

Computergrafik

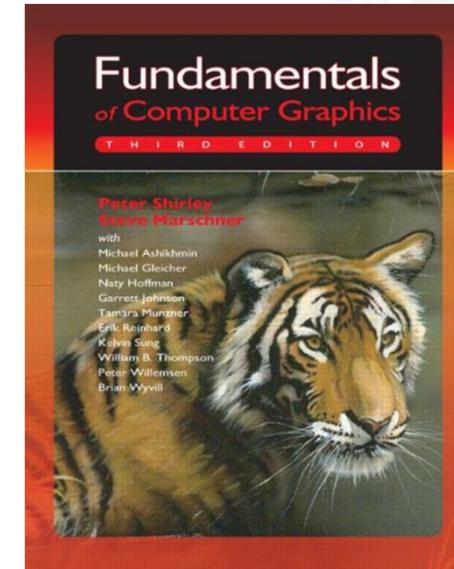
**Vorlesung im Wintersemester 2014/15
Kapitel 2: Ray Tracing**

Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie

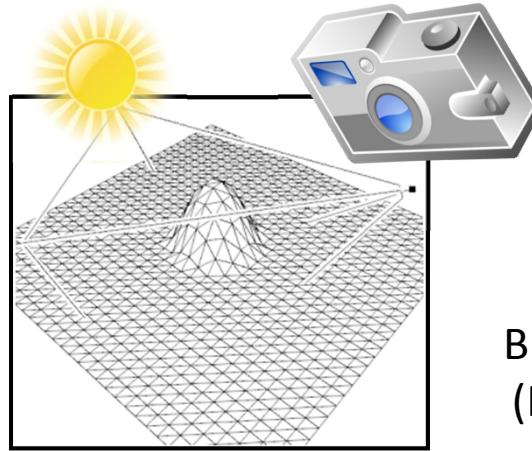
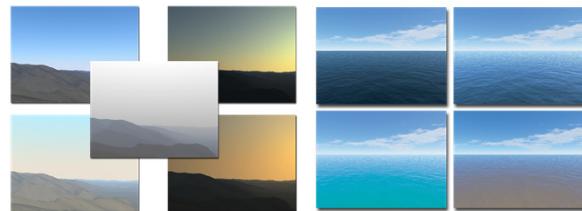


Literatur

- ▶ **Fundamentals of Computer Graphics,**
P. Shirley, S. Marschner, 3rd Edition, AK Peters
→ Kapitel 4



Computergrafik und Farbbilder



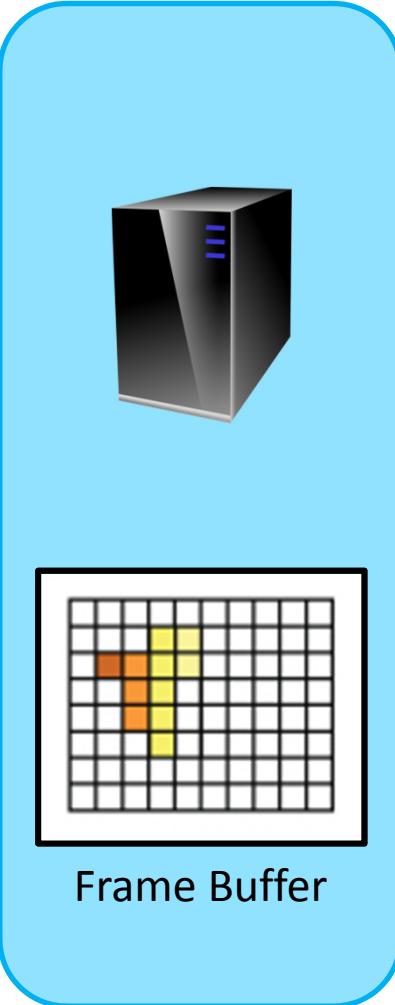
Bildsynthese
(Rendering)

Virtuelle Szene
(Geometrie, Material, Kamera, Lichtquellen, ...)



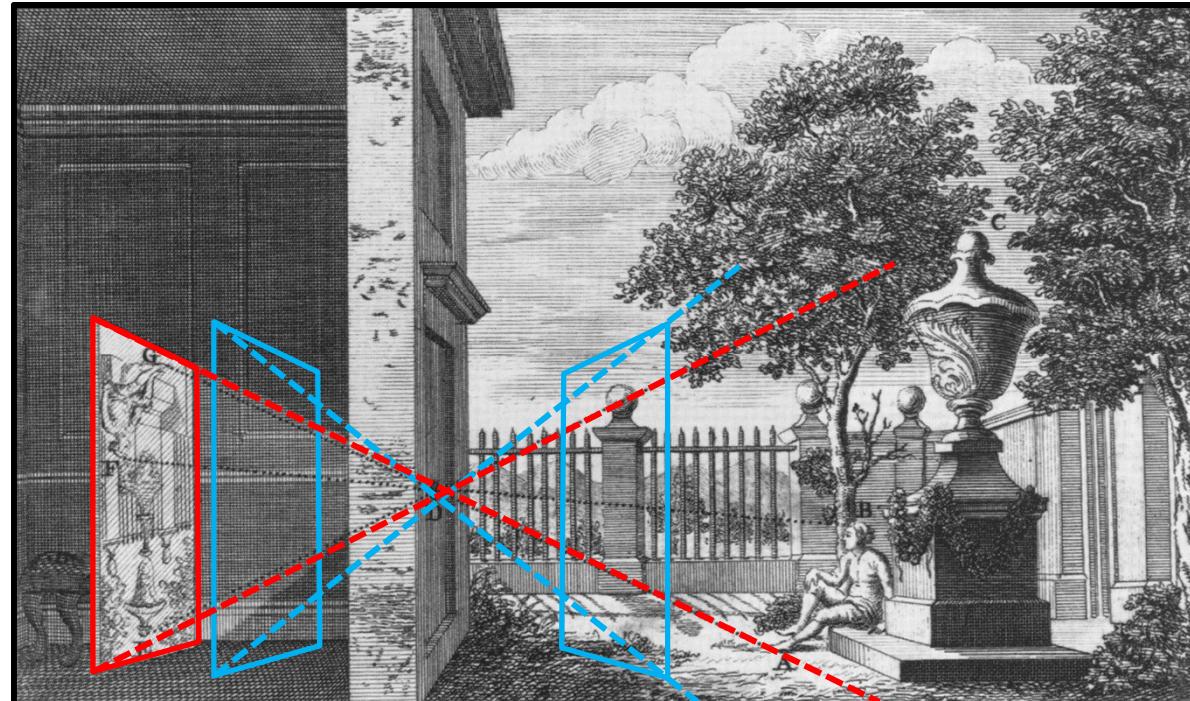
Monitor

Ansteuerung des
Monitors



Lochkamera

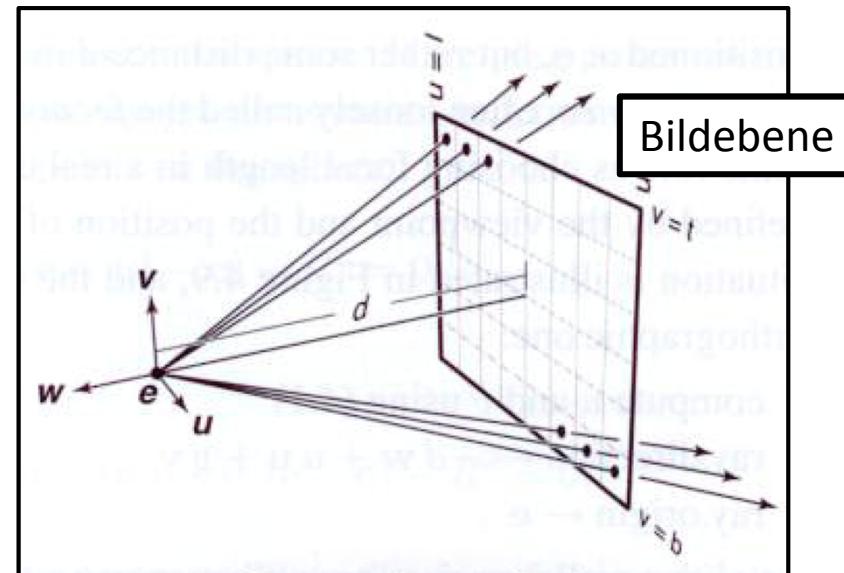
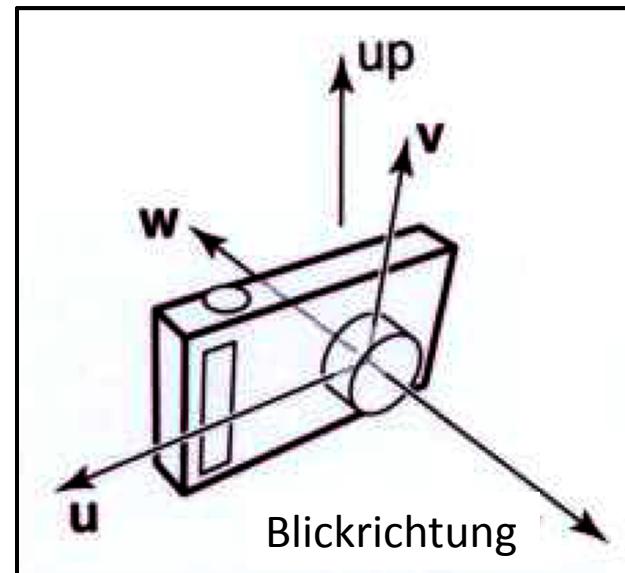
- ▶ in der CG verwendet man am einfachsten das Modell der Lochkamera
 - ▶ Kammer mit kleiner Öffnung → auf der gegenüberliegenden Wand entsteht ein spiegelverkehrtes Bild



- ▶ (im Prinzip) unbegrenzte Schärfentiefe
- ▶ Kamera definiert durch Position der Öffnung und Bildebene

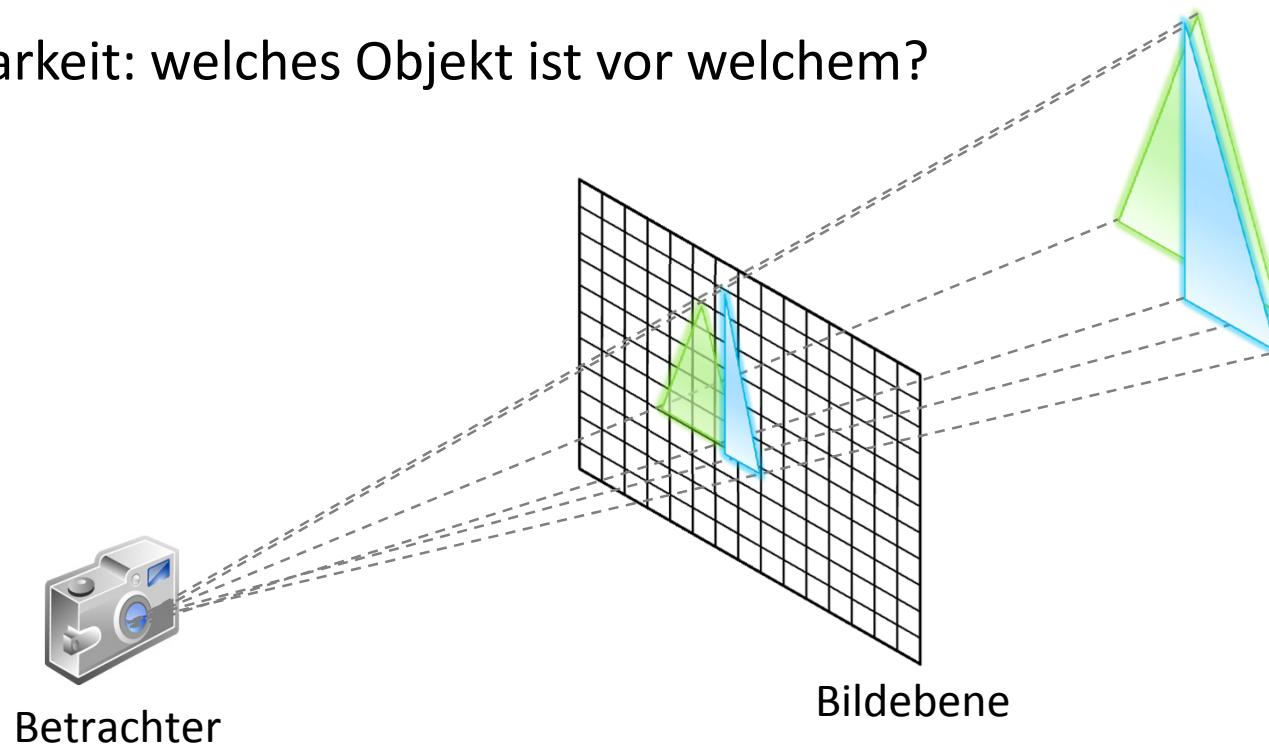
Virtuelle Kamera

- ▶ in der CG verwendet man am einfachsten das Modell der Lochkamera
- ▶ eine virtuelle Kamera ist definiert durch
 - ▶ Position und Blickrichtung
 - ▶ Orientierung der vertikalen Achse
 - ▶ eine Bildebene mit Breite, Höhe und Abstand **vor** der Kamera
 - ▶ die Bildebene entspricht dem Bild der virtuellen Kamera



