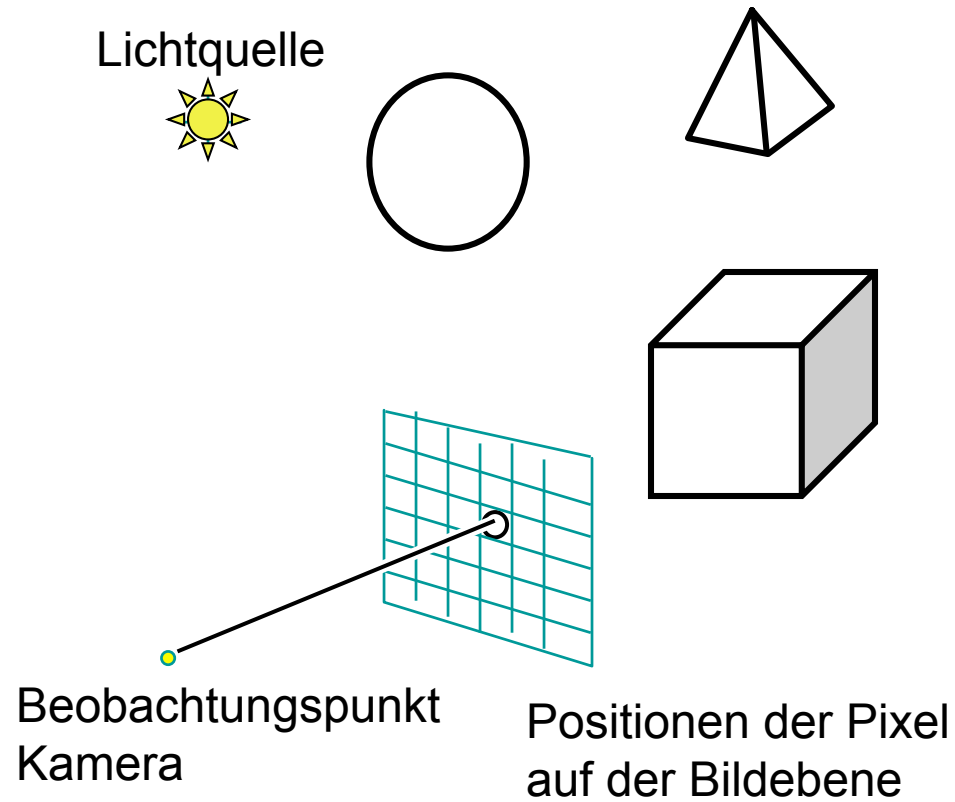
# Ein einfacher Ray Tracer

Unterstützt

- Punktlichtquellen und Schatten
- Spiegelung
- Brechung

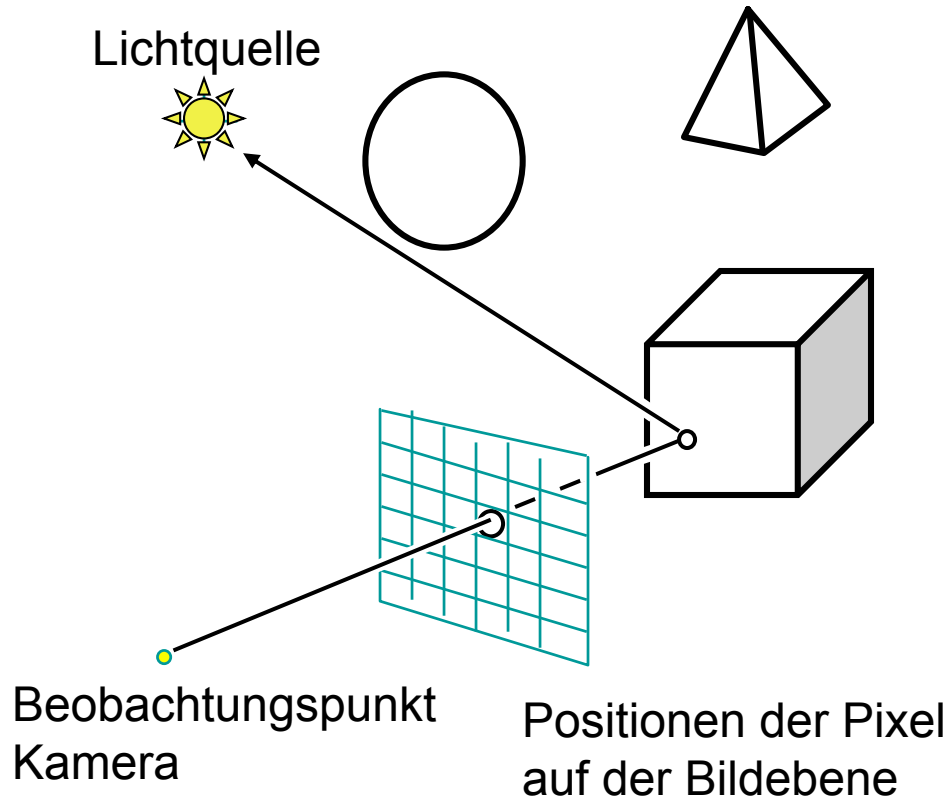
# Verfolgen von Strahlen in die Szene

Die virtuelle Bildebene wird entsprechend der Auflösung des gewünschten Bildes unterteilt. Ein Strahl wird vom Beobachtungspunkt durch jedes Pixel verfolgt.



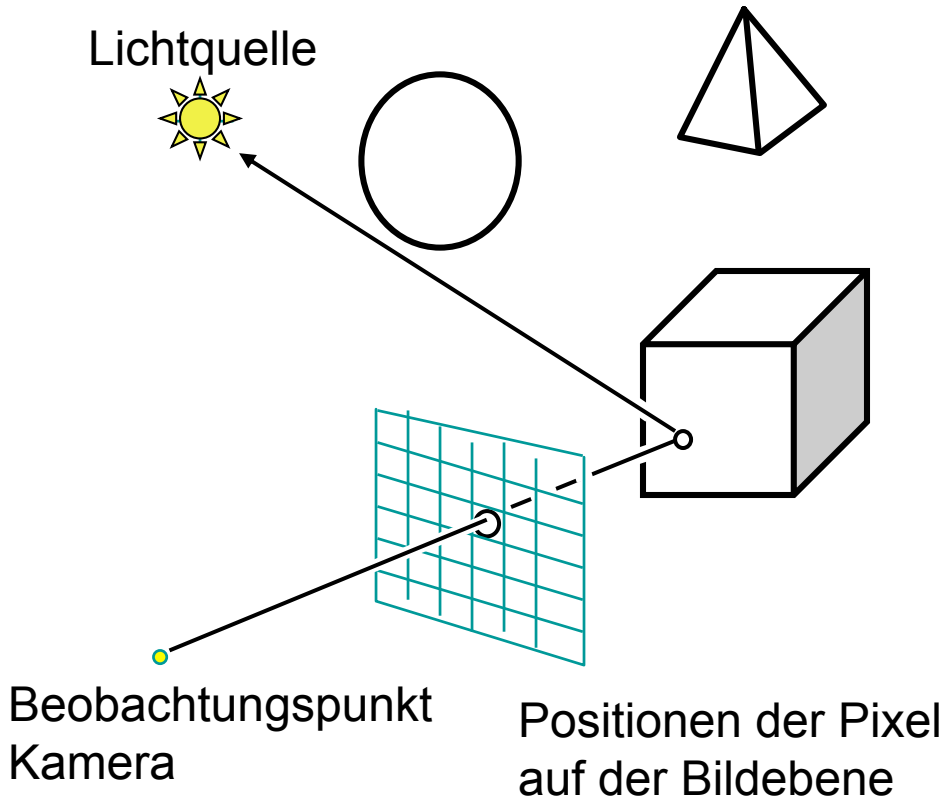
# Verfolgen von Strahlen in die Szene

Es wird der Schnittpunkt des Strahls mit allen Objekten in der Szene ausgerechnet. Der am nächsten zum Beobachter liegende Schnittpunkt bestimmt die sichtbare Oberfläche



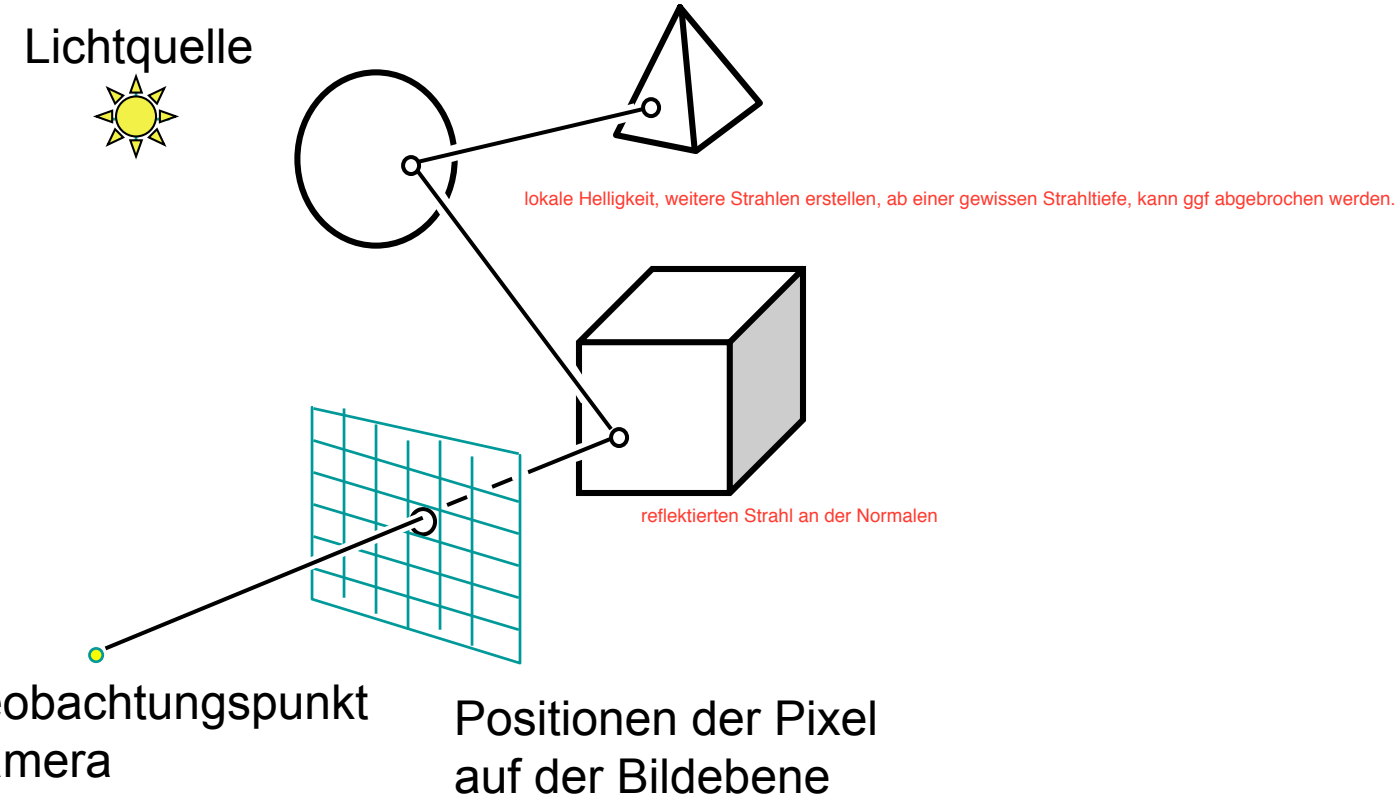
# Schatten

Es wird ein Strahl in Richtung der Lichtquelle geschossen. Liegt ein Objekt zwischen der Lichtquelle und dem betrachteten Oberflächenpunkt, dann liegt der Punkt um Schatten, sonst nicht.