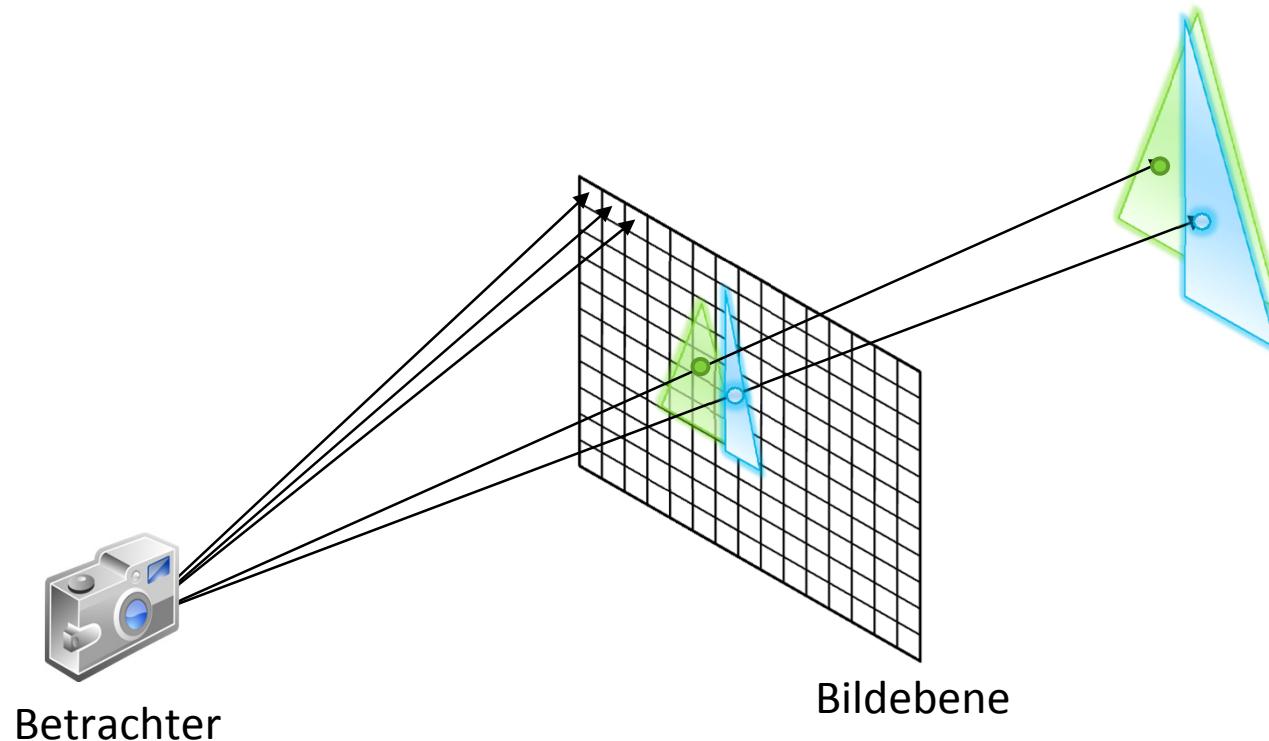
Rasterisierung (Object Order)

- ▶ Grundidee: Transformation von Geometrie
 - ▶ 3D-Objekte werden auf die 2D-Bildebene projiziert
 - ▶ für jedes Objekt
 - finde alle Pixel, die das Objekt bedeckt/beeinflusst
 - bestimme Pixelfarbe
 - ▶ Sichtbarkeit: welches Objekt ist vor welchem?



Ray Tracing (Image Order)

- ▶ Grundidee: geometrische Überlegungen über Lichtstrahlen
 - ▶ verfolge die Lichtstrahlen, die die Öffnung der Lochkamera passieren
 - ▶ für jeden Pixel
 - finde alle Objekte, die den Pixel beeinflussen
 - bestimme Pixelfarbe



- ▶ Bildsynthese (engl. Rendering)
 - ▶ erzeugt ein Rasterbild aus einer Szenenbeschreibung (Objekte)
 - ▶ also: bestimme, welche Objekte die Farbe jedes Pixels beeinflussen
- ▶ **Objektbasiert (Object-Order Rendering)**
 - ▶ betrachte ein Objekt/eine Fläche nach der anderen
 - ▶ finde heraus, welche Pixel das Objekt bedeckt
 - ▶ bestimme die Pixelfarbe
- ▶ **Bildbasiert (Image-Order Rendering)**
 - ▶ betrachte einen Pixel nach dem anderen
 - ▶ finde heraus, welches Objekt an dieser Stelle sichtbar ist
 - ▶ bestimme die Pixelfarbe

Ray Tracing Beispiel: Metall- und Glaskugeln



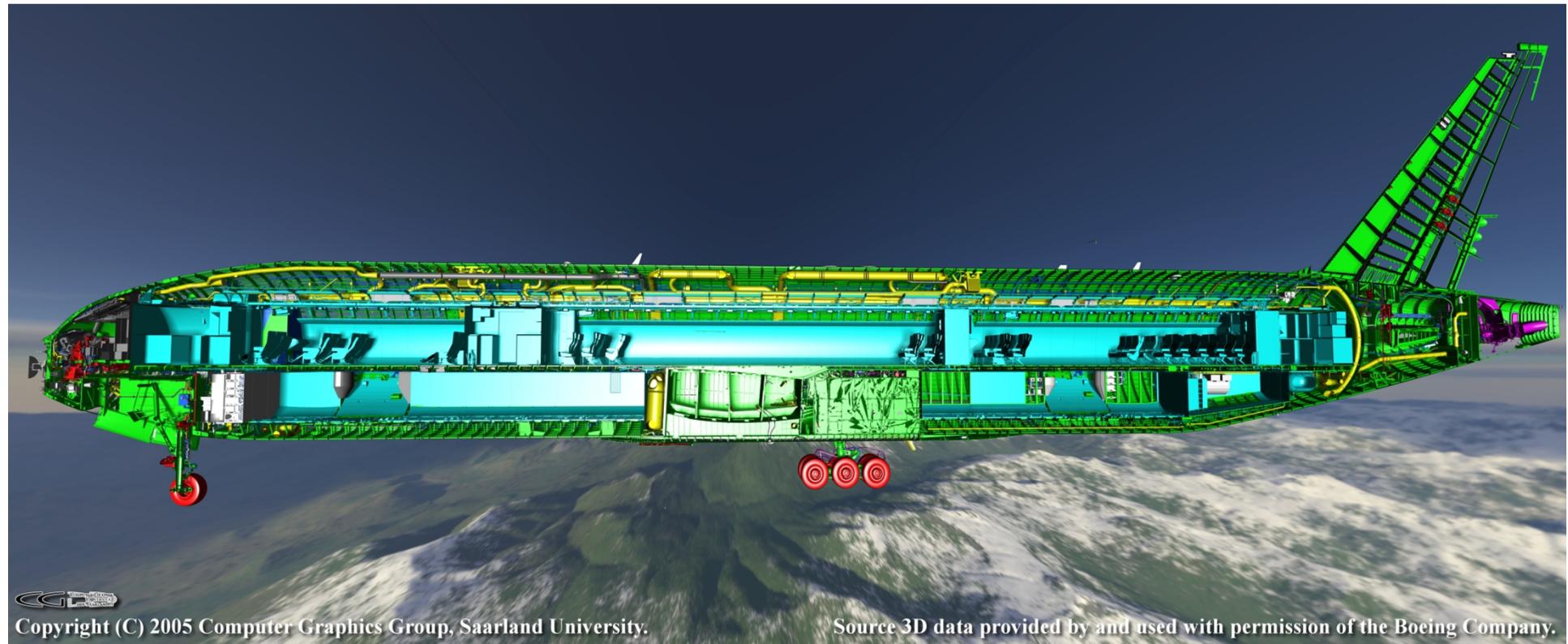
► Bild: Pat Hanrahan



Ray Tracing Beispiele



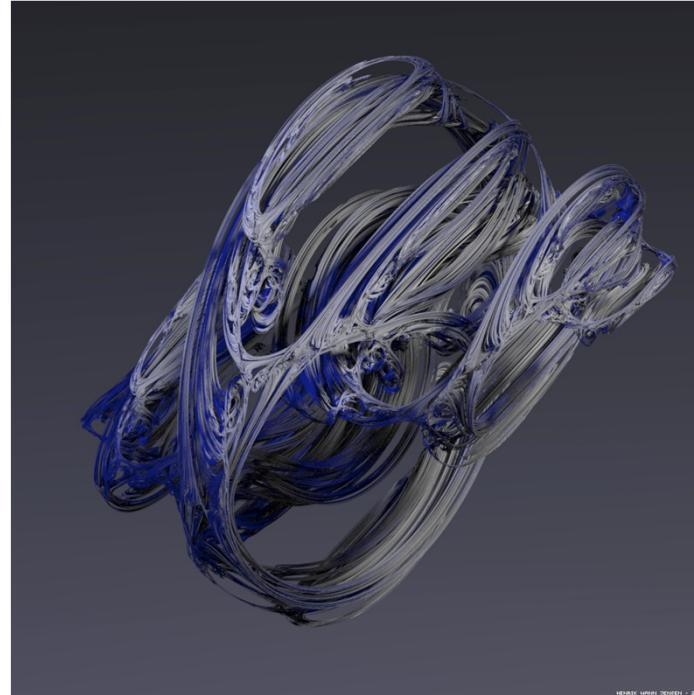
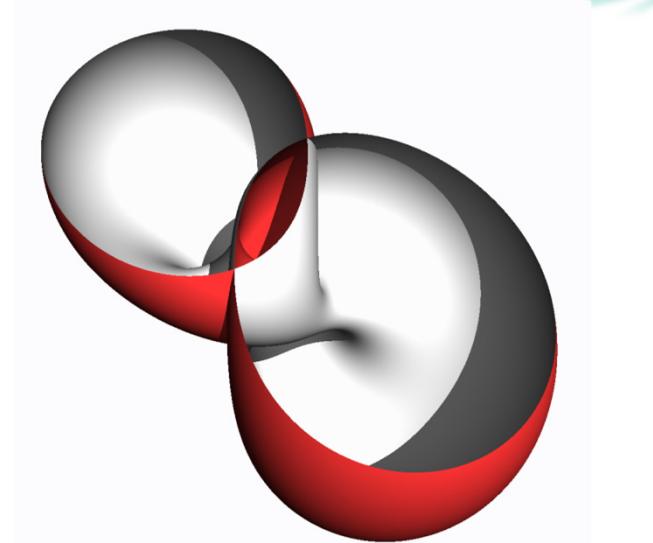
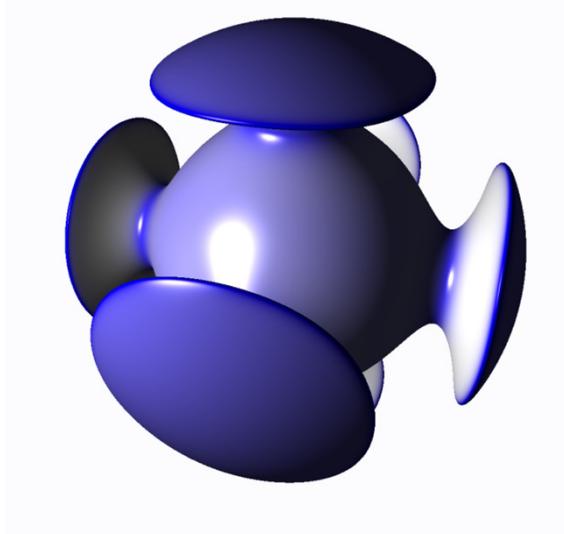
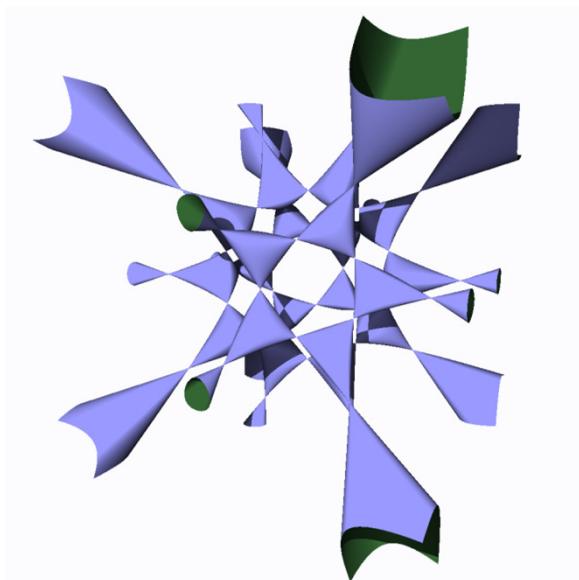
- komplexe Geometrie: 350 Mio. Dreiecke



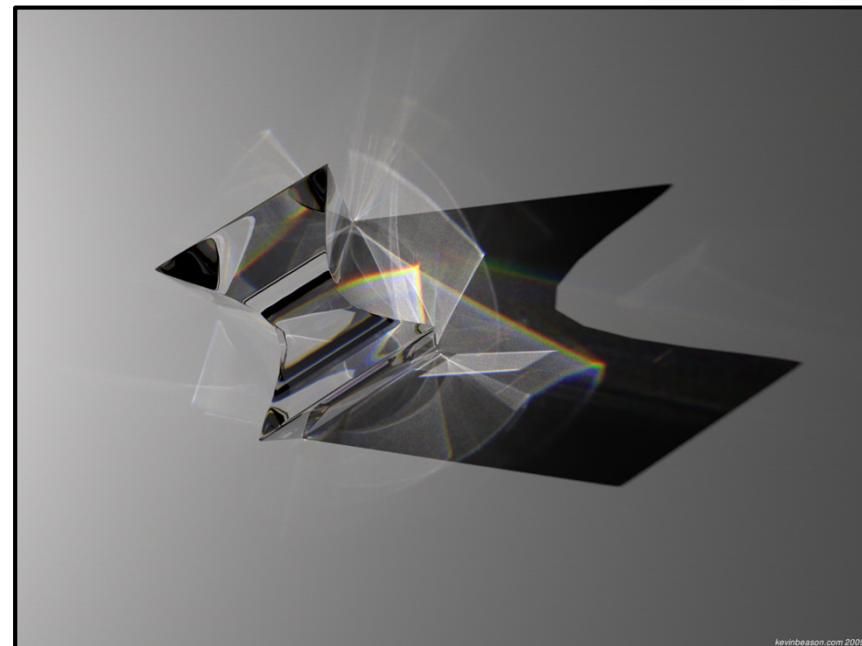
Copyright (C) 2005 Computer Graphics Group, Saarland University.

Source 3D data provided by and used with permission of the Boeing Company.

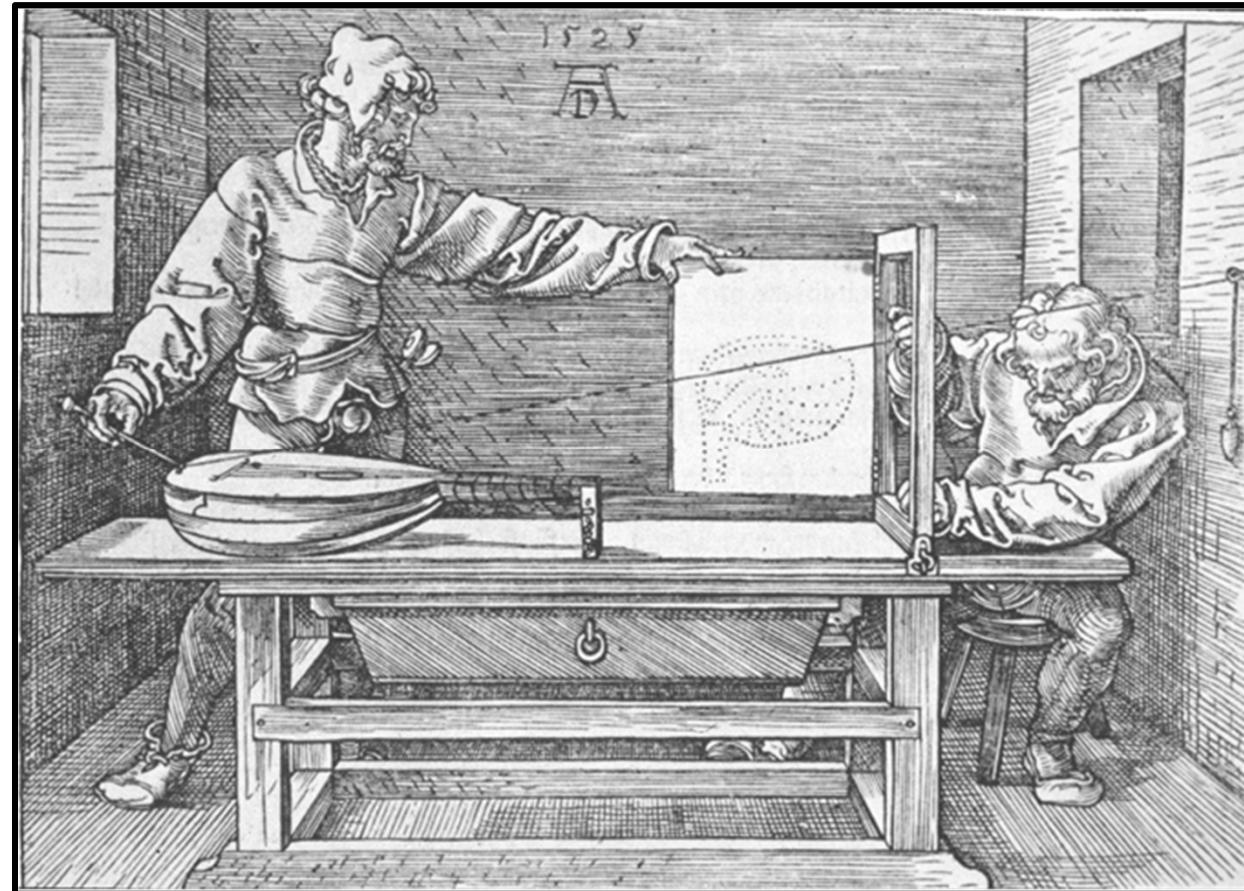
Ray Tracing Beispiele



Ray Tracing Beispiele



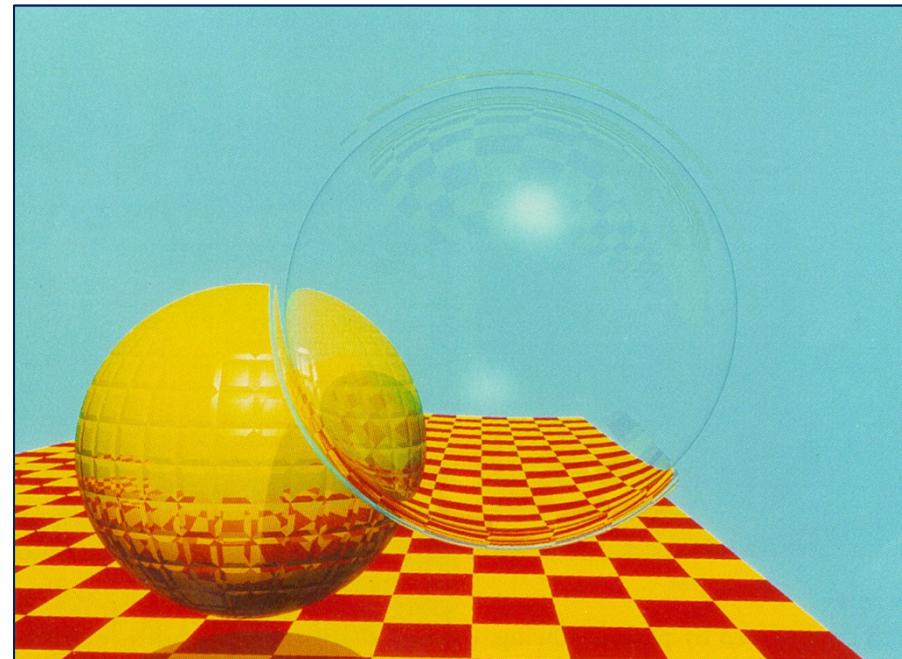
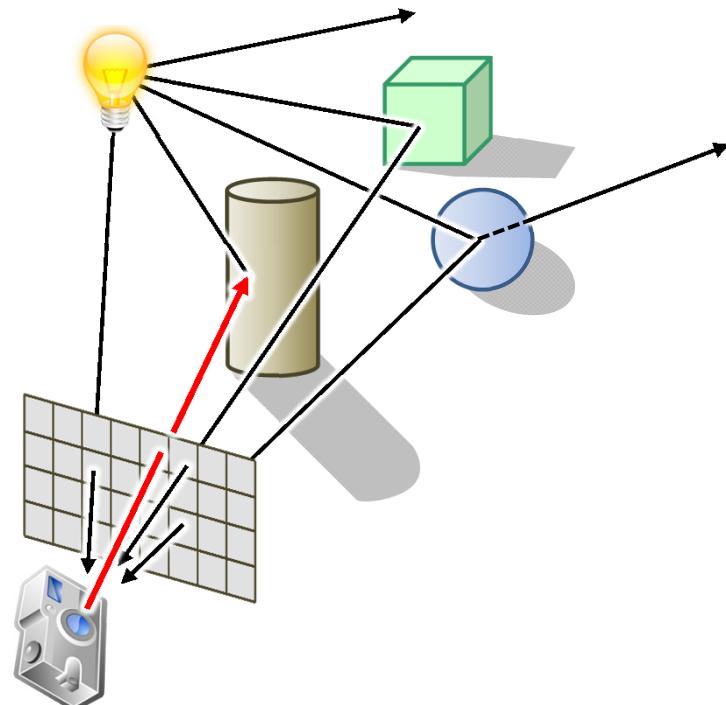
Ray Tracing?



Albrecht Dürer: „Underweysung der messung mit dem Zirkel und
richtscheyt, in Linien Ebnen un gantzen Corporen“, 1525

Ray Tracing

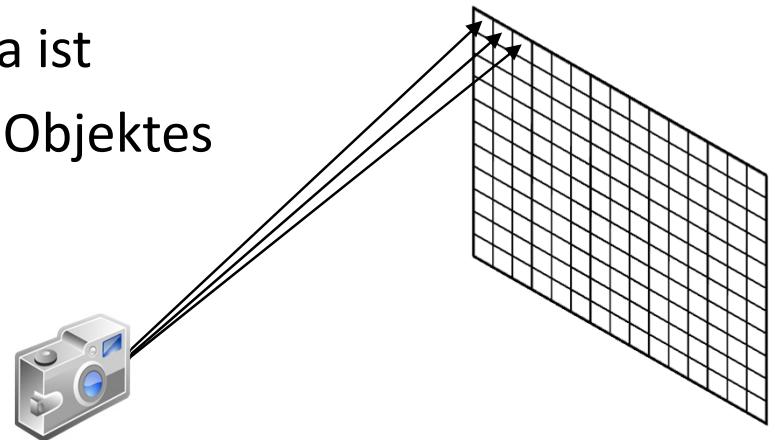
- ▶ Rückwärts-Lichttransport („von wo kommt Licht zur Kamera“)
 - ▶ starte an der Kamera
 - ▶ suche Pfade, auf denen das Licht dort hinkommt
- ▶ Annahme: Lichttransport folgt den Gesetzen der geometrischen Optik



[T. Whitted: An Improved Illumination Model
for Shaded Display, 1980]

Ray Tracing: Prinzip

- ▶ betrachte einen Pixel nach dem anderen
- ▶ finde das Objekt, das die Kamera „durch“ diesen Pixel sieht
 - ▶ Sichtstrahl: eine Halbgerade von der Kamera durch den Pixel
 - ▶ gesuchtes Objekt ist das, das den Sichtstrahl schneidet und dessen Schnittpunkt am nahsten an der Kamera ist
- ▶ berechne die Farbe und Schattierung des Objektes



- ▶ ein Ray Tracing-Programm besteht also aus
 - ▶ Erzeugung der Sichtstrahlen (ray generation)
 - ▶ Schnittberechnung (ray intersection)
 - ▶ Schattierung/Beleuchtungsberechnung (shading)

Analytische Geometrie



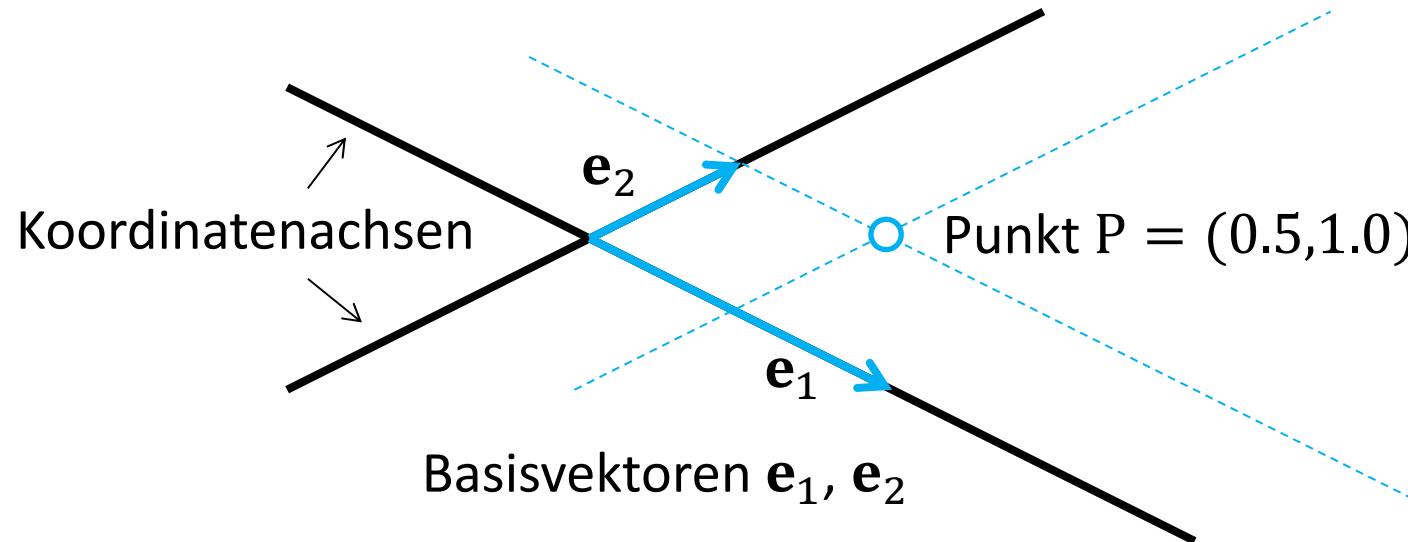
- ▶ die Geometrie ist ein Grundpfeiler der Computergrafik
 - ▶ Euklid (300 v.Chr.), René Descartes (*1596 +1650)

- ▶ ein kleines bisschen analytische Geometrie...
 - ▶ Koordinatensysteme, Vektoren und Matrizen
 - ▶ (Halb-)Geraden, Ebenen, ...
 - ▶ Ziel: Auffrischen der Kenntnisse, wichtige Konzepte für die Computergrafik