# Spiegelung

Es wird der ankommende Strahl an einer Oberfläche gespiegelt und ein neuer Strahl in Spiegelungsrichtung losgeschickt. Das Verfahren kann rekursiv fortgesetzt werden. Entsprechend kann für Brechung verfahren werden.



# Minimales Ray-Tracing-Programm als Pseudo-Code

**PROZEDURAL – nicht OO!**

```
void raycast()
  for all pixels (x,y)
    image(x,y) = trace( compute_eye_ray(x,y) )

rgbColor trace(ray r)
  for all objects o
    t = compute_intersection(r, o)
    if (t < closest_t)
      closest_t = t
      closest_o = o
  if(closest_o != 0)
    return shade(closest_o, r, closest_t)
  else
    return background_color
```

keinen Schnittpunkt gefunden - also Hintergrund returnen.

```
rgbColor shade(object o, ray r, double t)
  point x = r(t)
  // evaluate (Phong) illumination equation
  return color
```

# Beleuchtungsberechnung

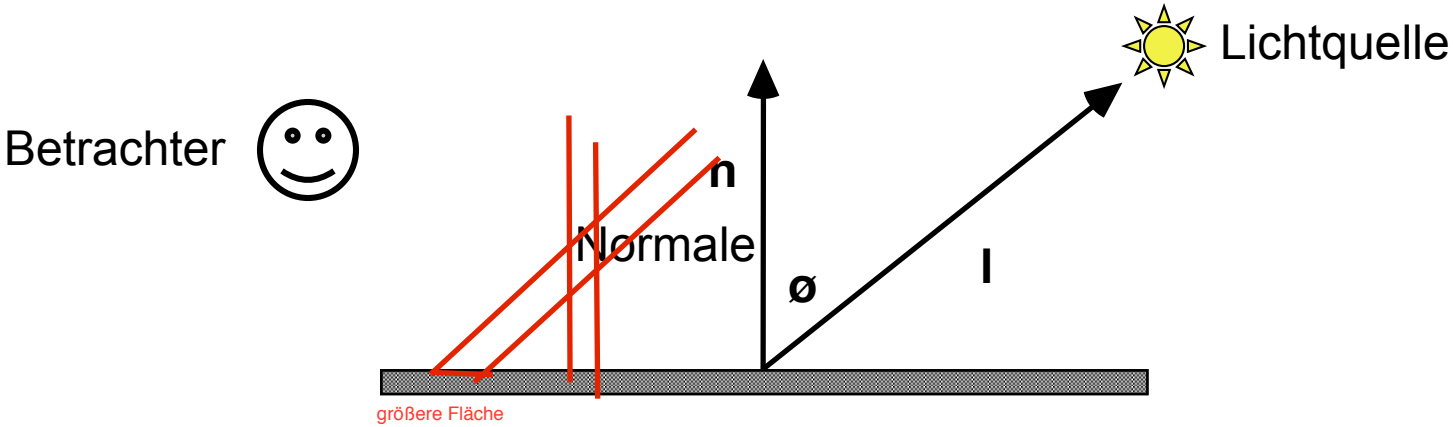
# Diffuse Reflexion nach Lamberts Gesetz

## Lamberts Gesetz

Die Intensität der Lichtreflexion auf einer vollständig diffusen Oberfläche ist proportional zum Kosinus des Winkels zwischen dem Vektor **l** zur Lichtquelle und dem Normalenvektor **n**, der senkrecht auf der Oberfläche steht.

wie hell ist ein Objekt abhängig vom Winkel —> approximation für diffuse Oberflächen

nur so hell, wie für den Winkel in dem es auftrift




größere Fläche

Der Anteil des reflektierten Lichts hängt von der Position der Lichtquelle und von der Position und Normale des Objekts ab, aber nicht von der Position des Betrachters.

# Ein einfaches Beleuchtungsmodell

I = Intensität der Beleuchtung

$I_p$ = Helligkeit einer Punktlichtquelle

$k_d$ float 0..1 = diffuser Reflexionskoeffizient der Oberfläche ($0 \le k_d \le 1$)

Ein einfaches Beleuchtungsmodell: $I = I_p k_d \cos \mathbf{ø} = I_p k_d \cos(\mathbf{n},\mathbf{l})$

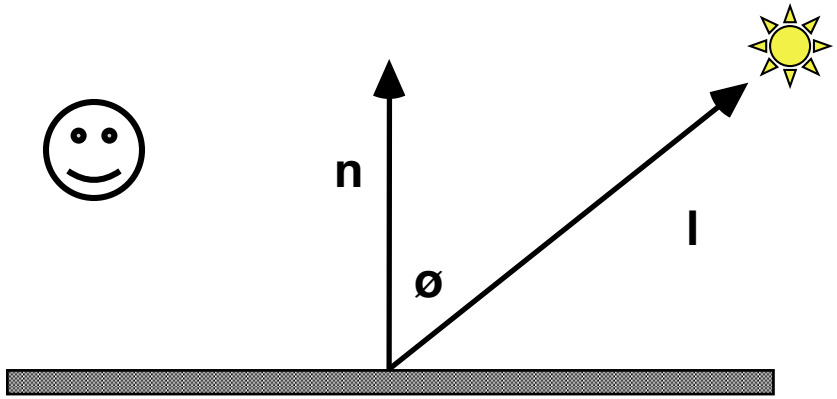
$\cos \mathbf{ø} = (\mathbf{l} \bullet \mathbf{n})/(\|\mathbf{l}\| \, \|\mathbf{n}\|)$

kreuzprodukt, Skalarprodukt

summe aus dem Elementweisen Produkt —> Normierte größen, ergibt den Cosinus des Winkels

Falls **n** und **l** vorher normiert werden

$I = I_p k_d (\mathbf{l} \bullet \mathbf{n})$



# Ambiente Beleuchtung

$I_a$ = Intensität des ambienten Lichts

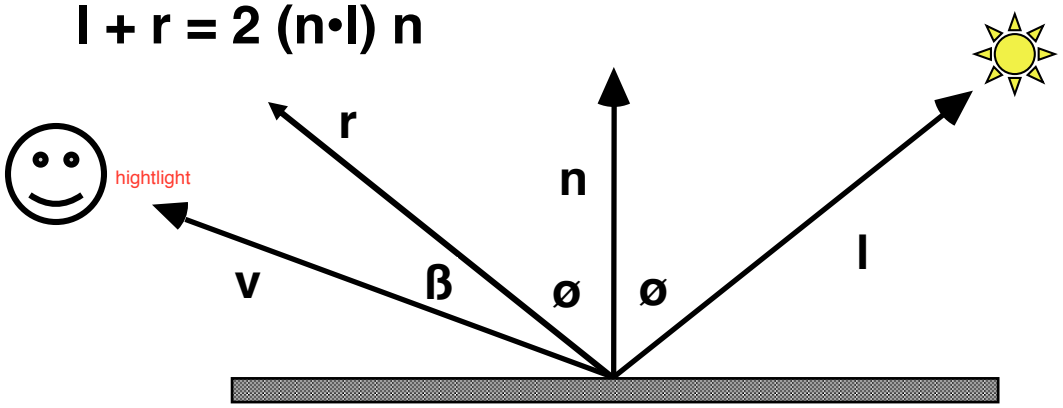
$k_a$ = ambienter Reflexionskoeffizient – Anteil des ambienten Lichts, der von der Oberfläche reflektiert wird

Erweitertes Beleuchtungsmodell

$$I = I_a k_a + I_p k_d (\mathbf{l} \cdot \mathbf{n})$$

Ambiente Beleuchtung fasst Lichtreflexionen anderer Objekte zusammen, die in Richtung der betrachteten Oberfläche reflektiert werden. Die genaue Berechnung dieser Beleuchtung ist aufwändig und wird deshalb oft durch einen konstanten Term approximiert.

# Spiegelnde Reflexion




bekannt

**v** = Vektor zum Betrachter

**n** = Normalenvektor

**I** = Vektor zur Lichtquelle

**r** = reflektierter Lichtvektor

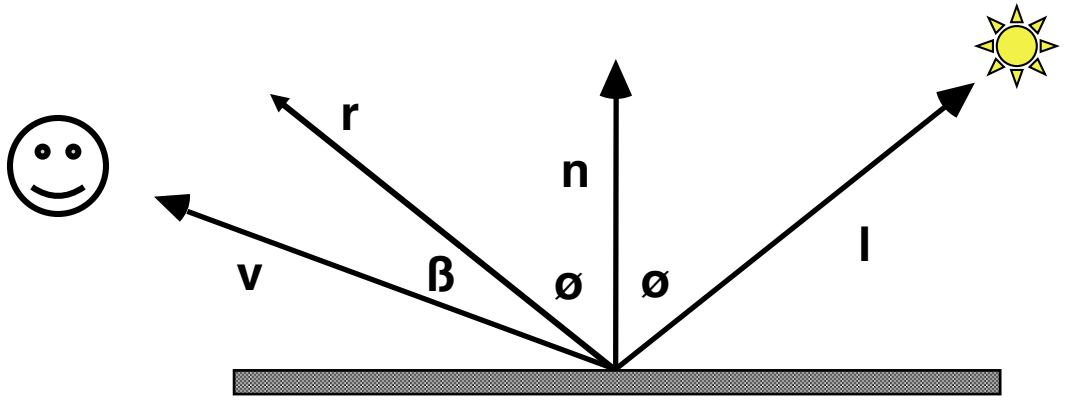
Die Lichtreflexion auf einer spiegelnden Oberfläche hat die Farbe der Lichtquelle. Spiegelnde Reflexion ist abhängig vom Beobachtungspunkt. Maximale spiegelnde Reflexion bei **r** = **v**. Phongs Modell für die Modellierung nicht perfekt spiegelnder Oberflächen:

- $I = I_p\, k_s \cos^m ß$
- $\cos^m ß = (\mathbf{r} \bullet \mathbf{v})^m$ (für **r** und **v** normiert)

m —> grlße des hightlights variabar halten

- $k_s$ Reflexionskoeffizient für die Spiegelung
- Der Exponent m legt die Spiegelungseigenschaften fest

# Vollständiges Beleuchtungsmodell
# Ambiente + Diffuse + Spiegelnde Reflexion




**v** = Vektor zum Betrachter

**n** = Normalenvektor

**l** = Vektor zur Lichtquelle

**r** = reflektierter Lichtvektor

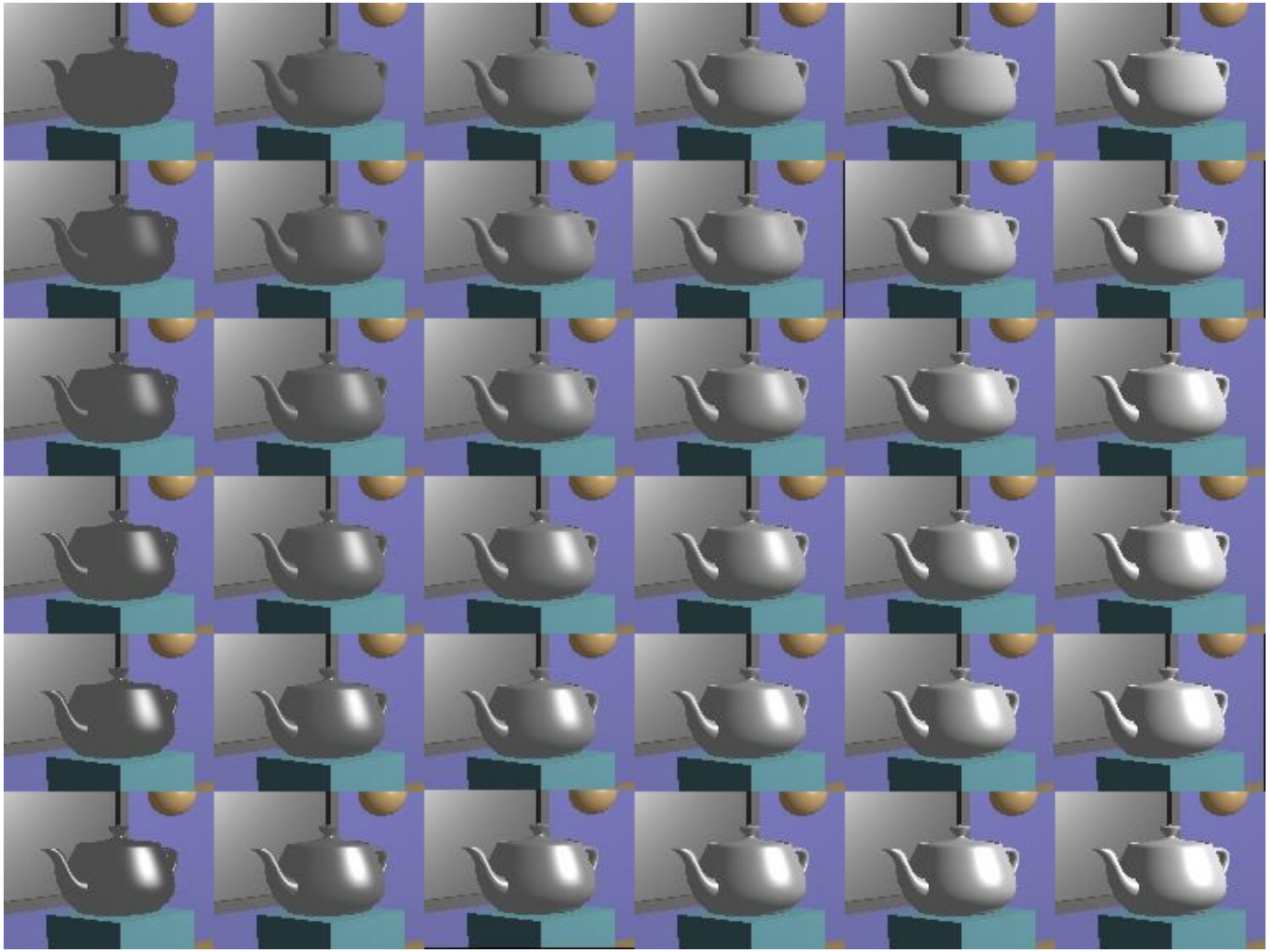
lambda tripel aus r,g,b

$$I(\lambda) = I_a(\lambda)\, k_a(\lambda) + I_p(\lambda)(k_d(\lambda)\cos ø + k_s(\lambda)\cos^m ß)$$

$$I(\lambda) = I_a(\lambda)\, k_a(\lambda) + I_p(\lambda)(\, k_d(\lambda)\,(\mathbf{l}\bullet\mathbf{n}) + k_s(\lambda)\,(\mathbf{r}\bullet\mathbf{v})^m\,)$$

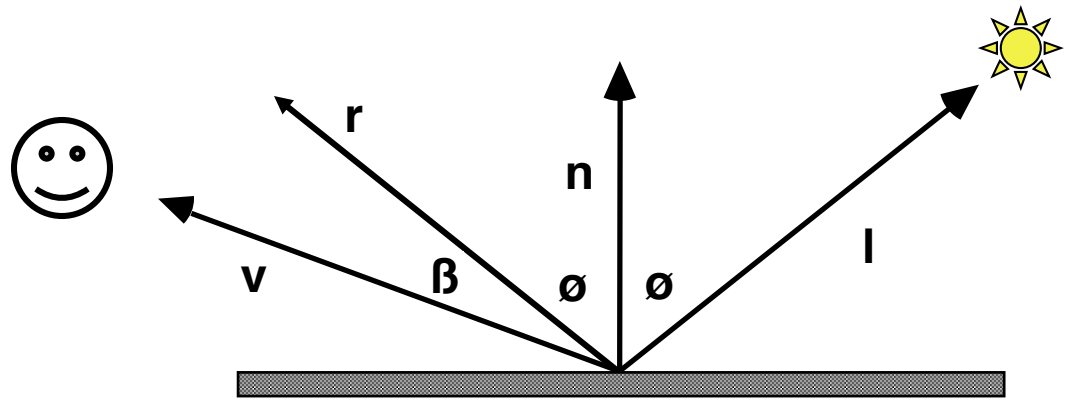
Meist wird die Wellenlänge durch die drei Grundfarben rot, grün und blau repräsentiert – eine starke Vereinfachung. Man erhält damit drei Gleichungen zur Berechnung der Beleuchtung, jeweils eine für jede Grundfarbe. Ausdruck für rot grün und blau

Phongs Beleuchtungsmodell: 0.0 <= $k_s$ <=1.0, 0.0 <= $k_d$ <= 1.0
($k_a$ = 0.7, m = 10.0)



Phongs Beleuchtungsmodell: 0.0 <= $k_s$ <=1.0, 10.0 <= m <= 810.0
($k_a$ = 0.7, $k_d$ = 1.0)

# Vollständiges Beleuchtungsmodell
# Ambient + Diffus + SpiegelInd + Schatten




**v** = Vektor zum Betrachter

**n** = Normalenvektor

**l** = Vektor zur Lichtquelle

**r** = reflektierter Lichtvektor

$$I(\lambda) = I_a(\lambda)\, k_a(\lambda) + I_p(\lambda)\, \delta(\mathbf{l})(k_d(\lambda) \cos \varnothing + k_s(\lambda) \cos^m \beta)$$

$\delta(\mathbf{l})$ = 1 falls Lichtquelle sichtbar – 0 sonst

schattenfunktion - falls sich ein Objekt dazwischen befindet (abnegert)

Für mehrere Lichtquellen

$$I(\lambda) = I_a(\lambda)\, k_a(\lambda) + \sum_i I_{i,p}(\lambda)\, \delta(\mathbf{l}_i)\, (k_d(\lambda) \cos \varnothing_i + k_s(\lambda) \cos^m \beta_i)$$



besser als Phong wären Glossy-Reflexions

# Berechnung von Strahl-Objekt-Schnittpunkten

Ein grosser Teil der Rechenzeit beim Ray Tracing geht in die Berechnung der Schnittpunkte von Strahlen mit Objekten

Zwei einfache Beispiele

- Kugel
- Achsenparalleler Quader

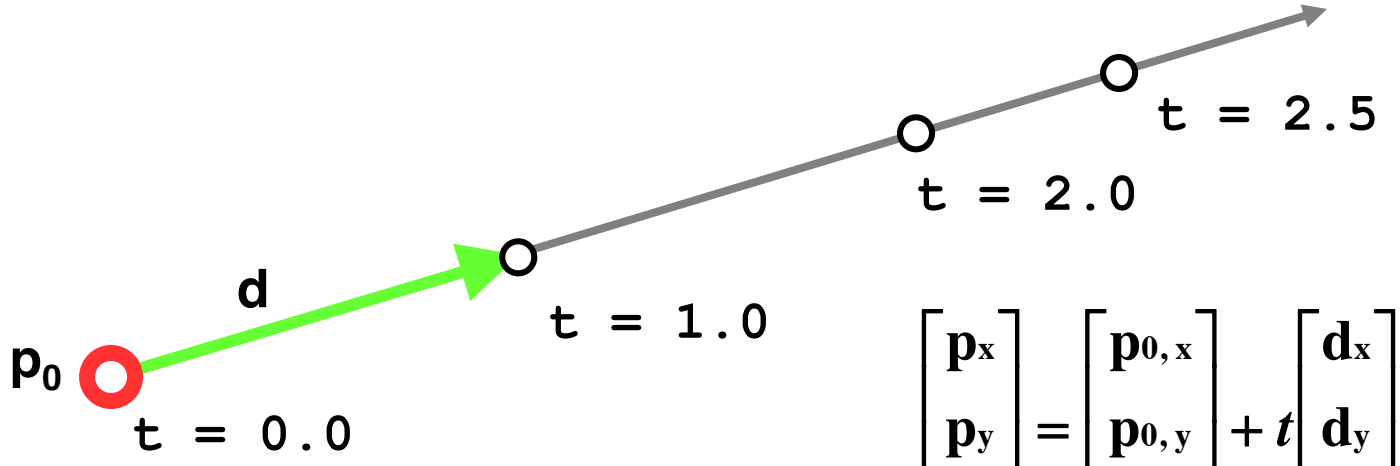
Schnittpunktberechnungen für andere Objekte in der Literatur

# Repräsentation eines Strahls

= Punkt und Richtungsvektor

## Parametrische Repräsentation

- $\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$
- $\mathbf{p}_0$ : der Ursprung des Strahls
- $\mathbf{d}$ : die Richtung des Strahls – sollte Länge 1 haben
- t : Strahlparameter




$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} p_{0,x} \\ p_{0,y} \\ p_{0,z} \end{bmatrix} + t \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$$

# Schnitt Strahl – Kugel

Die Gleichung für eine Kugel mit der Mitte im Ursprung und Radius r

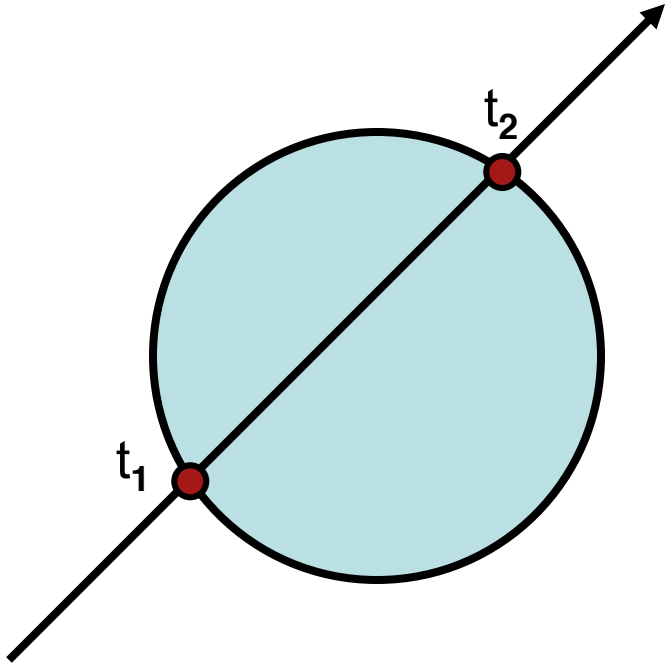
- $x^2 + y^2 + z^2 - r^2 = \mathbf{x} \bullet \mathbf{x} - r^2 = 0$

Einsetzen der Strahlgleichung $\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$

- $(\mathbf{p}_0 + t\,\mathbf{d})(\mathbf{p}_0 + t\,\mathbf{d}) - r^2 = 0$
- $t^2\,\mathbf{d} \bullet \mathbf{d} + 2t\,\mathbf{p}_0 \bullet \mathbf{d} + \mathbf{p}_0 \bullet \mathbf{p}_0 - r^2 = 0$

Quadratische Gleichung in t

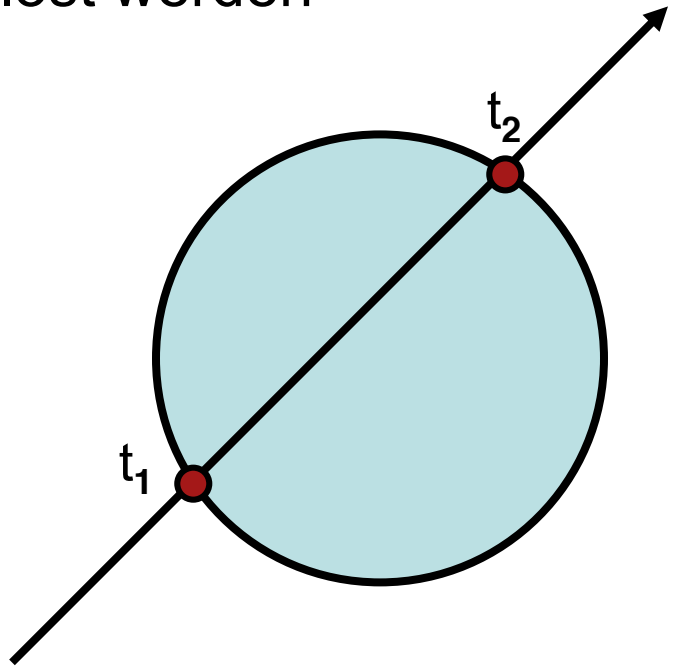
- $at^2 + bt + c = 0$ mit
  - $a = \mathbf{d} \bullet \mathbf{d} = 1$ ($\mathbf{d}$ hat Länge 1)
  - $b = 2\,\mathbf{p}_0 \bullet \mathbf{d}$
  - $c = \mathbf{p}_0 \bullet \mathbf{p}_0 - r^2$