- ▶ Konventionen
 - ▶ $\alpha, \beta, \gamma, \dots$ reelle Zahlen, Skalare
 - ▶ a, b, c, \dots reelle Zahlen
 - ▶ i, j, k, \dots ganze Zahlen
 - ▶ $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$ Vektoren
 - ▶ P, Q, R, ... Punkte
 - ▶ $\mathbf{P}, \mathbf{Q}, \mathbf{R}, \dots$ Ortsvektoren
 - ▶ s, t, u, v, \dots Parameterwerte



Parallel-Koordinatensysteme

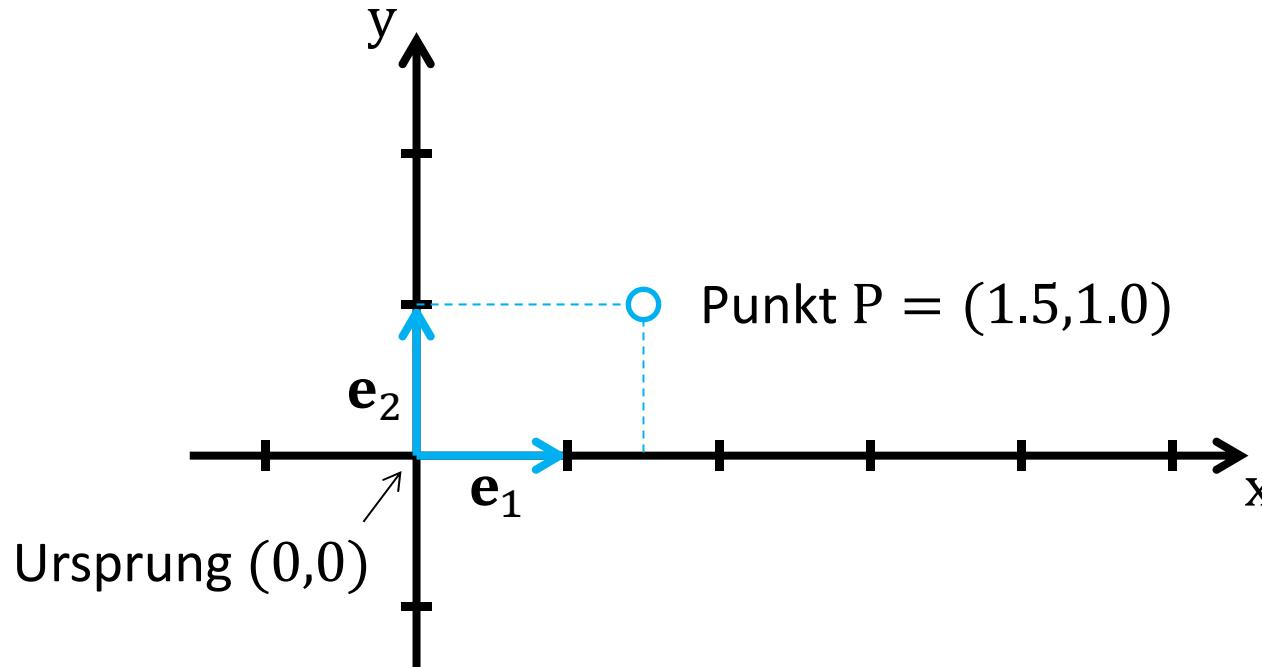


- ▶ nicht notwendigerweise rechtwinklig
- ▶ Basisvektoren nicht zwingend gleich lang
- ▶ Lage eines Punktes in der Ebene durch 2 Koordinaten eindeutig festgelegt
- ▶ Verallgemeinerung in höhere Dimensionen leicht möglich

Koordinatensysteme



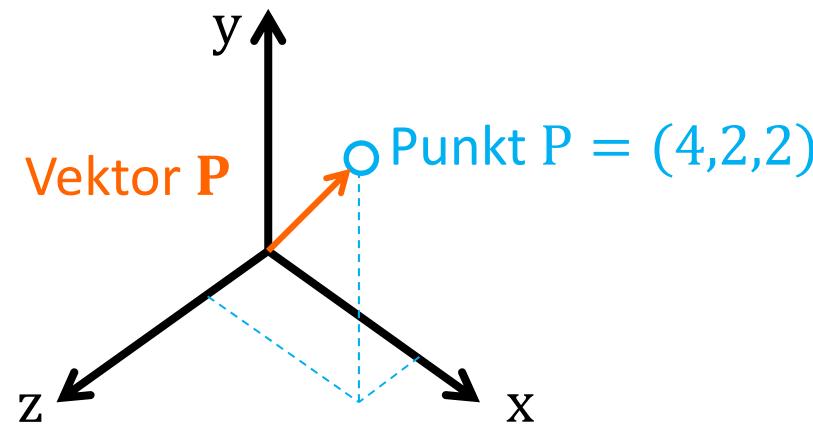
Kartesisches Koordinatensystem (nach René Descartes *1596 +1650)



- ▶ Koordinatenachsen senkrecht
- ▶ Einheitsvektoren gleicher Länge (Einheit zur Längenmessung, z.B. m)
- ▶ Basisvektoren e_1, e_2, \dots bilden eine Orthonormalbasis
- ▶ wird nichts anderes gesagt, dann beziehen wir uns auf kartesische Koord.

Vektor, Punkt und Ortsvektor

- ▶ ein Vektor beschreibt anschaulich eine Länge und Richtung („Pfeil“)
- ▶ geben i.d.R. Richtungen oder Verschiebungen an



- ▶ Ortsvektor **P** bezeichnet den Vektor
 - ▶ von einem Bezugspunkt (typ. Koordinatenursprung)
 - ▶ zu einem Aufpunkt mit den Punktkoordinaten $P = (p_1, p_2, \dots, p_n)$
 - ▶ p_i sind die Gewichtungsfaktoren der Linearkombination der aufspannenden Vektoren

Vektor, Punkt und Ortsvektor



- ▶ beachte den semantischen Unterschied
 - ▶ wir addieren zwei Vektoren
 - ▶ z.B. Hintereinanderausführung von Verschiebungen
 - ▶ wir addieren keine Ortsvektoren
 - ▶ es sei denn, als Zwischenoperation, z.B. Berechnung eines Mittelpunkts oder des Differenzvektors
- ▶ Rechenoperationen
 - ▶ Länge eines Vektors: $|\mathbf{a}| = \sqrt{\sum_{i=1}^n a_i^2}$
 - ▶ Einheitsvektor $\leftrightarrow |\mathbf{a}| = 1$
 - ▶ Vektoraddition: $\mathbf{a} \pm \mathbf{b} = (a_1 \pm b_1, a_2 \pm b_2, \dots, a_n \pm b_n)$
 - ▶ Skalierung: $\lambda \mathbf{a} = (\lambda a_1, \lambda a_2, \dots, \lambda a_n)$

Skalarprodukt (inneres Produkt, engl. dot product)

► Definition

► $\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$

► als Matrix-Multiplikation

► $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$

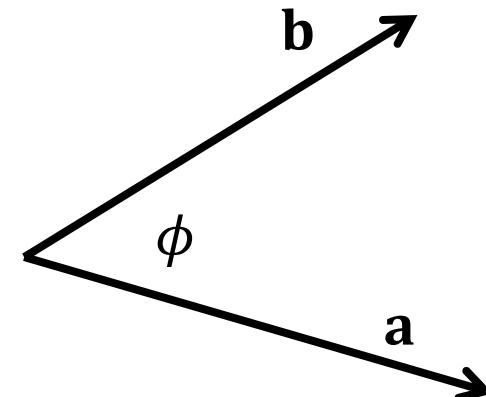
► Eigenschaften

► $\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| |\mathbf{b}| \cos \phi$

► $\mathbf{a} \cdot \mathbf{a} = |\mathbf{a}|^2$

► $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$ (kommutativ)

► $\mathbf{a} \cdot (\mathbf{b} \pm \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} \pm \mathbf{a} \cdot \mathbf{c}$ (distributiv)



$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| |\mathbf{b}| \cos \phi$$

Vektoren

Kreuzprodukt (auch äußeres/vektorielles Produkt, engl. cross product)

► Definition (in 3D)

$$\blacktriangleright \mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix} = \mathbf{n} |\mathbf{a}| |\mathbf{b}| \sin \phi$$

► $|\mathbf{n}| = 1$ und \mathbf{n} steht senkrecht auf \mathbf{a} und \mathbf{b}

► Graßmann, Lagrange, ... Identitäten

$$\blacktriangleright \mathbf{a} \cdot (\mathbf{a} \times \mathbf{b}) = \mathbf{b} \cdot (\mathbf{a} \times \mathbf{b}) = 0$$

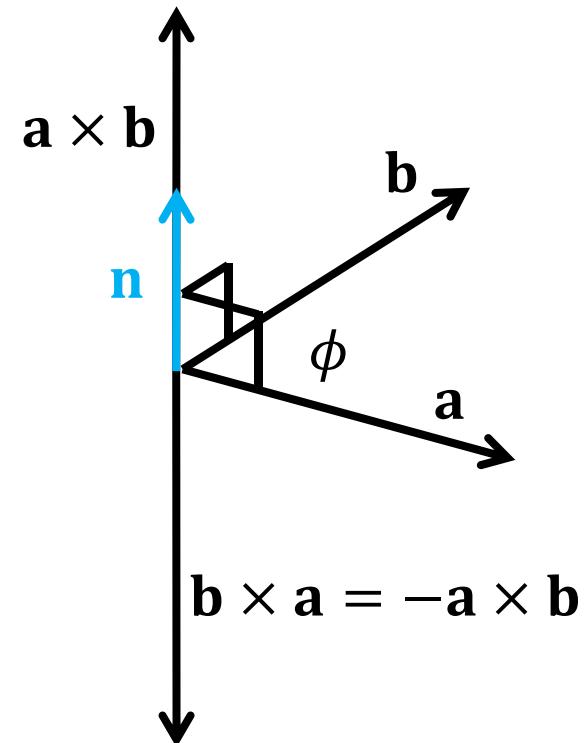
$$\blacktriangleright \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}$$

$$\blacktriangleright \mathbf{a} \times (\mathbf{b} + \mathbf{c}) = \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c}$$

$$\blacktriangleright \mathbf{a} \times (k\mathbf{b}) = k(\mathbf{a} \times \mathbf{b})$$

$$\blacktriangleright \mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c}) \cdot \mathbf{b} - (\mathbf{a} \cdot \mathbf{b}) \cdot \mathbf{c}$$

$$\blacktriangleright (\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = (\mathbf{a} \cdot \mathbf{c})(\mathbf{b} \cdot \mathbf{d}) - (\mathbf{b} \cdot \mathbf{c})(\mathbf{a} \cdot \mathbf{d})$$



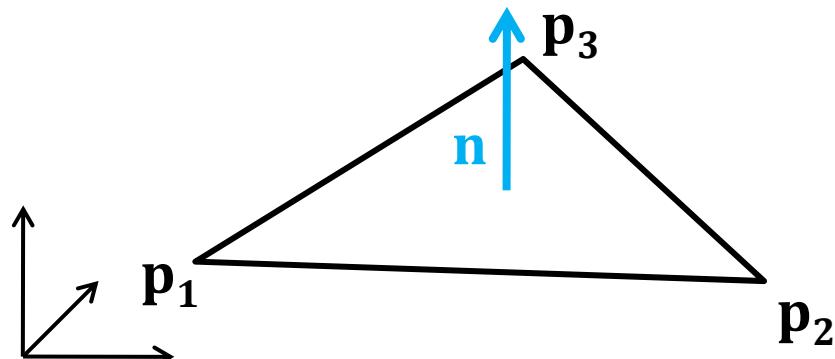
Vektoren

Kreuzprodukt (auch äußeres/vektorielles Produkt, engl. cross product)

► Definition (in 3D)

$$\blacktriangleright \mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix} = \mathbf{n} |\mathbf{a}| |\mathbf{b}| \sin \phi$$

- $|\mathbf{n}| = 1$ und \mathbf{n} steht senkrecht auf \mathbf{a} und \mathbf{b}
- Anwendungsbeispiel:
Berechnung der Oberflächennormale und Fläche eines Dreiecks
 - gegeben: $\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$
 - $\mathbf{n}' = (\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)$
 - $\mathbf{n} = \mathbf{n}' / |\mathbf{n}'|$
 - $A_{\Delta} = \frac{1}{2} |\mathbf{n}'|$