# Schnitt Strahl – Kugel

Quadratische Gleichung kann direkt gelöst werden

- $at^2 + bt + c = 0$ mit
  - $a = \mathbf{d} \bullet \mathbf{d} = 1$ (**d** hat Länge 1)
  - $b = 2\ \mathbf{p_0} \bullet \mathbf{d}$
  - $c = \mathbf{p_0} \bullet \mathbf{p_0} - r^2$




Ergibt 2 potentielle Lösungen

- $t_1 = 0.5\ (-b - \mathrm{sqrt}(b^2 - 4c))$
- $t_2 = 0.5\ (-b + \mathrm{sqrt}(b^2 - 4c))$
- Keine Lösung: Strahl verfehlt Kugel
- Eine Lösung: Strahl ist tangent
- Zwei Lösungen: die kleinste positive ist der gesuchte Schnittpunkt

# Schnittberechnung Strahl – beliebige Kugel

Gleichung einer Kugel bzw. der Kugeloberfläche mit Mittelpunkt **s** und Radius r:

$$(x - s_x)^2 + (y - s_y)^2 + (z - s_z)^2 = r^2$$

Einsetzen der Strahlgleichungen an der Stelle x, y und z

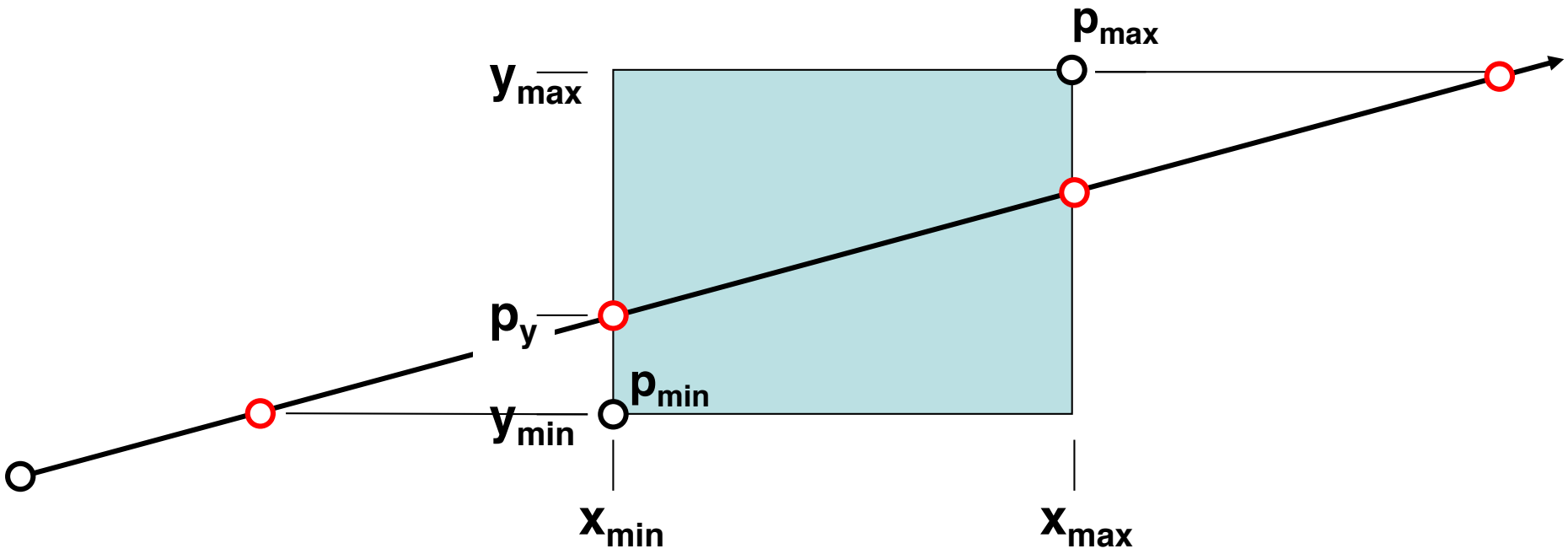
$$(p_x(t) - s_x)^2 + (p_y(t) - s_y)^2 + (p_z(t) - s_z)^2 = r^2$$

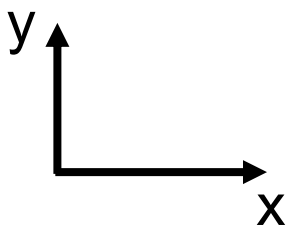
$$((p_{0,x} + t\, d_x) - s_x)^2 + ((p_{0,y} + t\, d_y) - s_y)^2 + ((p_{0,z} + t\, d_z) - s_z)^2 = r^2$$

Ergibt quadratische Gleichung in t

- $at^2 + bt + c = 0$
- Kleinstes positives t ist erster Schnittpunkt, falls er existiert

# Schnitt Strahl – achsenparalleler Quader



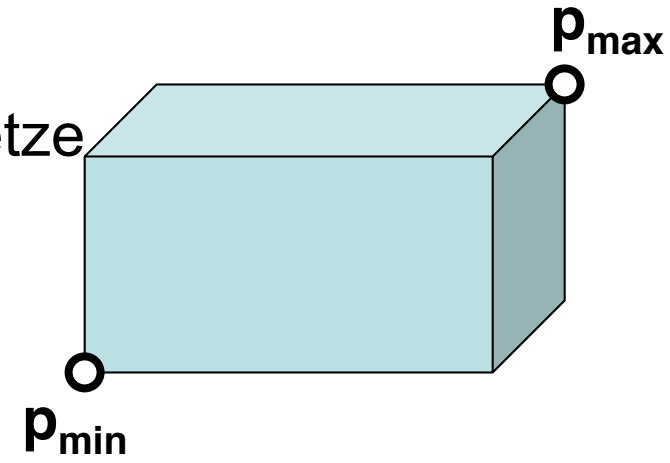

| | |
|---|---|
| Linke Ebene | $x = x_{min}$ |
| Strahl | $p_x = p_{0x} + t\, d_x$ |
| Gleichsetzen: | $x_{min} = p_{0x} + t\, d_x$ |
| Umstellen: | $t = (x_{min}-p_{0x}) / d_x$ |
| Y-Koord. : | $p_y = p_{0y} + t\, d_y$ |

# Schnitt Strahl – achsenparalleler Quader

Quader festgelegt durch Mittelpunkt und Kantelänge in x-, y- und z-Richtung oder durch die Punkte mit den minimalen und maximalen x-, y- und z-Koordinaten. $p_{min} = (x_{min}\, y_{min}\, z_{min})$

Schneide Strahl mit der Ebene $x = x_{min}$

$p_{0,x} + t\, d_x = x_{min}$ Löse nach t auf und setze Ergebnis in die Strahlgleichung ein.




Überprüfe ob die y- und z-Koordinaten des Schnittpunkts zwischen $y_{min}$ und $y_{max}$ und $z_{min}$ und $z_{max}$ liegen.

Falls ja – liegt ein Schnitt vor.

Teste alle Seitenflächen des Quaders um den am nächsten zum Ursprung liegenden Schnittpunkt zu finden.

Optimierung: Teste nur die Flächen des Quaders, die dem Betrachter

# Transformationen

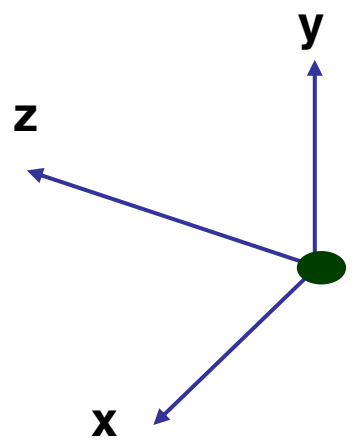
Die grundlegenden linearen Transformationen:

- Translation: **q** = **p** + **v** und **v** ist Translationsvektor punkt verschieben
- Skalierung: **q** = S **p** und S ist eine Skalierungsmatrix
- Rotation: **q** = R **p** und R ist eine Rotationsmatrix

Um alle Transformationen als Matrizen ausdrücken zu können, werden homogene bzw. erweiterte Koordinaten verwendet.

Wir verwenden ein rechtshändiges Koordinatensystem

- Rechte Hand
  - Daumen: x-Achse
  - Zeigefinger: y-Achse
  - Mittelfinger: z-Achse



# Homogene Koordinaten

3D-Punkt $\mathbf{p} = (x,y,z)^T$ wird $\mathbf{p} = (x,y,z,1)^T$

x_4 ist das Tag zur Unterscheidung

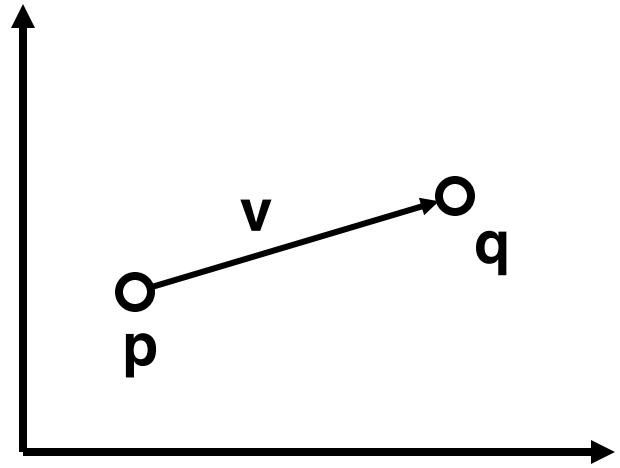
3D-Vektor $\mathbf{v} = (x,y,z)^T$ wird $\mathbf{v} = (x,y,z,0)^T$

- 3D-Vektor ist Differenz zweier Punkte!