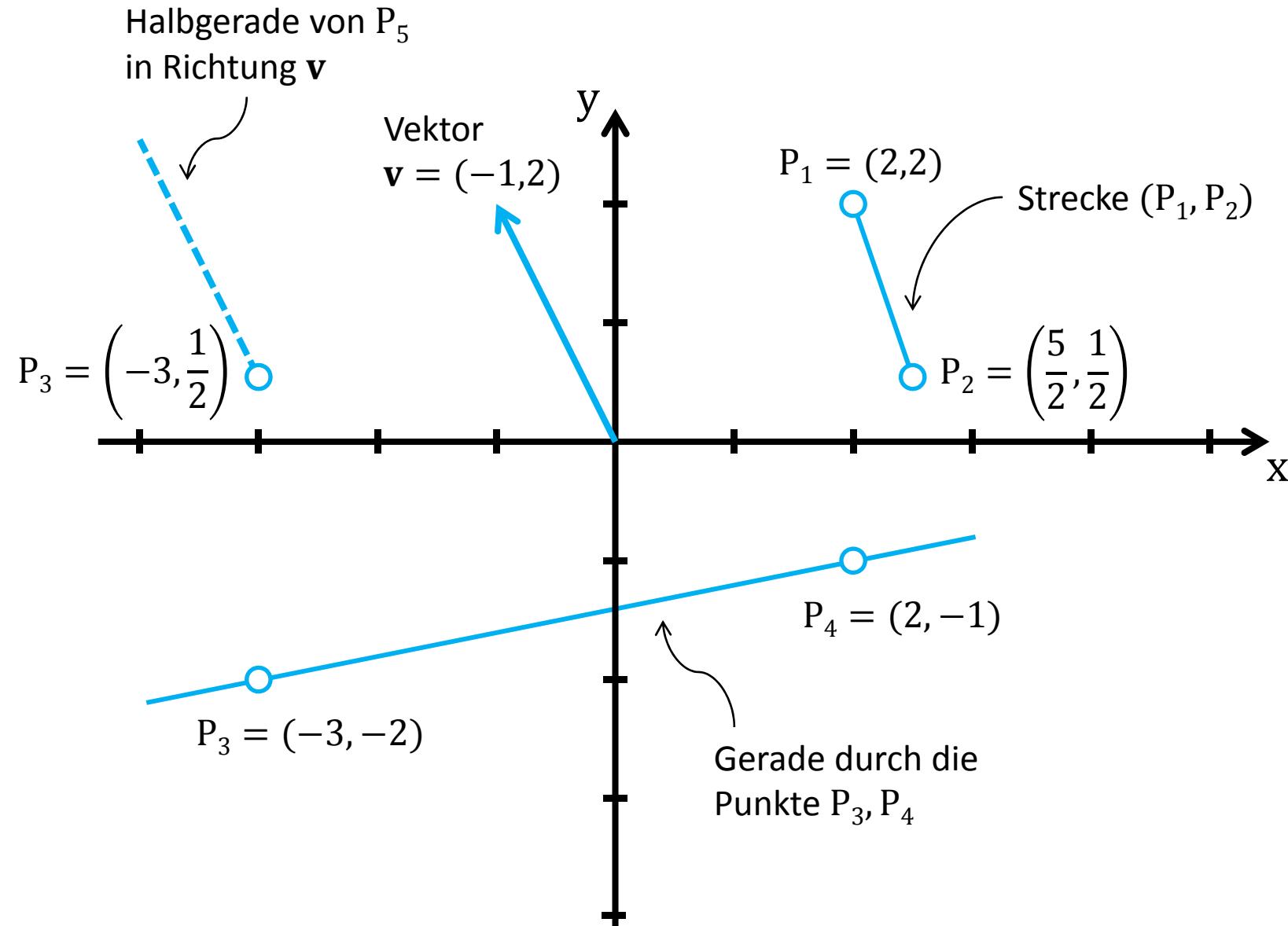
Parameterdarstellungen



- eine Gerade ist eindeutig bestimmt durch zwei Punkte P und Q ($P \neq Q$)
 - ▶ als Richtung der Gerade bezeichnet man den Vektor von P nach Q
 - ▶ Gerade
$$g(t) = P + t \cdot (Q - P), t \in \mathbb{R}$$
 - ▶ Halbgerade
$$g(t) = P + t \cdot (Q - P), t \in \mathbb{R}_0^+ \text{ (also } t \geq 0\text{)}$$
 - ▶ Strecke
$$g(t) = P + t \cdot (Q - P), t \in \mathbb{R}, 0 \leq t \leq 1$$

- eine Ebene ist eindeutig bestimmt durch drei Punkte P, Q, R, die nicht auf einer Geraden liegen (nicht kollinear)
 - ▶ die aufspannenden Vektoren der Ebene sind $(Q - R)$ und $(R - P)$
 - ▶ Ebene
$$g(s, t) = P + s \cdot (Q - P) + t \cdot (R - P), s, t \in \mathbb{R}$$
 - ▶ Parallelogramm
$$g(s, t) = P + s \cdot (Q - P) + t \cdot (R - P), s, t \in \mathbb{R} \wedge s, t \in [0; 1]$$

Geraden, Strecken, ...



Baryzentrische Koordinaten



► Definition:

► gegeben k Punkte $P_1, \dots, P_k \in \mathbb{R}^n$

► wenn ein Punkt Q in der Form

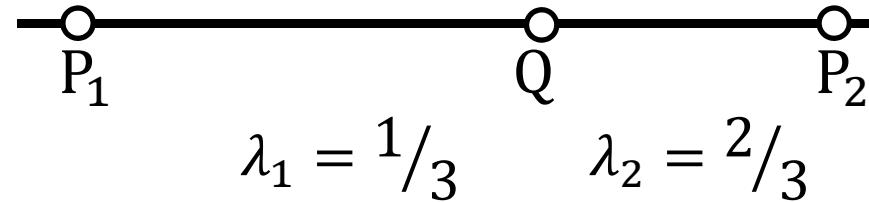
$$Q = \lambda_1 P_1 + \lambda_2 P_2 + \dots + \lambda_k P_k$$

mit $\lambda_1 + \lambda_2 + \dots + \lambda_k = 1$ (Affinkombination) dargestellt werden kann, so bezeichnet man $(\lambda_1, \lambda_2, \dots, \lambda_k)$ als die **baryzentrischen Koordinaten** von Q bzgl. der Basispunkte P_1, \dots, P_k ($k \leq n + 1$)

► Beispiel $k = 2, n = 2$

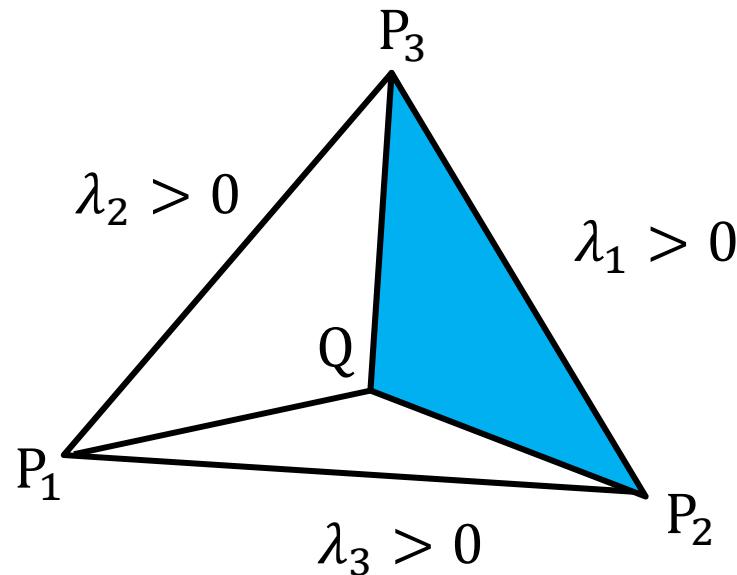
► die Punkte $Q = \lambda_1 P_1 + \lambda_2 P_2$ mit $1 = \lambda_1 + \lambda_2$ liegen auf der Geraden durch P_1 und P_2

► $Q = \lambda_1 P_1 + \lambda_2 P_2 = (1 - \lambda_2)P_1 + \lambda_2 P_2 = P_1 + \lambda_2(P_2 - P_1)$, wegen $\lambda_1 = 1 - \lambda_2$



Baryzentrische Koordinaten

- ▶ Beispiel $k = 3, n = 2$ (Dreieck im \mathbb{R}^2)
 - ▶ die Punkte Q mit $Q = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3, 1 = \lambda_1 + \lambda_2 + \lambda_3$ liegen innerhalb des Dreiecks $\Delta(P_1, P_2, P_3)$, wenn $\lambda_1, \lambda_2, \lambda_3$ positiv sind



- ▶ geometrische Interpretation der baryzentrischen Koordinaten

$$\lambda_1 = \frac{A_{\Delta}(Q, P_2, P_3)}{A_{\Delta}(P_1, P_2, P_3)}$$

$$\lambda_2 = \frac{A_{\Delta}(P_1, Q, P_3)}{A_{\Delta}(P_1, P_2, P_3)}$$

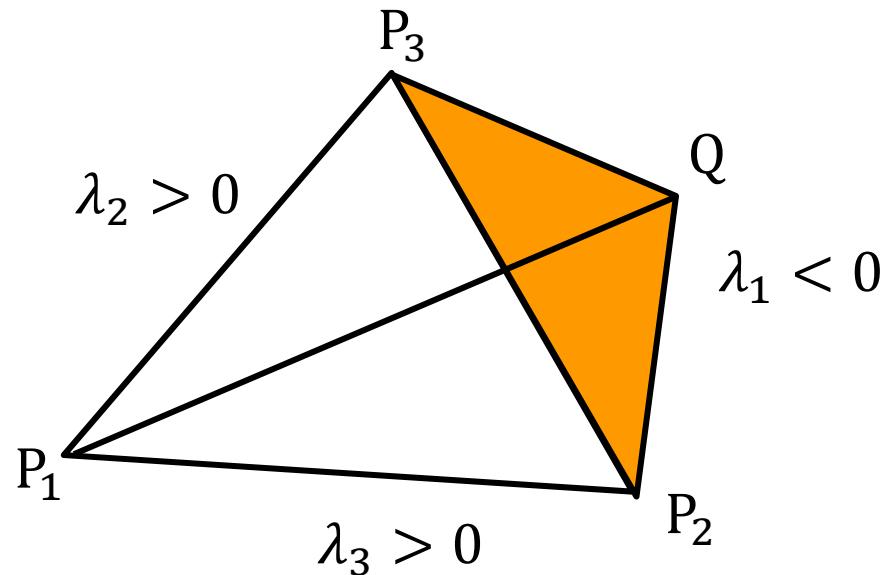
$$\lambda_3 = \frac{A_{\Delta}(P_1, P_2, Q)}{A_{\Delta}(P_1, P_2, P_3)}$$

- ▶ offensichtlich: $\lambda_1 + \lambda_2 + \lambda_3 = 1$



Baryzentrische Koordinaten

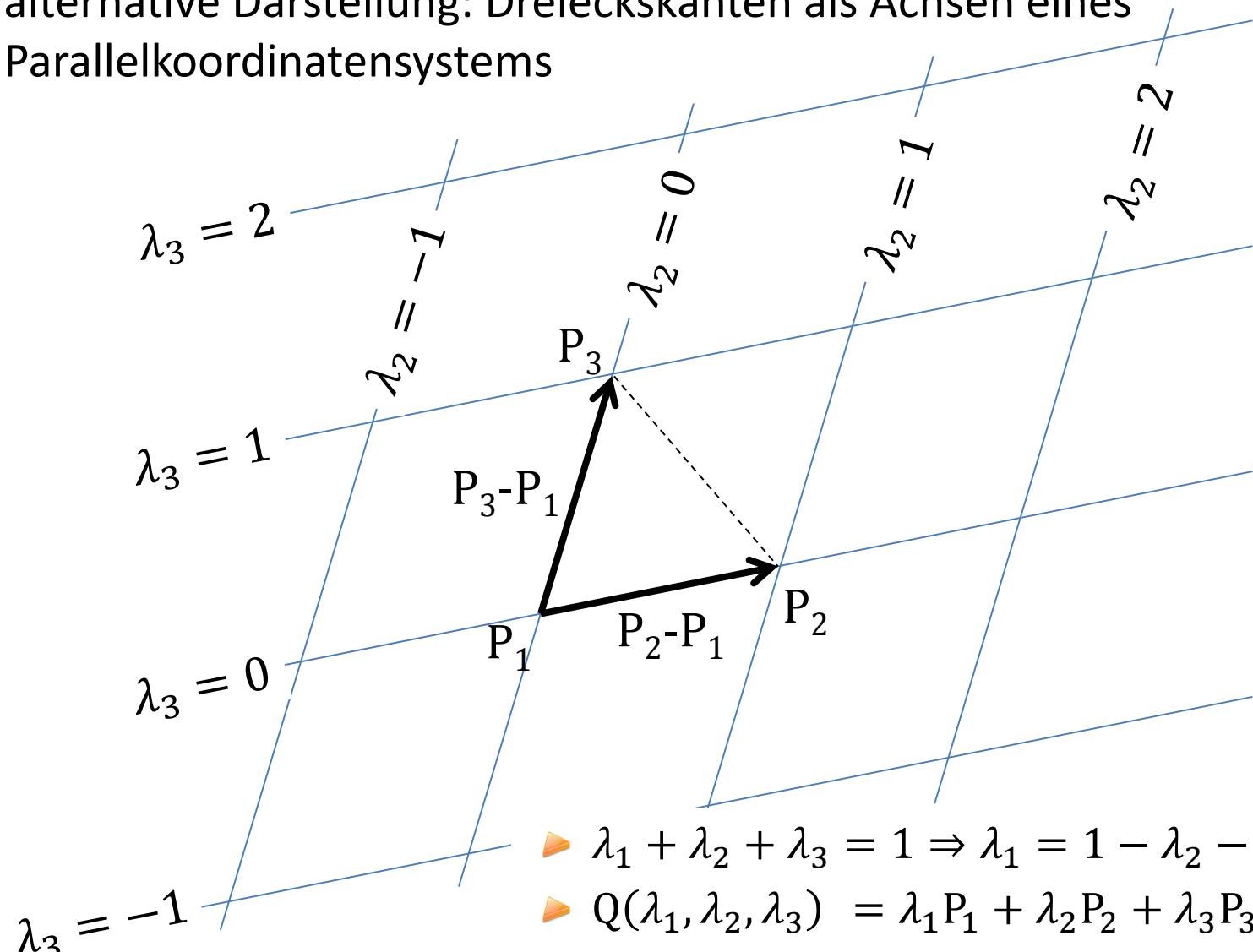
- ▶ Beispiel $k = 3, n = 2$ (Dreieck im \mathbb{R}^2)
 - ▶ die Punkte Q mit $Q = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3, 1 = \lambda_1 + \lambda_2 + \lambda_3$ liegen innerhalb des Dreiecks $\Delta(P_1, P_2, P_3)$, wenn $\lambda_1, \lambda_2, \lambda_3$ positiv sind