Koordinaten voneinander abziehen
zwei Punkte - ergibt dann am Ende 1-1 = 0
einen Vektor

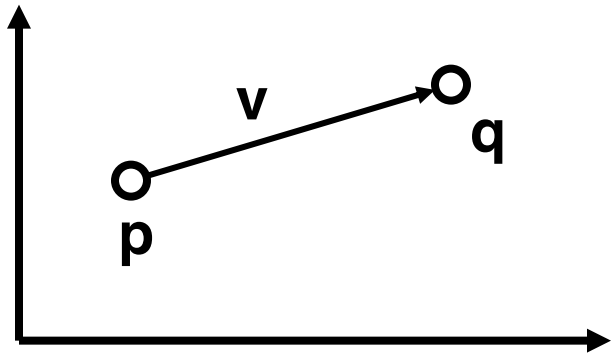
Transformationsmatrizen sind 4x4 Matrizen

- Translation
- Skalierung
- Rotation
- Projektion
- …



# Translation

Verschiebung eines Punktes **p** um einen Vektor **v**: **q** = **p** + **v**

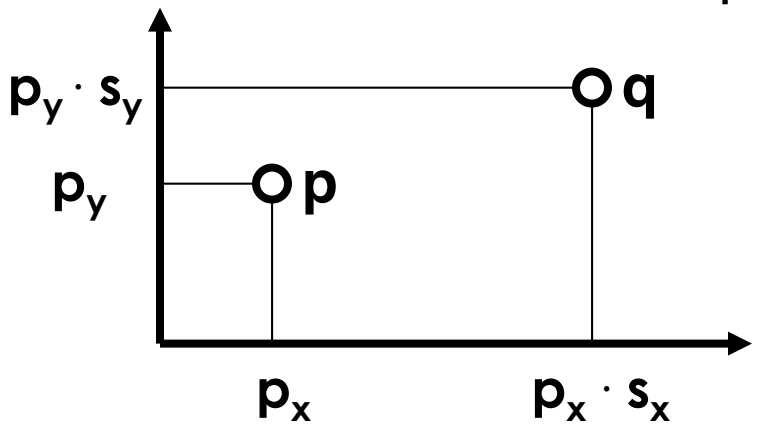



In Matrixschreibweise: **q** = T **p**

$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \\ 1 \end{bmatrix}$$

# Skalierung

Skalierung eines Punktes relativ zum Ursprung




In Matrixschreibweise: **q** = S **p**

$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x s_x \\ p_y s_y \\ p_z s_z \\ 1 \end{bmatrix}$$

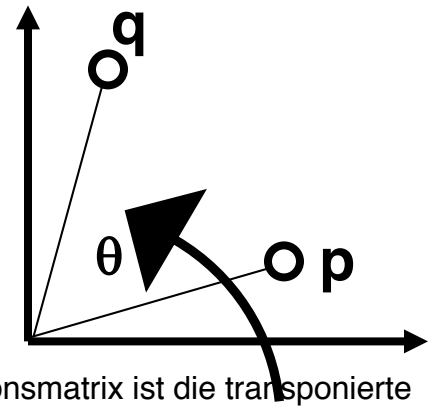
# Rotation

Rotation wird bezüglich einer Achse spezifiziert

Rotation um die x-Achse: $\mathbf{q} = R_x(\theta)\,\mathbf{p}$

$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Punkte auf x bleiben auf x




inverse rotationsmatrix ist die transponierte

y-Achse und z-Achse entsprechend

Positiver Winkel entspricht einer Rotation gegen den Uhrzeigersinn, wenn man von der positiven Achse in Richtung Ursprung schaut.

–

# Zusammengesetzte Transformationen

Mit homogenen Koordinaten lassen sich mehrere aufeinander folgende Transformationen in einer Matrix zusammenfassen.

Beispiel:

- Skalierung S
- Rotation R
- Translation T

**q** = T R S **p** = M **p** mit M = T R S

Hinweis: dies ist oft eine gute Reihenfolge um ein Objekt zu transformieren, falls das Objekt am Anfang zentriert um den Ursprung liegt.

# Inverse Transformationen

$M^{-1} M = I$ (Einheitsmatrix)

Translation: $T^{-1}(a, b, c) = T(-a, -b, -c)$

Scaling: $S^{-1}(s_x, s_y, s_z) = S(1/s_x, 1/s_y, 1/s_z)$

Rotation: $R_x^{-1}(\alpha) = R_x(-\alpha)$

# Anwendung von Transformation

Transformationsmatrix M wird folgendermaßen angewandt

Punkte: $\mathbf{q} = M\,\mathbf{p}$ mit $\mathbf{p}, \mathbf{q}$ in der Form $[x\ y\ z\ 1]^T$

Vektoren: $\mathbf{w} = M\,\mathbf{v}$ mit $\mathbf{w}, \mathbf{v}$ in der Form $[x\ y\ z\ 0]^T$

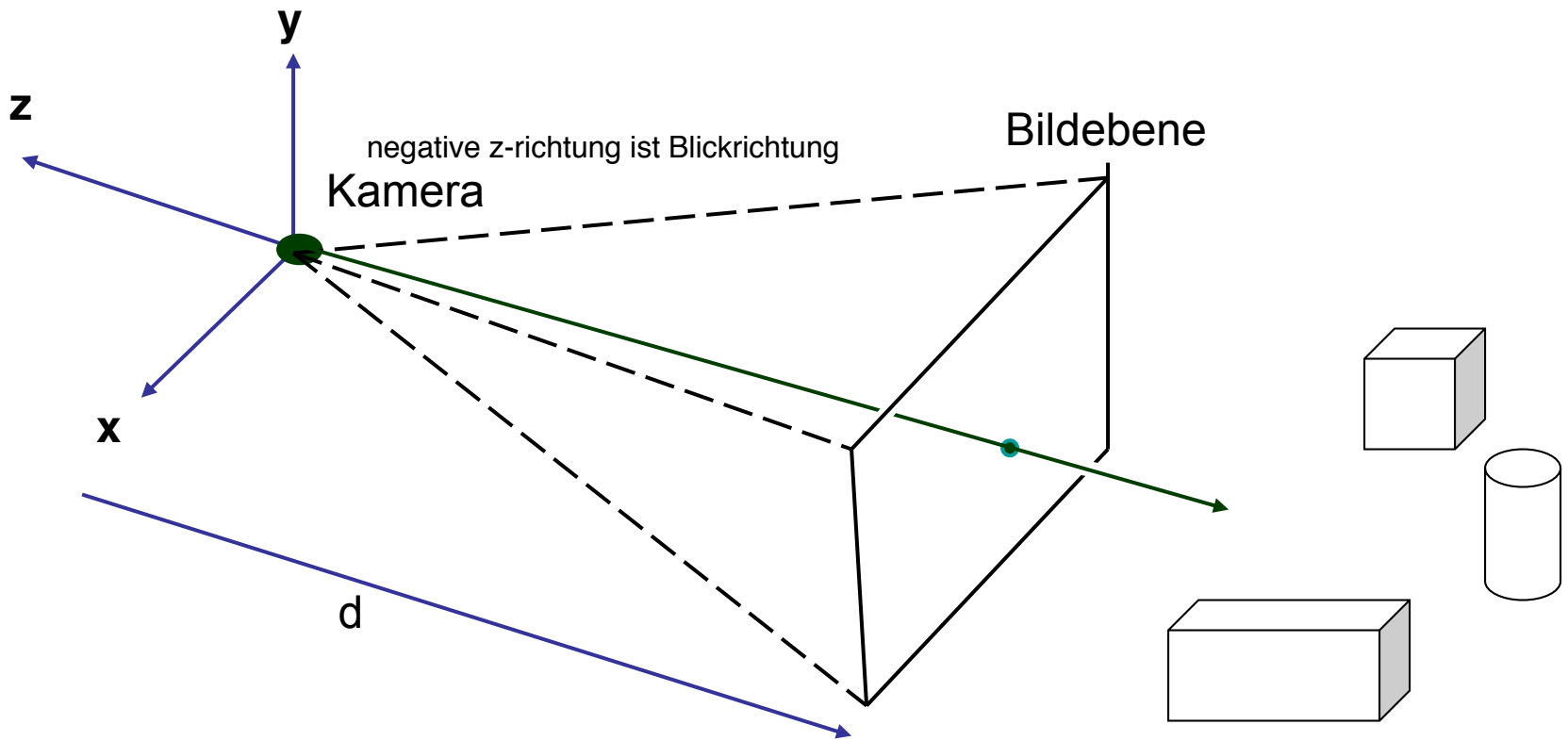
Normalen: $\mathbf{m} = (M^{-1})^T\,\mathbf{n}$ mit $\mathbf{m}, \mathbf{n}$ in der Form $[x\ y\ z\ 0]^T$

Strahlen: $\mathbf{p}(t) = \mathbf{p_0} + t\,\mathbf{d}$

$\mathbf{p_0}$ als Punkt und $\mathbf{d}$ als Vektor transformieren

3D-Objekte: Jeden Punkt getrennt transformieren und ggf. die Normalen entsprechend