- ▶ geometrische Interpretation der baryzentrischen Koordinaten
$$\lambda_1 = \frac{A_{\Delta}(Q, P_2, P_3)}{A_{\Delta}(P_1, P_2, P_3)} \quad \lambda_2 = \frac{A_{\Delta}(P_1, Q, P_3)}{A_{\Delta}(P_1, P_2, P_3)} \quad \lambda_3 = \frac{A_{\Delta}(P_1, P_2, Q)}{A_{\Delta}(P_1, P_2, P_3)}$$
- ▶ $A_{\Delta}(Q, P_2, P_3) = \frac{1}{2} |(P_2 - Q) \times (P_3 - Q)| < 0$ und daher $\lambda_1 + \lambda_2 + \lambda_3 = 1$

Baryzentrische Koordinaten

- alternative Darstellung: Dreieckskanten als Achsen eines Parallelkoordinatensystems

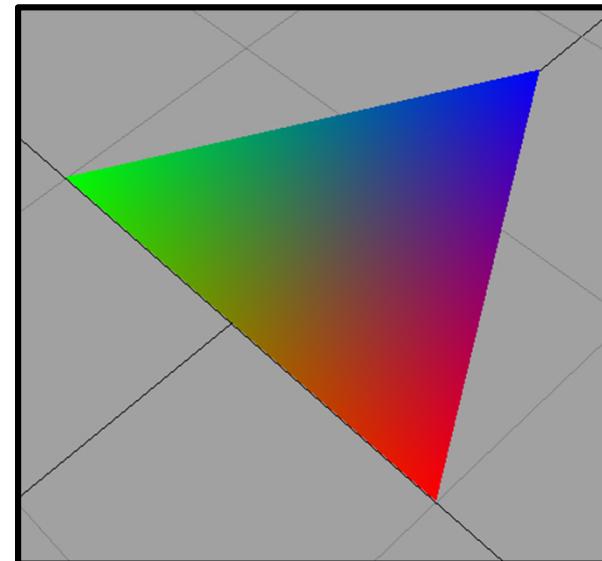


- $\lambda_1 + \lambda_2 + \lambda_3 = 1 \Rightarrow \lambda_1 = 1 - \lambda_2 - \lambda_3$
- $Q(\lambda_1, \lambda_2, \lambda_3) = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3$
- $Q(\lambda_2, \lambda_3) = (1 - \lambda_2 - \lambda_3)P_1 + \lambda_2 P_2 + \lambda_3 P_3$
 $= P_1 + \lambda_2(P_2 - P_1) + \lambda_3(P_3 - P_1)$

Anwendungen: Baryzentrische Koordinaten

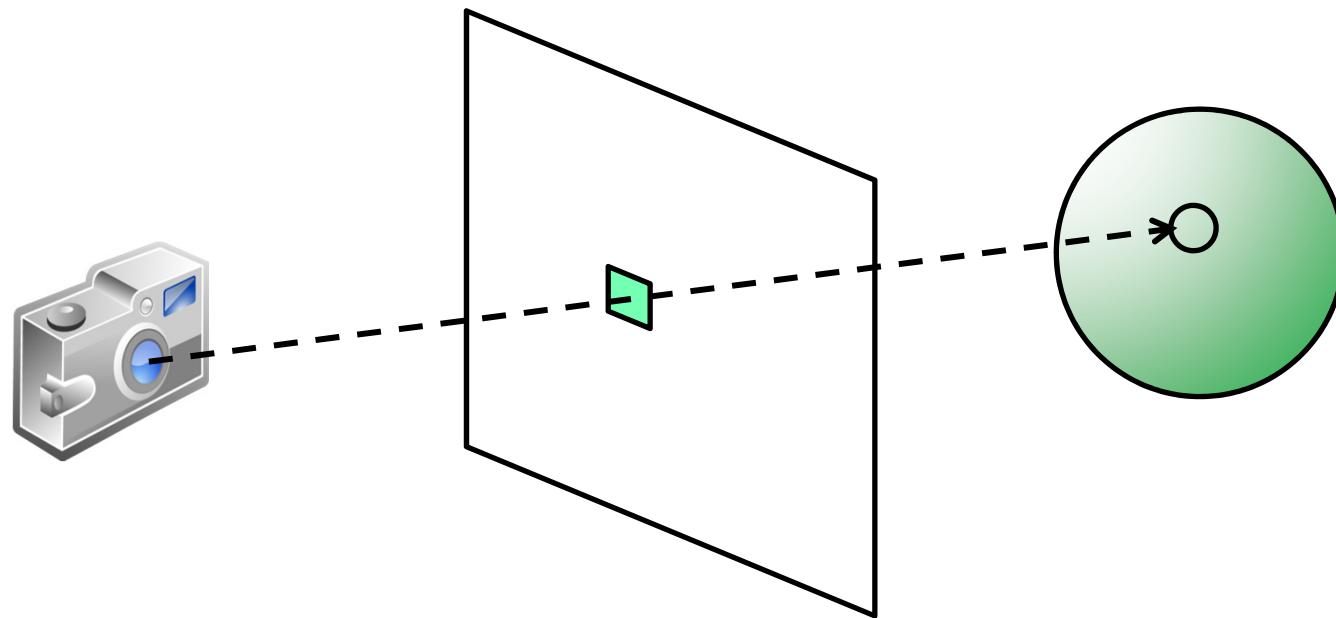


- ▶ Beispiel: teste, ob ein Punkt Q innerhalb eines Dreiecks $\Delta(P_1, P_2, P_3)$ liegt
 - ▶ Lösung: berechne $\lambda_1, \lambda_2, \lambda_3$ über Teilflächen und Dreiecksfläche
 - ▶ Punkt $Q \in \Delta(P_1, P_2, P_3)$ gdw. $\lambda_1, \lambda_2, \lambda_3 \geq 0$
- ▶ Beispiel: lineare Interpolation von (Farb-)Werten
 - ▶ gegeben: eine Farbe c_1, c_2, c_3 (als RGB-Tripel) zu jedem Eckpunkt eines Dreiecks $\Delta(P_1, P_2, P_3)$
 - ▶ gesucht: interpolierte Farbe c_Q an einem Punkt Q auf dem Dreieck
 - ▶ Lösung: berechne $\lambda_1, \lambda_2, \lambda_3$,
dann ist $c_Q = \lambda_1 c_1 + \lambda_2 c_2 + \lambda_3 c_3$
- ▶ **Interpolation mit baryzentrischen Koordinaten = lineare Interpolation**



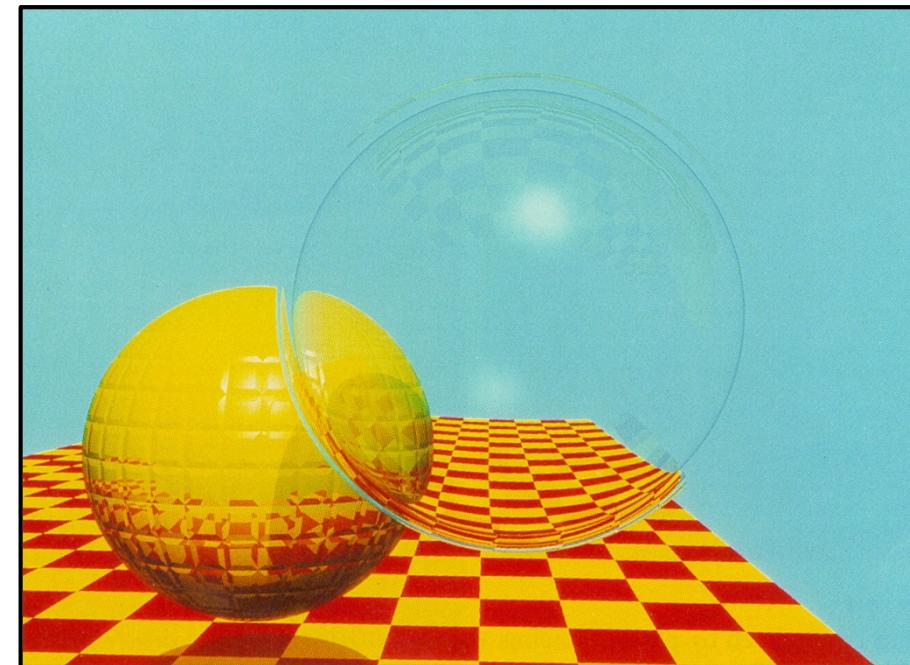
Ray Tracing: Prinzip

- ▶ betrachte einen Pixel nach dem anderen
- ▶ finde das Objekt, das die Kamera „durch“ diesen Pixel sieht
 - ▶ Sichtstrahl/Primärstrahl: eine Halbgerade vom Kamera durch den Pixel
 - ▶ gesuchtes Objekt ist das, das den Sichtstrahl schneidet und dessen Schnittpunkt am nächsten an der Kamera ist
- ▶ berechne die Farbe und Schattierung des Objektes



Ray Tracing: Prinzip

- ▶ betrachte einen Pixel nach dem anderen
- ▶ finde das Objekt, das die Kamera „durch“ diesen Pixel sieht
 - ▶ Sichtstrahl/Primärstrahl: eine Halbgerade vom Kamera durch den Pixel
 - ▶ gesuchtes Objekt ist das, das den Sichtstrahl schneidet und dessen Schnittpunkt am nächsten an der Kamera ist
- ▶ berechne die Farbe und Schattierung des Objektes
- ▶ rekursive Fortsetzung der Strahlverfolgung



[T. Whitted: An Improved Illumination Model
for Shaded Display, 1980]

Ray Tracing

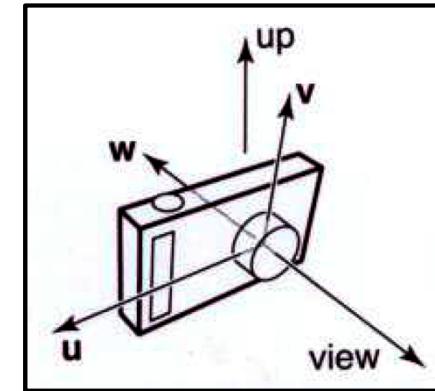


- ▶ prinzipielles Vorgehen
 - ▶ Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
 - ▶ Schnittberechnung (ray intersection, ray casting)
 - ▶ finde Objekt bzw. geometrisches Primitiv, das den Sichtstrahl schneidet und dessen Schnittpunkt am nächsten an der Kamera ist
 - ▶ Schattierung/Beleuchtungsberechnung (shading)

Erzeugung von Sichtstrahlen (ray generation)



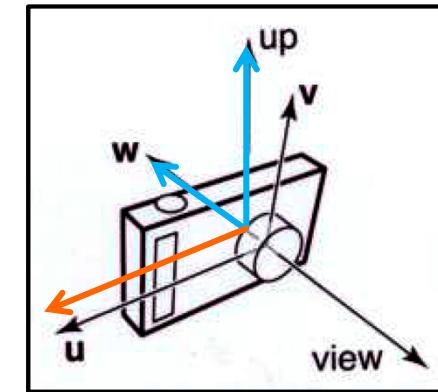
- ▶ virtuelle Kamera (Lochkamera-Modell) definiert durch
 - ▶ Position (= Projektionszentrum, „eye“) e
 - ▶ Zielpunkt z
 - ▶ „Up-Vektor“ up ($|up| = 1$)
 - ▶ negative Blickrichtung $w = (e - z)/|e - z|$
 - ▶ $\Rightarrow u = up \times w$ und $v = w \times u$
 - ▶ Normalisiere die Vektoren u, v, w



Erzeugung von Sichtstrahlen (ray generation)



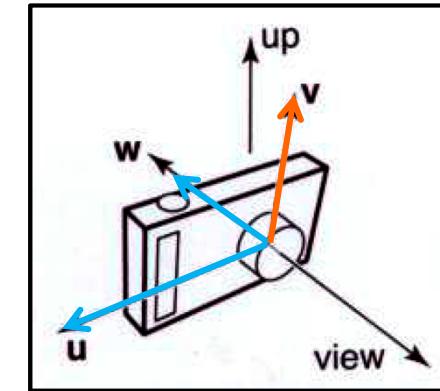
- ▶ virtuelle Kamera (Lochkamera-Modell) definiert durch
 - ▶ Position (= Projektionszentrum, „eye“) e
 - ▶ Zielpunkt z
 - ▶ „Up-Vektor“ up ($|up| = 1$)
 - ▶ negative Blickrichtung $w = (e - z)/|e - z|$
 - ▶ $\Rightarrow u = up \times w$ und $v = w \times u$
 - ▶ Normalisiere die Vektoren u, v, w



Erzeugung von Sichtstrahlen (ray generation)