# Ein einfaches Kameramodell




Beobachter/Kamera im Ursprung

Blickrichtung ist die negative z-Richtung

Die Bildebene liegt zentriert um die z-Achse

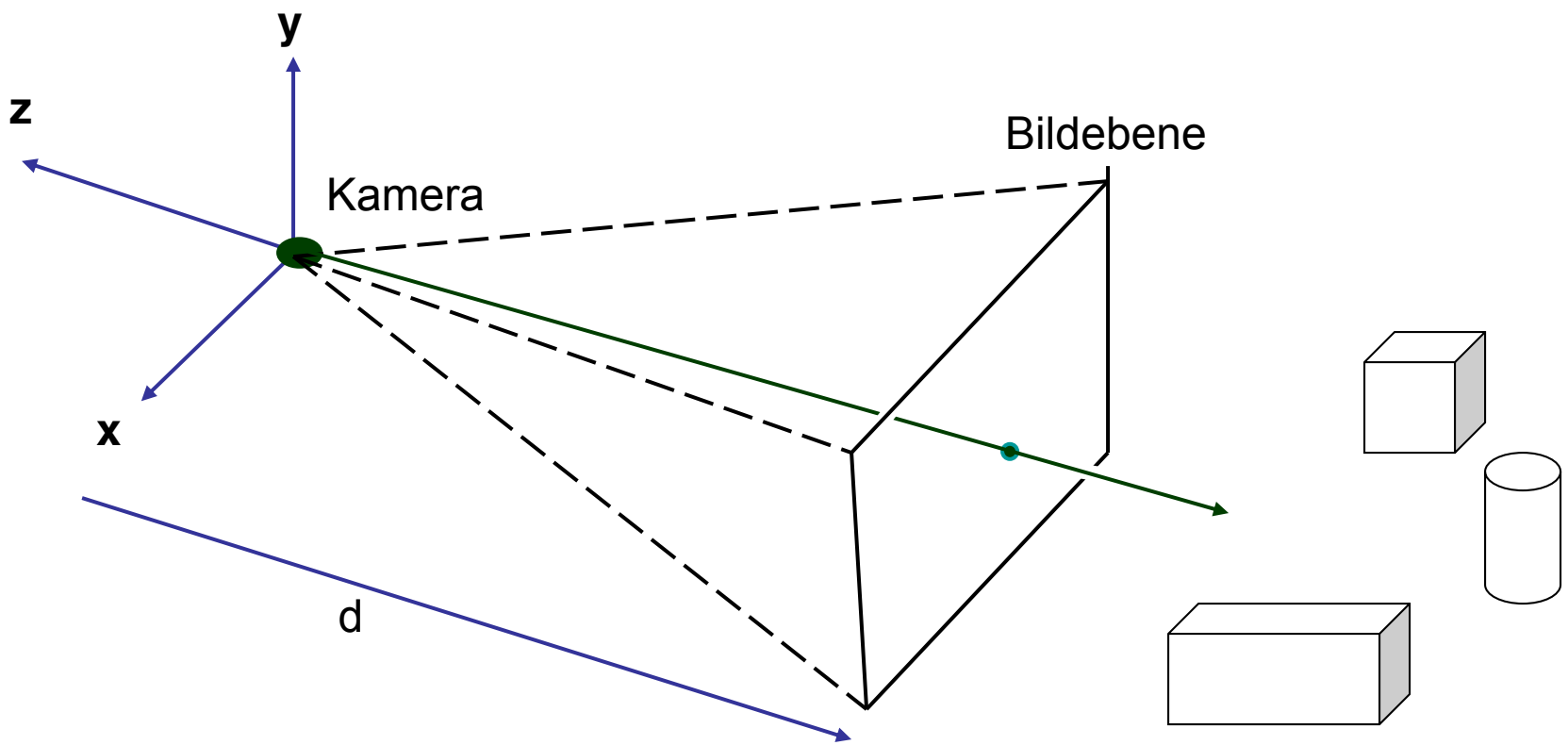
Die Breite der Bildebene ist 1

Die Bildebene ist entsprechend der Auflösung in x- und y-Richtung gerastert

Der Abstand der Bildebene ergibt sich aus dem Kameraöffnungswinkel in x-Richtung

Die Höhe der Bildebene ergibt sich aus dem Kameraöffnungswinkel in y-Richtung

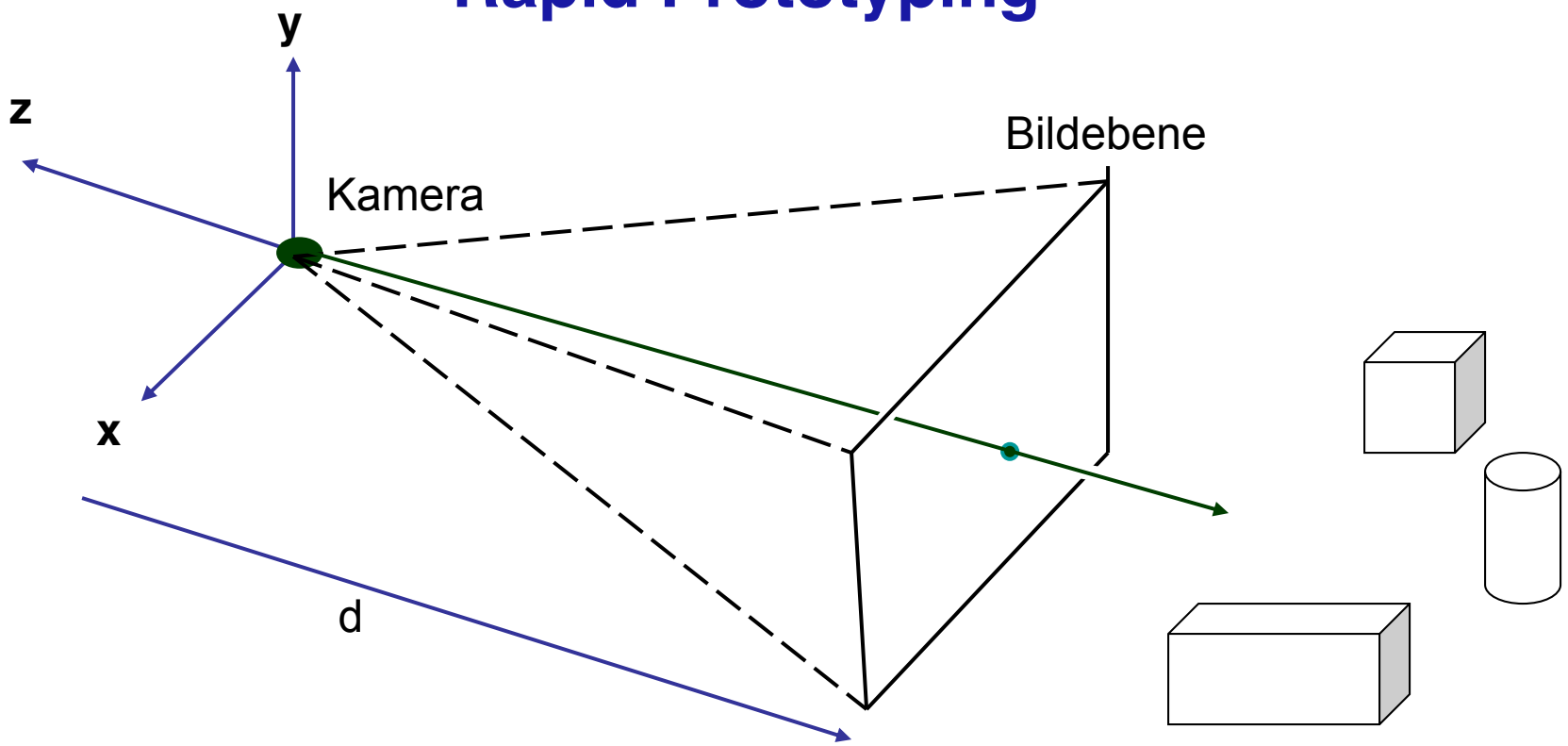
# Generierung der Strahlen für die einfache Kamera




$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} p_{0,x} \\ p_{0,y} \\ p_{0,z} \end{bmatrix} + t \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + t \begin{bmatrix} d_x \\ d_y \\ -d \end{bmatrix}$$

Für quadratisches Bild:
$d_x$ = [-0.5..0.5]
$d_y$ = [-0.5..0.5]

# Generierung der Strahlen für die einfache Kamera
# Rapid Prototyping

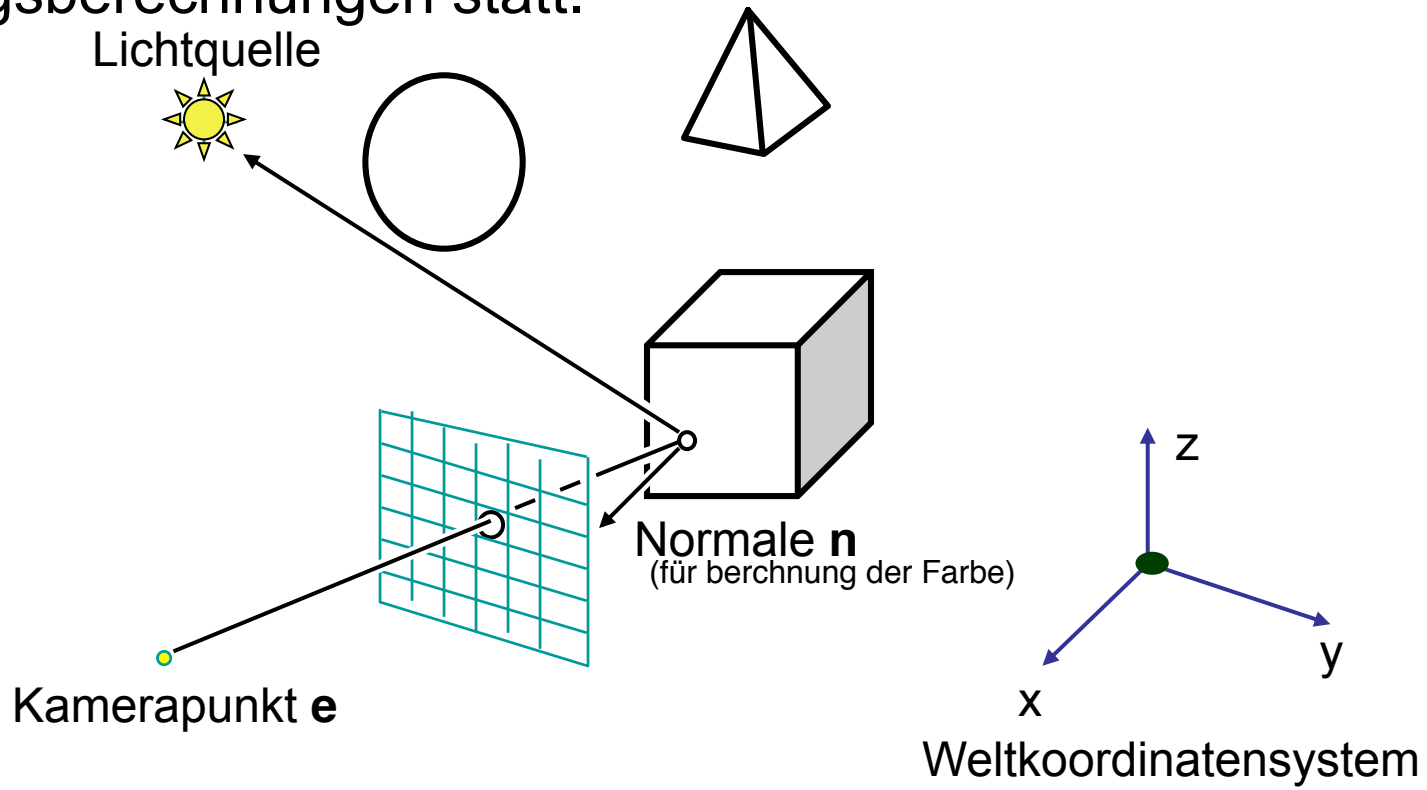



$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} p_{0,x} \\ p_{0,y} \\ p_{0,z} \end{bmatrix} + t \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + t \begin{bmatrix} d_x \\ d_y \\ -d \end{bmatrix}$$

Für quadratisches 401x401 Pixel-Bild:
$d_x$ = [-200..200] Schrittweite 1
$d_y$ = [-200..200] Schrittweite 1
d = 400 // Kamerawinkel 60 Grad

# Modellierung im Weltkoordinatensystem

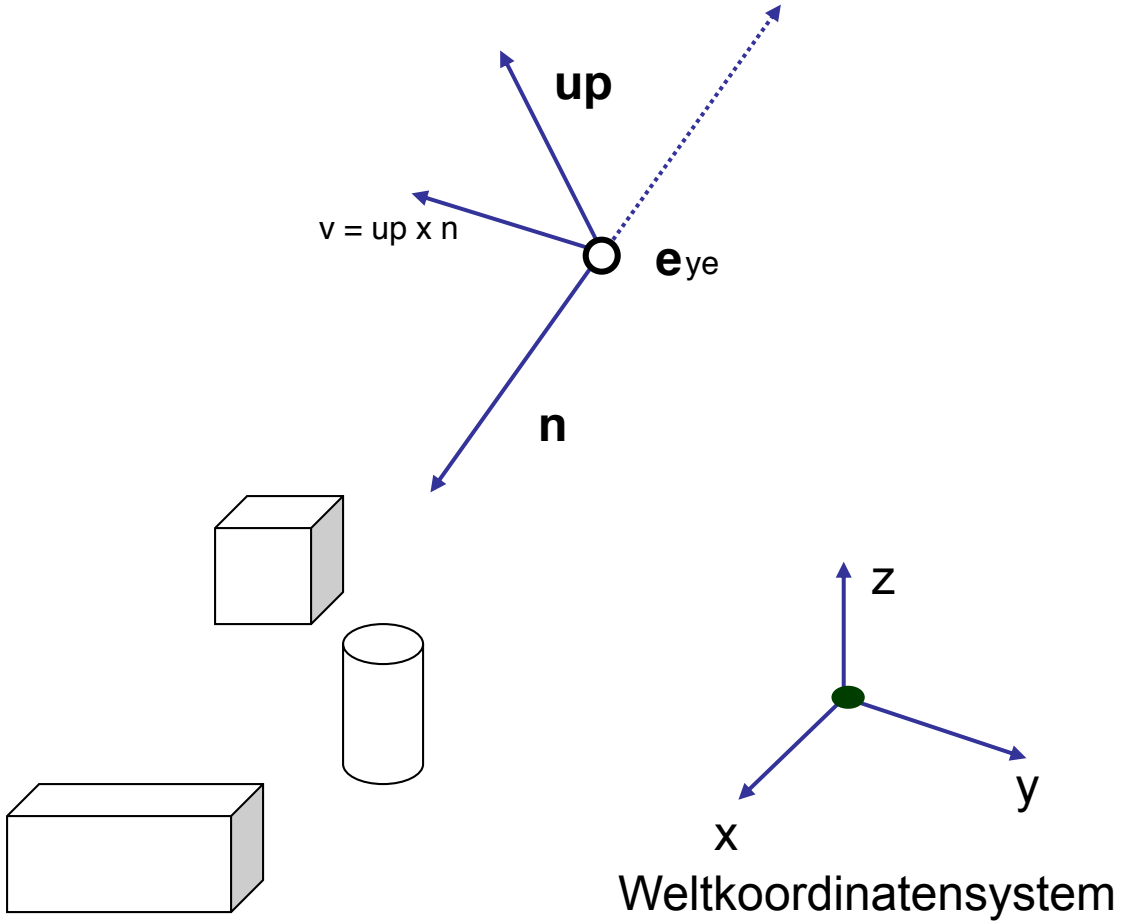
Alle Objekte, sowie die Lichtquellen und die Kamera werden bezüglich eines gedachten Weltkoordinatensystems positioniert. In diesem Koordinatensystem finden auch die Beleuchtungsberechnungen statt.



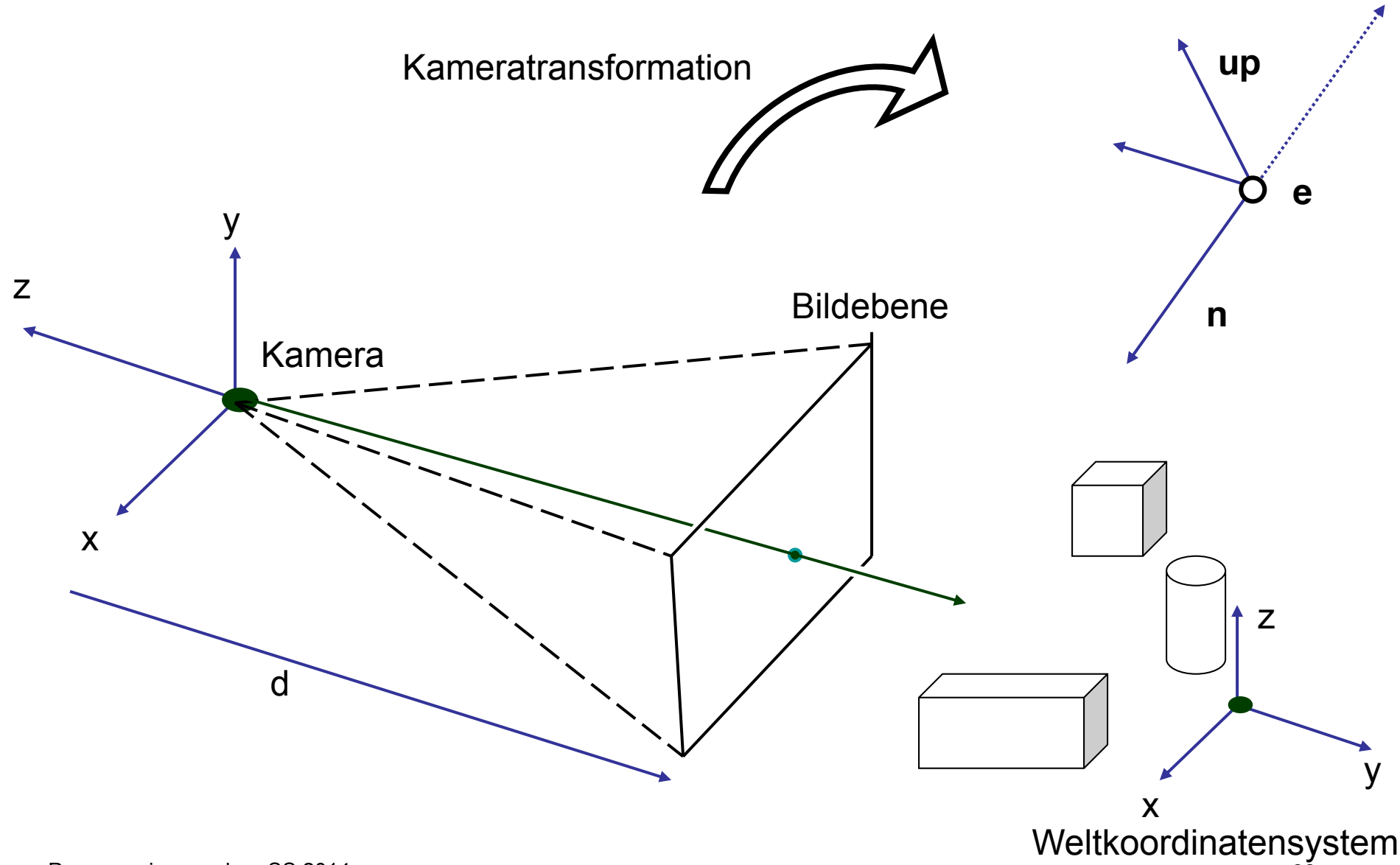
# Positionierung der Kamera im Raum

Die Kameraposition und -orientierung wird durch 3 Angaben festgelegt

- Beobachtungpunkt **e**
- Blickrichtung **n**
- Up-Vektor (oben) **up**



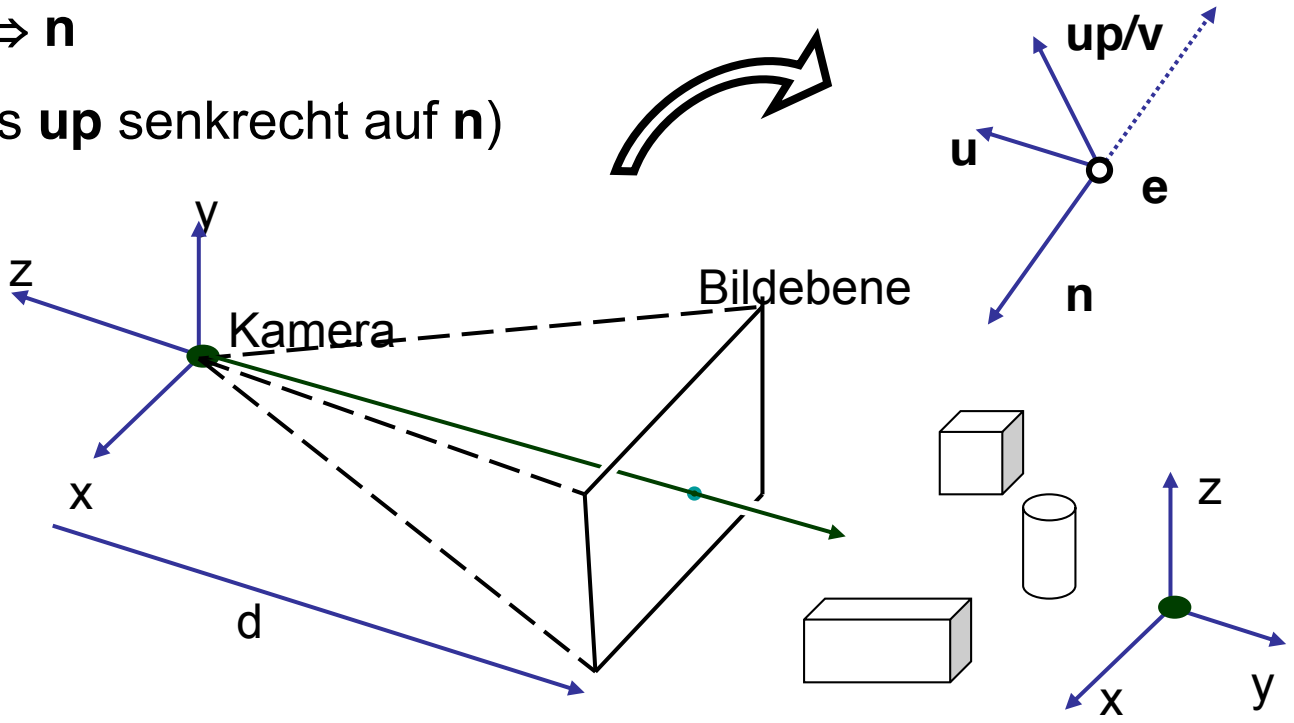
# Kameratransformation



# Kameratransformation

Die Kameratransformation C transformiert die Kamera aus ihrer Standardorientierung in eine beliebige Lage. Dabei werden folgende Abbildungen vorgenommen:

- Ursprung ⇒ **e**
- negative **z**-Achse ⇒ **n**
- **y**-Achse ⇒ **up** (falls **up** senkrecht auf **n**)



# Kameratransformation in Matrixschreibweise

**u** = **n** x **up** (**u** steht senkrecht auf **n** und **up**)

**v** = **u** x **n** (um sicherzustellen, dass **up** senkrecht zu **u** und **n** ist)

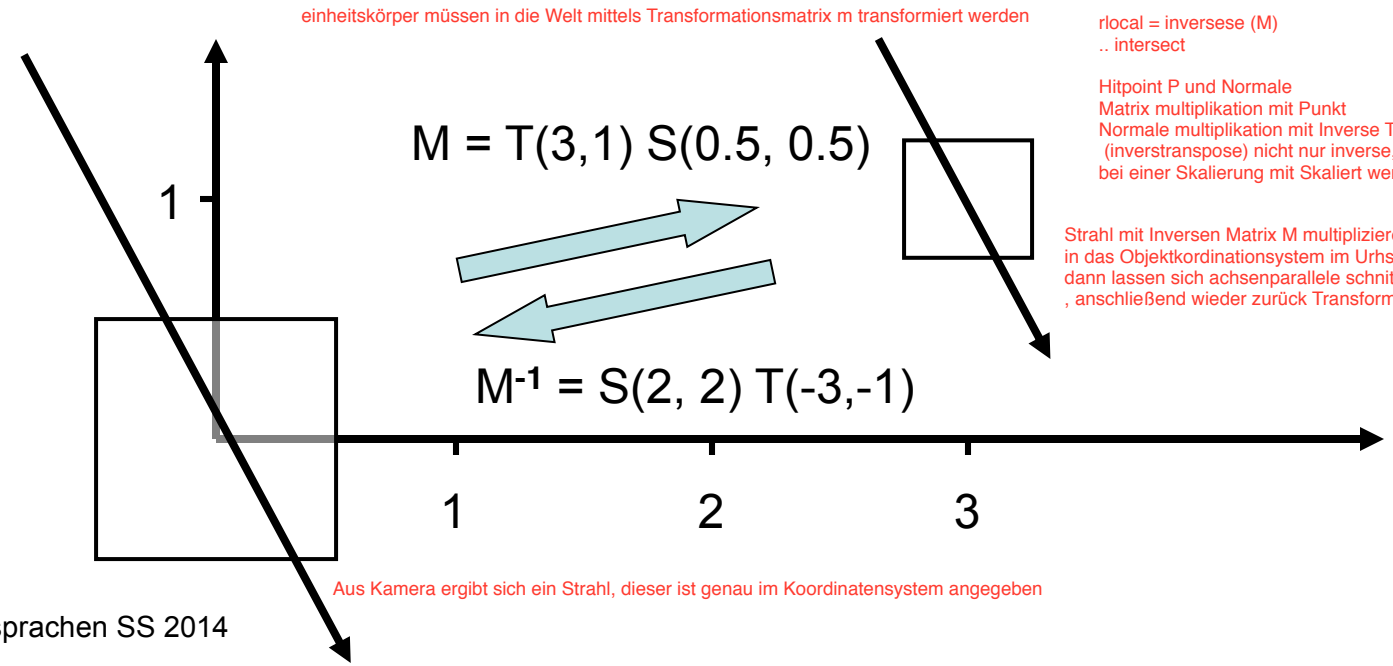
**u**, **v** und **n** sind zu normieren und sind damit die x-, y- und negative z-Achse des Kamerakoordinatenssystems. Damit ergibt sich die Kameratransformation als

$$C = \begin{bmatrix} \mathbf{u}_x & \mathbf{v}_x & -\mathbf{n}_x & \mathbf{e}_x \\ \mathbf{u}_y & \mathbf{v}_y & -\mathbf{n}_y & \mathbf{e}_y \\ \mathbf{u}_z & \mathbf{v}_z & -\mathbf{n}_z & \mathbf{e}_z \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Die Strahlen werden in dem einfachen Kamerakoordinatensystem erzeugt und mit der Matrix C in das beliebig orientierte Kamerakoordinatensystem transformiert.

# Transformation von Objekten

Oft wird bei der Modellierung von Objekten mit einer Standardlage begonnen – zum Beispiel mit einer Kugel mit dem Mittelpunkt im Ursprung und Radius 1 oder einem Würfel mit Kantenlänge 1 und Mittelpunkt im Ursprung. Die Objekte werden dann durch Transformationen M bewegt und in die gewünschte Lage und Grösse gebracht. Zur Schnittberechnung Strahl – Objekt kann man den Strahl invers dazu mit der Matrix $M^{-1}$ transformieren und diesen dann mit dem Objekt in Standardlage schneiden. Durch die Transformation des Strahls hat der Richtungsvektor im allgemeinen nicht mehr die Länge 1 – kann aber auch nicht einfach normiert werden, da dies den Strahlparameter t verändert. Der Strahlparameter t ist invariant unter den Transformationen.