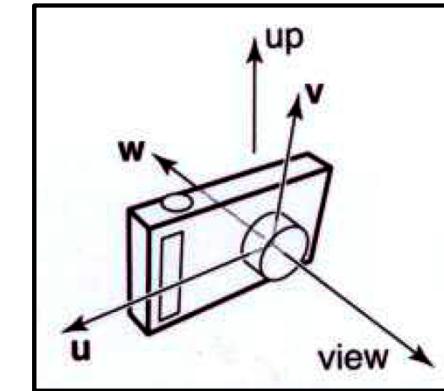


- ▶ virtuelle Kamera (Lochkamera-Modell) definiert durch
 - ▶ Position (= Projektionszentrum, „eye“) e
 - ▶ Zielpunkt z
 - ▶ „Up-Vektor“ \mathbf{up} ($|\mathbf{up}| = 1$)
 - ▶ negative Blickrichtung $\mathbf{w} = (e - z)/|e - z|$
 - ▶ $\Rightarrow \mathbf{u} = \mathbf{up} \times \mathbf{w}$ und $\mathbf{v} = \mathbf{w} \times \mathbf{u}$
 - ▶ Normalisiere die Vektoren $\mathbf{u}, \mathbf{v}, \mathbf{w}$

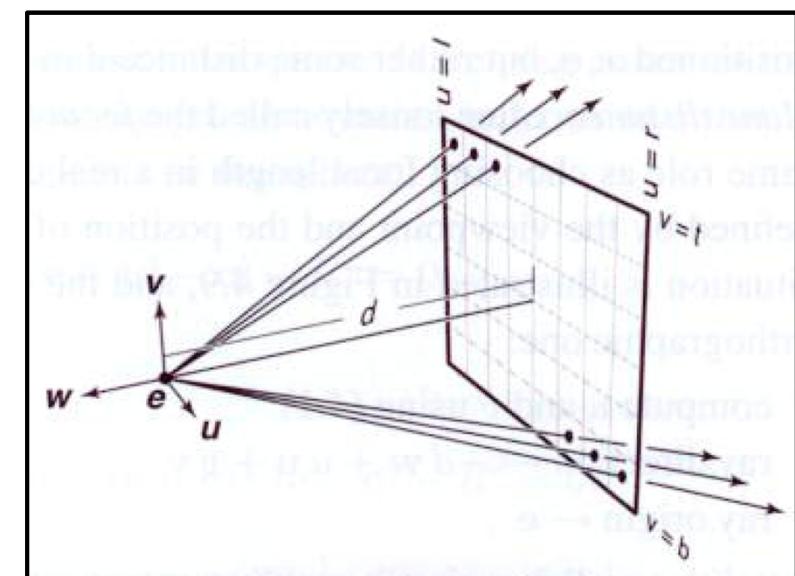


Erzeugung von Sichtstrahlen (ray generation)

- ▶ virtuelle Kamera (Lochkamera-Modell) definiert durch
 - ▶ Position (= Projektionszentrum, „eye“) e
 - ▶ Zielpunkt z
 - ▶ „Up-Vektor“ \mathbf{up} ($|\mathbf{up}| = 1$)
 - ▶ negative Blickrichtung $\mathbf{w} = (e - z)/|e - z|$
 - ▶ $\Rightarrow \mathbf{u} = \mathbf{up} \times \mathbf{w}$ und $\mathbf{v} = \mathbf{w} \times \mathbf{u}$
 - ▶ Normalisiere die Vektoren $\mathbf{u}, \mathbf{v}, \mathbf{w}$



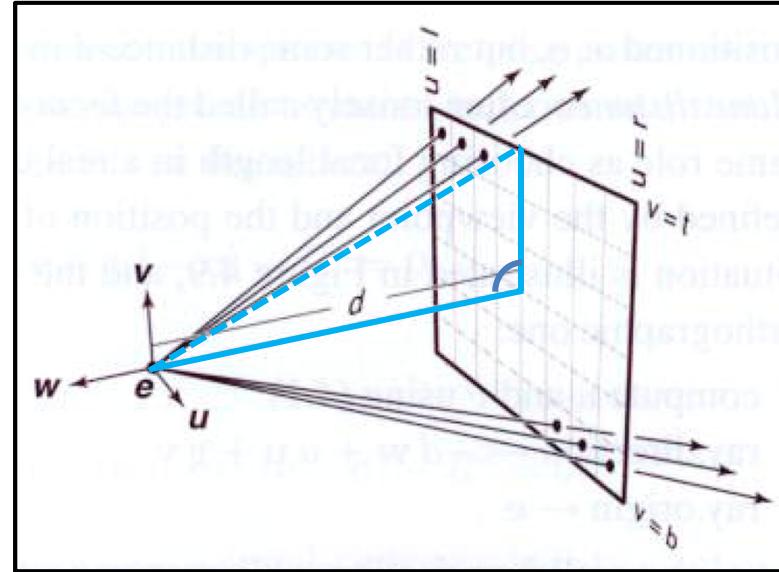
- ▶ Erzeugen von Sichtstrahlen: Zielpunkte s auf der Bildebene (von e aus gesehen)
 - ▶ Bildebene geg. durch Abstand zur Kamera: d
linker/rechter Rand: l und r
unterer/oberer Rand: b und t
 - ▶ $\mathbf{s} = u \cdot \mathbf{u} + v \cdot \mathbf{v} - d \cdot \mathbf{w}$, mit $u \in [l, r]$ und $v \in [b, t]$



Erzeugung von Sichtstrahlen (ray generation)



- ▶ Bildebene gegeben durch
 - ▶ Abstand zur Kamera: d
 - ▶ linker/rechter Rand: l und r
 - ▶ unterer/oberer Rand: b und t
- ▶ Zielpunkte s auf der Bildebene
 - ▶ $s = u \cdot \mathbf{u} + v \cdot \mathbf{v} - d \cdot \mathbf{w}$, mit $u \in [l, r]$ und $v \in [b, t]$
- ▶ Bsp. typisches Sichtfeld ist 90° und symmetrisch, d.h. $l = -r \wedge b = -t$
 - ▶ $\tan 90^\circ/2 = t/d$ (d wird festgelegt) und
 - ▶ $(r - l)/(t - b) = r/t$ „Aspect Ratio“
 - ▶ Aspect Ratio = Verhältnis Breite zu Höhe des Bildschirms

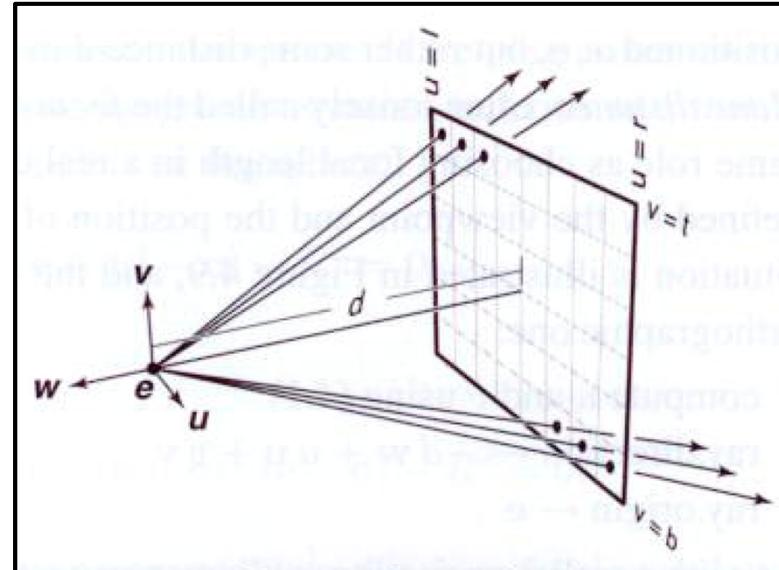


Erzeugung von Sichtstrahlen (ray generation)



- ▶ Zielpunkte s auf der Bildebene
 - ▶ $s = u \cdot \mathbf{u} + v \cdot \mathbf{v} - d \cdot \mathbf{w}$, mit $u \in [l, r]$ und $v \in [b, t]$
- ▶ Strahlgleichung $\mathbf{r}(t) = \mathbf{e} + ts$
 - ▶ $t = 0 \rightarrow \mathbf{e}$ (Kameraposition)
 - ▶ $t = 1 \rightarrow \mathbf{e} + s$ (Pixelmitte)
 - ▶ i.d.R. $\mathbf{r}(t) = \mathbf{e} + t\mathbf{d}$, mit $\mathbf{d} = s/|s|$
- ▶ Pseudocode: ein Zielpunkt pro Pixel

```
for ( y = 0; y < height; y++ ) {
    for ( x = 0; x < width; x++ ) {
        u = l + (r-l) * (x+0.5) / width;
        v = t + (b-t) * (y+0.5) / height;
        ...
    }
}
```



Ray Tracing



- ▶ prinzipielles Vorgehen
 - ▶ Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
 - ▶ Schnittberechnung (ray intersection, ray casting)
 - ▶ finde Objekt bzw. geometrisches Primitiv, das den Sichtstrahl schneidet und dessen Schnittpunkt am nächsten an der Kamera ist
 - ▶ Schattierung/Beleuchtungsberechnung (shading)

► Parameterdarstellung

alle Punkte eines geometrischen Gebildes (Linie, Ebene etc.) kann man durch Einsetzen aller zulässigen Parameterwerte gewinnen

- Bsp.: Kreis in 2D: $x = r \cdot \cos(t)$, $y = r \cdot \sin(t)$, $t \in [0..2\pi]$

► Explizite Darstellung

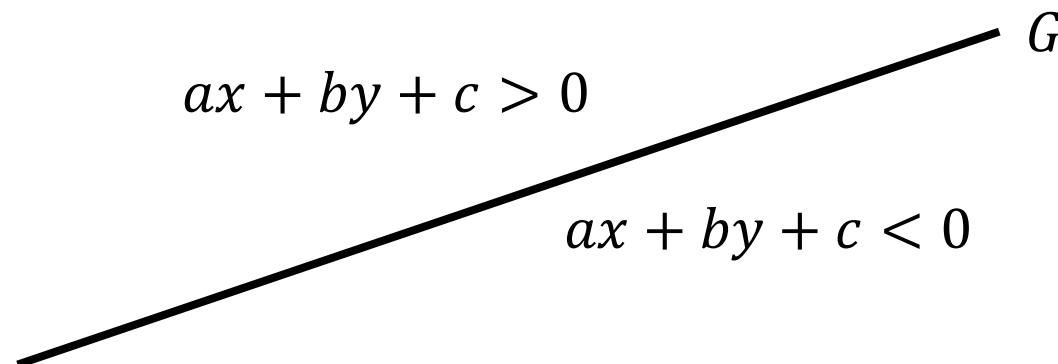
- Bsp.: Kurve in 2D: $y = f(x)$, Fläche in 3D: $z = f(x, y)$

► Implizite Darstellung

die Punkte die zu einem geometrischen Gebilde gehören bilden die Lösungsgesamtheit eines Systems von Gleichungen

- Bsp.: Gerade in 2D

$$G = \{(x, y) | ax + by + c = 0 \wedge a, b, c \in \mathbb{R}\}$$

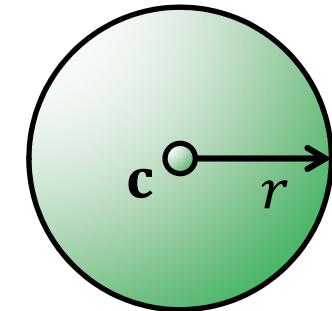


Implizite Darstellung von Kugel und Torus



- Kugel: alle Punkte auf der Kugeloberfläche haben den Abstand r vom Mittelpunkt $\mathbf{c} = (c_x, c_y, c_z)$

$$K = \{(x, y, z) \mid |(x - c_x, y - c_y, z - c_z)| = r\}$$



- Torus (nur damit Sie es mal gesehen haben):

- implizit:

$$(x^2 + y^2 + z^2 + b^2 - a^2)^2 = 4b^2(x^2 + y^2)$$

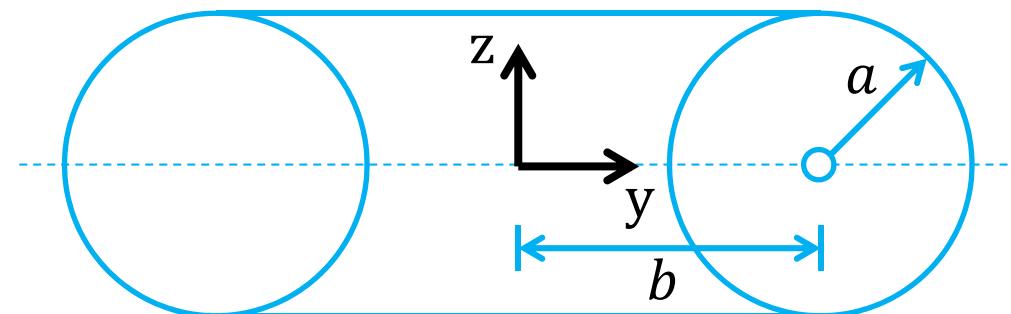
- parametrisch:

$$x = -\sin(u)(b + a \cdot \cos(v))$$

$$y = \cos(u)(b + a \cdot \cos(v))$$

$$z = a \cdot \sin(v)$$

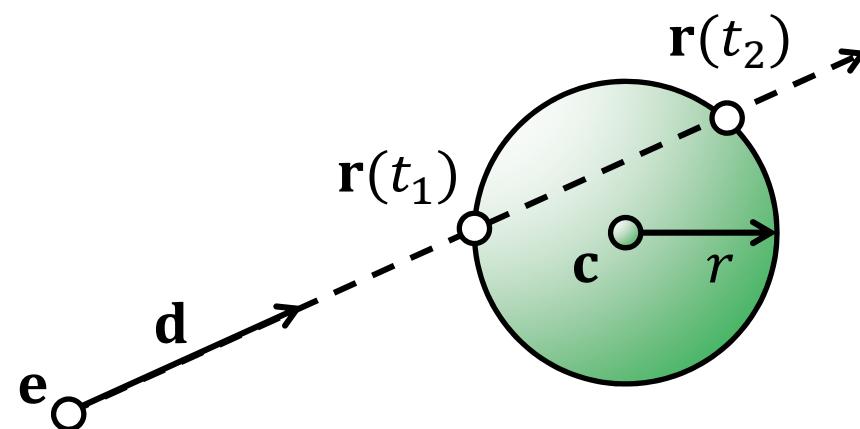
$$u, v \in [0..2\pi)$$



Schnittpunktberechnung

Strahl-Kugel-Schnitt

- ▶ Strahlgleichung (parametrisch) $\mathbf{r}(t) = \mathbf{e} + t\mathbf{d}$
- ▶ Kugel (implizite Darstellung) $|\mathbf{x} - \mathbf{c}|^2 - r^2 = 0$
- ▶ Einsetzen:
 - ▶ $|\mathbf{r}(t) - \mathbf{c}|^2 - r^2 = 0$
 - ▶ $|\mathbf{e} + t\mathbf{d} - \mathbf{c}|^2 - r^2 = 0 \quad (|\mathbf{x}|^2 = \mathbf{x} \cdot \mathbf{x})$
 - ▶ $(\mathbf{e} + t\mathbf{d} - \mathbf{c}) \cdot (\mathbf{e} + t\mathbf{d} - \mathbf{c}) - r^2 = 0$
 - ▶ $\underbrace{(\mathbf{e} - \mathbf{c}) \cdot (\mathbf{e} - \mathbf{c}) - r^2}_{const.} + \underbrace{2(t\mathbf{d} \cdot (\mathbf{e} - \mathbf{c}))}_{t(2\mathbf{d} \cdot (\mathbf{e} - \mathbf{c}))} + \underbrace{(t\mathbf{d}) \cdot (t\mathbf{d})}_{t^2(\mathbf{d} \cdot \mathbf{d})} = 0$