# Identifizieren von Klassen und Methoden

„Wie finde ich die Objekte?“ ist die schwierigste Frage der objektorientierten Analyse.

Es gibt keine eindeutige Antwort

- Man kann ein Problem auf verschiedene Weisen objektorientiert modellieren.
- Es gibt keinen einfachen, festgelegten Weg vom Problem zur Zerlegung in geeignete Objekte

raytracing ist rekursiv, Grundalgo wird von Klasse für jeden Pixel einen srahl erzeugen, für jeden Strah einene Farbe errechnet und einen Schnittpunkt finden welcher beleuchtet wurde.

nicht in die Strahlklasse

# Entwurf nach Zuständigkeit

Ein sehr einfaches, verbreitetes Grundprinzip ist der Entwurf nach Zuständigkeit: Jedes Objekt ist für bestimmte Aufgaben zuständig und

- besitzt entweder die Fähigkeiten, die Aufgabe zu lösen
- oder es kooperiert dazu mit anderen Objekten.

Ziel ist damit das Auffinden von Objekten anhand ihrer Aufgaben in der Kooperation.

# Entwurf nach Zuständigkeit

Wir erstellen zunächst eine **informelle Beschreibung** der Problemstellung, die die **zentralen Anforderungen** an das Software-System verbal enthält.

Für den Entwurf werden dann zentrale Begriffe der verbalen Problembeschreibung betrachtet:

- Substantive in der Beschreibung werden zu Klassen und konkreten Objekten
- Verben in der Beschreibung werden zu Diensten (Methoden) –
  - entweder zu Diensten, die ein Objekt anbietet,
  - oder zu Aufrufen von Diensten kooperierender Objekte.
- Diese Dienste kennzeichnen die Zuständigkeit und Kooperationen der einzelnen Klassen.

# Vorgehen zur Bestimmung der Objekte und deren Methoden und Attribute (nach Abbot)

1. Erstellen einer verbalen Problembeschreibung
2. Identifizieren von Objekten, Attributen und Methoden
   - Substantive werden zu Objekten
   - Eigenschaften/Adjektive werden zu Attributen
   - Verben werden zu Methoden
3. Suchen nach geeigneten Basisklassen
   - Abstraktion
   - Hierarchisierung
4. Vervollständigen der Attribute und Methoden

# Objekte

Ein Objekt bietet eine Sammlung von Diensten (Methoden), die auf einem gemeinsamen Zustand arbeiten.

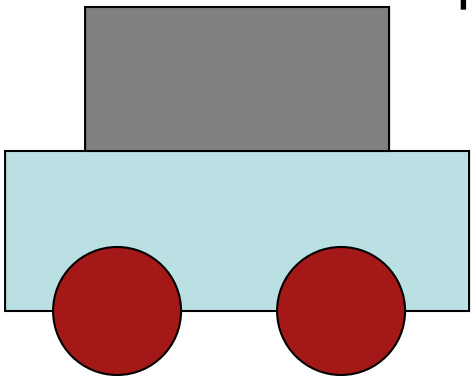
Typischerweise entsprechen

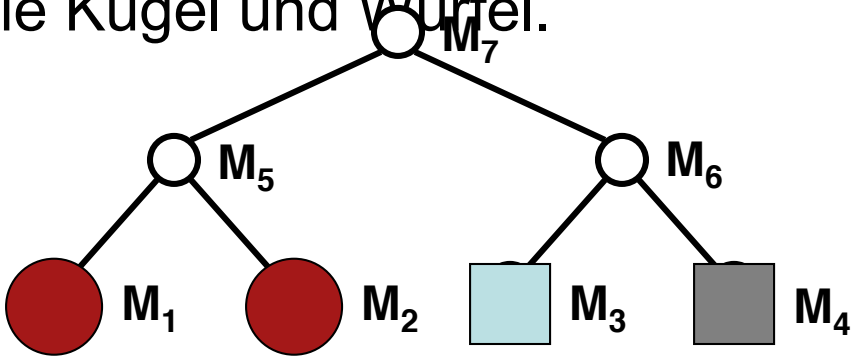
- Objekte den Gegenständen aus der Aufgabenstellung („Kamera“, „Kugel“, „Strahl“)
- Attribute den Eigenschaften der Gegenstände aus der Aufgabenstellung (Gegenstand „Strahl“: Eigenschaft: „Ursprung“, „Richtung“, Parameter“)
- Methoden den Verben aus der Aufgabenstellung („Strahlen erzeugen“, „schneiden“, „löschen“)

# ENDE

# Zusammengesetzte Objekte / Composites

Oft möchte man Objekte gemeinsam bewegen oder logisch zusammenfassen. Man führt dazu eine Modellierungshierarchie ein. Innere Knoten tragen eine Transformationsmatrix und verwalten eine Liste von darunter liegenden Knoten. Blätter der Hierarchie sind die geometrischen Grundkörper, wie Kugel und Würfel.

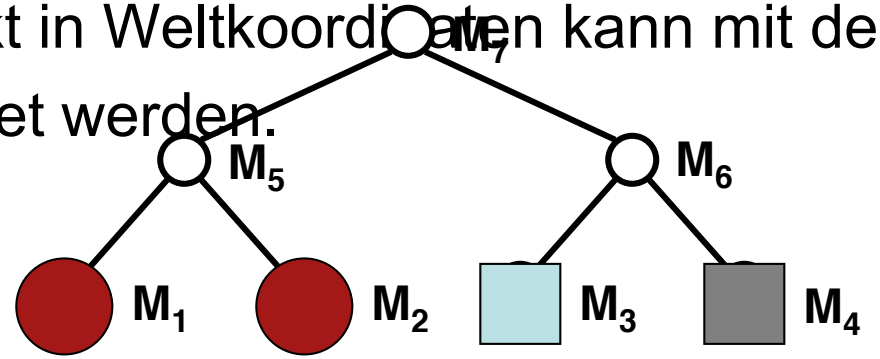





Falls Knoten mehrere Väter haben, dann erhält man einen gerichteten azyklischen Graphen (directed acyclic graph) DAG.

# Strahlschnitt mit zusammengesetzten Objekten

Die Modellierungshierarchie wird im allgemeinen rekursiv durchlaufen und der Strahl wird in jedem Knoten mit der inversen Matrix des Knotens transformiert. Der Normalenvektor zur Beleuchtungsberechnung muss dann entsprechend mit der transponierten invertierten kumulierten Pfadmatrix von der Wurzel bis zum Grundkörper transformiert werden. Liegt zum Beispiel ein Schnittpunkt mit Objekt 1 vor, dann muss der Normalenvektor am Schnittpunkt mit $((M_7\ M_5\ M_1)^{-1})^T = (M_1^{-1}\ M_5^{-1}\ M_7^{-1})^T$ transformiert werden. Der Strahlschnittpunkt in Weltkoordinaten kann mit dem Strahlparameter *t* ausgerechnet werden.



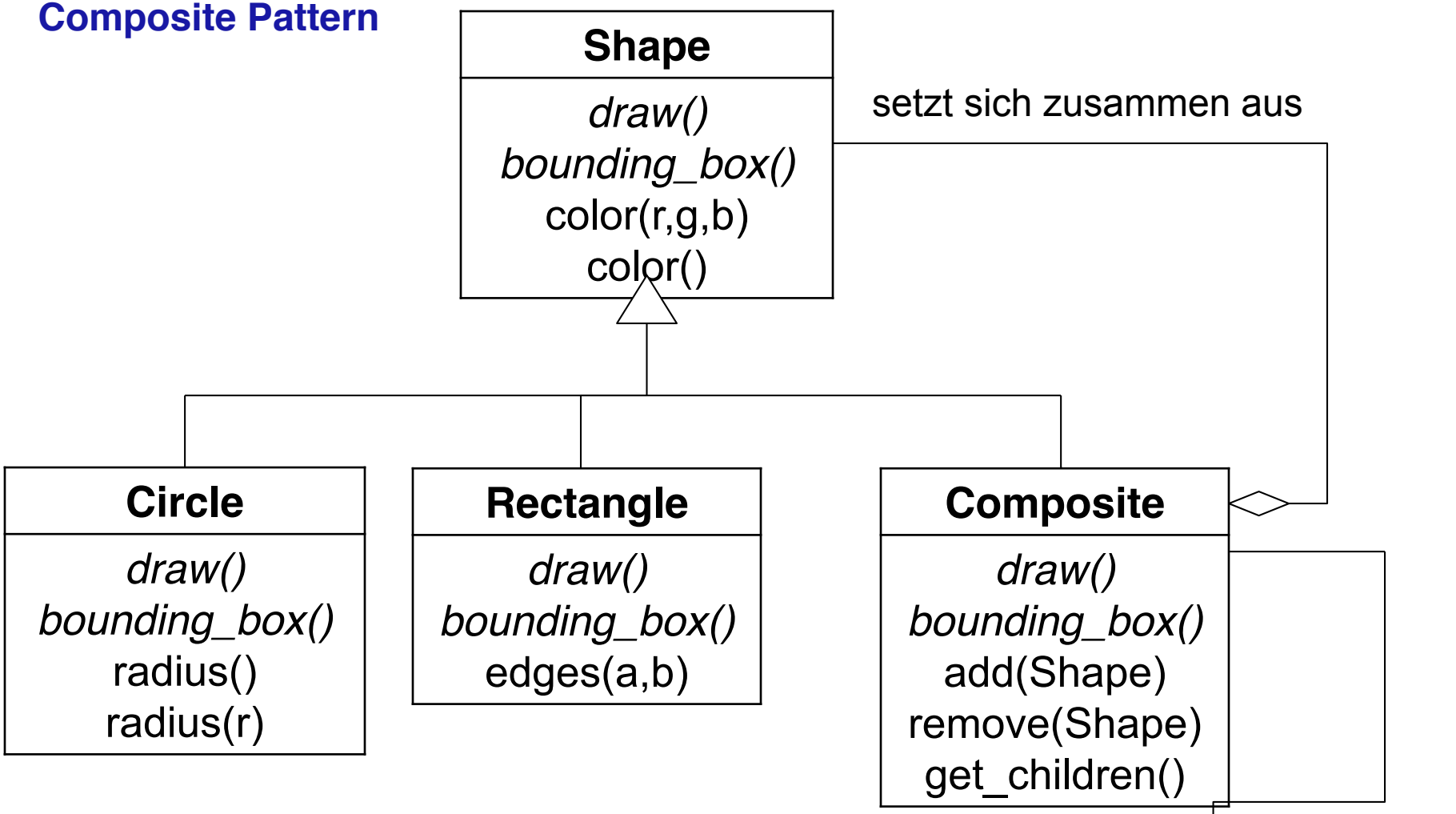
# Objektorientierte Modellierung

Die objektorientierte Modellierung mit UML (Unified Modeling Language) umfasst u. a. folgende Aspekte des Entwurfs:

- Erstellen eines Objekt-Modells: statische Strukturen von Klassen und Objekten, sowie deren Verhalten und Relationen werden grafisch in Klassendiagrammen, Objektdiagrammen, … festgelegt und veranschaulicht
- Welche Merkmale besitzen diese Objekte?
  - Attribute
  - Methoden
- Wie lassen sich diese Objekte klassifizieren?
  - Klassen und Klassenhierarchie
- Klassendiagramm: Welche Assoziationen bestehen zwischen den Klassen
- Objektdiagramm: Welche Assoziationen bestehen zwischen den Objekten?
- Sequenzdiagramm: Wie wirken die Objekte global zusammen?
- Zustandsdiagramm: In welchen Zuständen befinden sich die Objekte?

# Klassendiagramm

## Composite Pattern




Composite: Gruppierung von Objekten
Shape: abstrahiert beliebige grafische Objekte