Schnittpunktberechnung

Strahl-Kugel-Schnitt

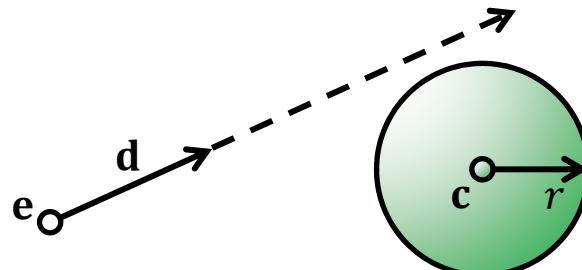
► $\underbrace{(\mathbf{e} - \mathbf{c}) \cdot (\mathbf{e} - \mathbf{c}) - r^2}_{const.} + \underbrace{2(t\mathbf{d} \cdot (\mathbf{e} - \mathbf{c}))}_{t(2\mathbf{d} \cdot (\mathbf{e} - \mathbf{c}))} + \underbrace{(t\mathbf{d}) \cdot (t\mathbf{d})}_{t^2(\mathbf{d} \cdot \mathbf{d})} = 0$

► quadratische Gleichung: Mitternachtsformel

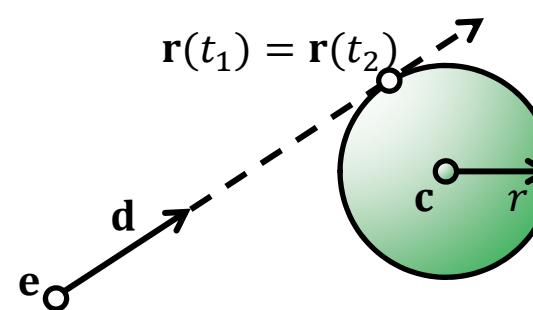
$$\triangle t_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

mit $a = \mathbf{d} \cdot \mathbf{d}$, $b = 2\mathbf{d} \cdot (\mathbf{e} - \mathbf{c})$, $c = (\mathbf{e} - \mathbf{c}) \cdot (\mathbf{e} - \mathbf{c}) - r^2$

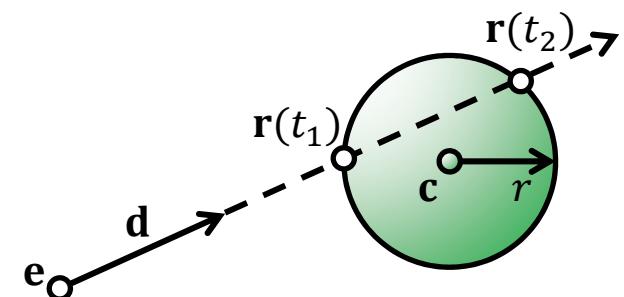
► Diskriminante $b^2 - 4ac$



Diskriminante < 0
→ keine Lösung



Diskriminante $= 0$
→ eine Lösung



Diskriminante > 0
→ zwei Lösungen

Implizite und Parameter-Darstellung einer Ebene



- ▶ eine Ebene E im \mathbb{R}^3 hat die Form

$$E = \{(x, y, z) | ax + by + cz + d = 0 \wedge a, b, c, d \in \mathbb{R}, \neg a, b, c = 0\}$$

- ▶ Parameterdarstellung \rightarrow implizite Darstellung

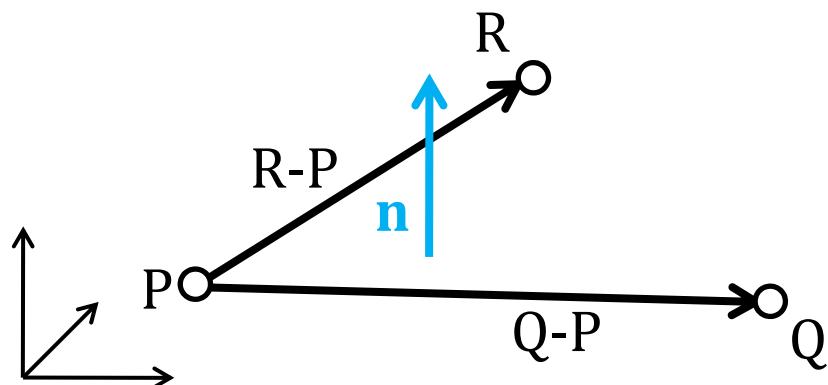
- ▶ geg. Ebene durch drei nicht-kollineare Punkte P, Q, R

- ▶ ges. Werte a, b, c, d der Ebenengleichung

- ▶ aufspannende Vektoren: $\mathbf{v}_1 = Q - P$ und $\mathbf{v}_2 = R - P$

- ▶ Normale der Ebene: $\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2$

- ▶ $a = n_x, b = n_y, c = n_z$ und $d = -\mathbf{P} \cdot \mathbf{n}$



Implizite und Parameter-Darstellung einer Ebene



- eine Ebene E im \mathbb{R}^3 hat die Form

$$E = \{(x, y, z) | ax + by + cz + d = 0 \wedge a, b, c, d \in \mathbb{R}, \neg a, b, c = 0\}$$

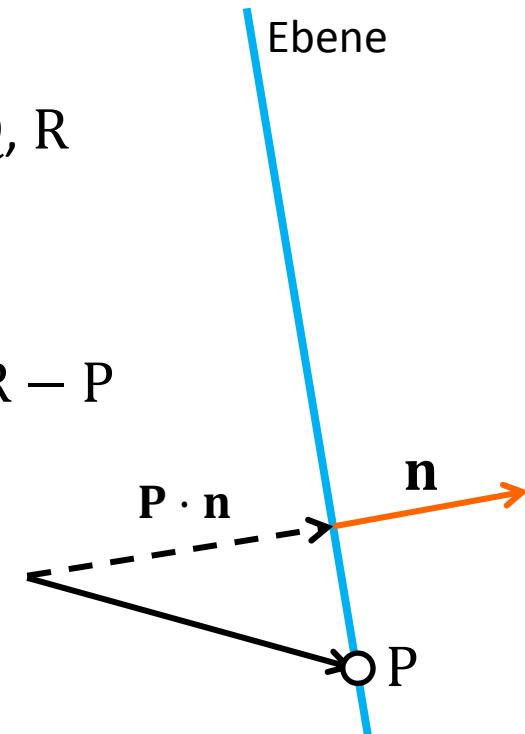
- Parameterdarstellung \rightarrow implizite Darstellung

- geg. Ebene durch drei nicht-kollineare Punkte P, Q, R
 - ges. Werte a, b, c, d der Ebenengleichung

- aufspannende Vektoren: $\mathbf{v}_1 = Q - P$ und $\mathbf{v}_2 = R - P$

- Normale der Ebene: $\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2$

- $a = n_x, b = n_y, c = n_z$ und $d = -P \cdot \mathbf{n}$

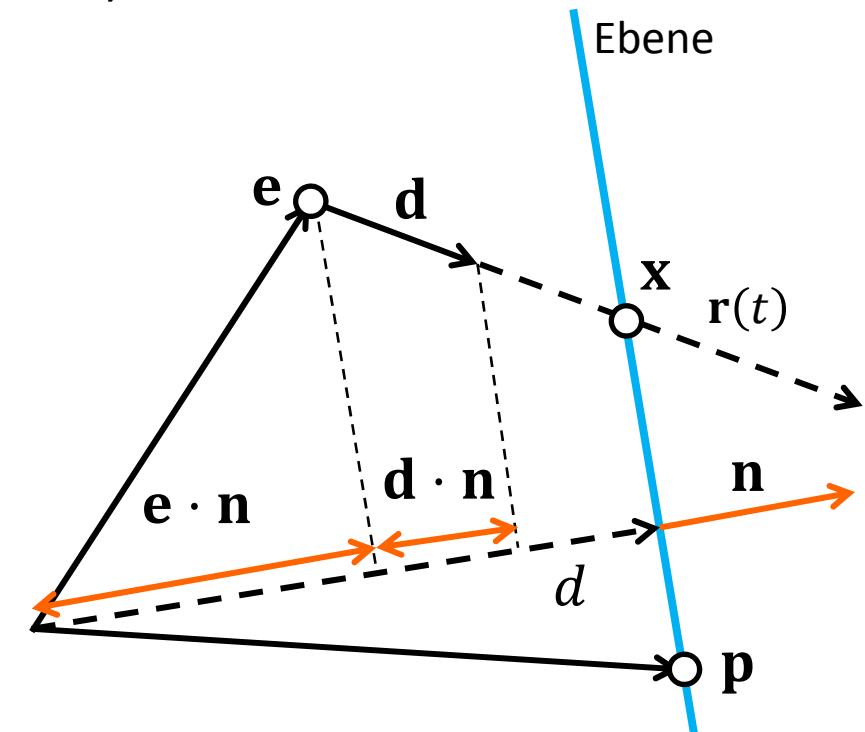


- ist $|\mathbf{n}| = 1$, dann spricht man von der Hesse-Normalform (HNF)
 - Überführung in HNF durch teilen von a, b, c, d durch $|\mathbf{n}|$

Schnittpunktsberechnung mit Ebene

Strahl-Ebene-Schnitt

- ▶ Strahl (parametrische Repr.) $\mathbf{r}(t) = \mathbf{e} + t\mathbf{d}, |\mathbf{d}| = 1$
 - ▶ Ursprung: \mathbf{e} , Richtung: \mathbf{d}
- ▶ Ebene (implizite Repr., HNF) $\mathbf{x} \cdot \mathbf{n} - d = 0, |\mathbf{n}| = 1$
 - ▶ Normalenvektor: \mathbf{n}
 - ▶ Abstand vom Ursprung: d (wg. $|\mathbf{n}| = 1$)
- ▶ Lösung durch Einsetzen
 - ▶ Einsetzen:
$$(\mathbf{e} + t\mathbf{d}) \cdot \mathbf{n} - d = 0$$
$$\mathbf{e} \cdot \mathbf{n} + t(\mathbf{d} \cdot \mathbf{n}) - d = 0$$
 - ▶ Auflösen ergibt: $t = \frac{d - \mathbf{e} \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}$
 - ▶ $\mathbf{d} \cdot \mathbf{n} = 0 \Leftrightarrow$
Strahl und Ebene sind parallel

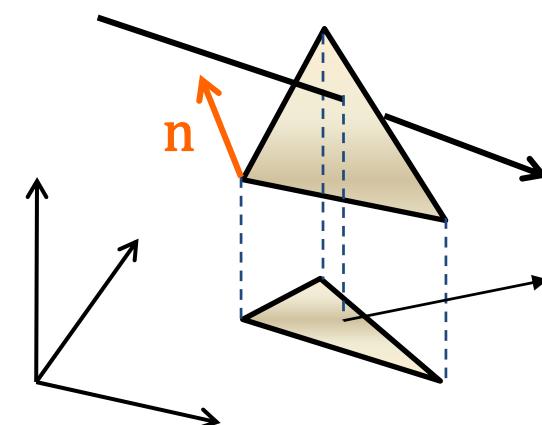
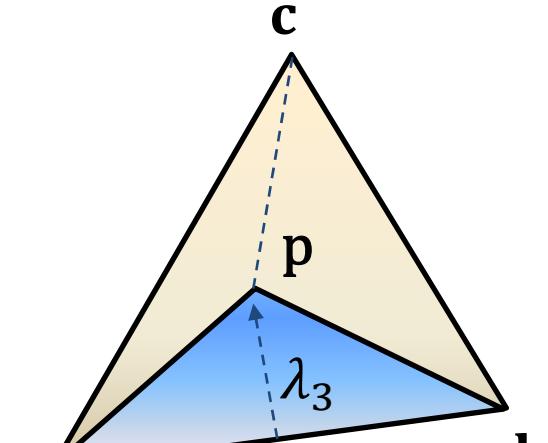


Schnittpunktberechnung mit Dreieck



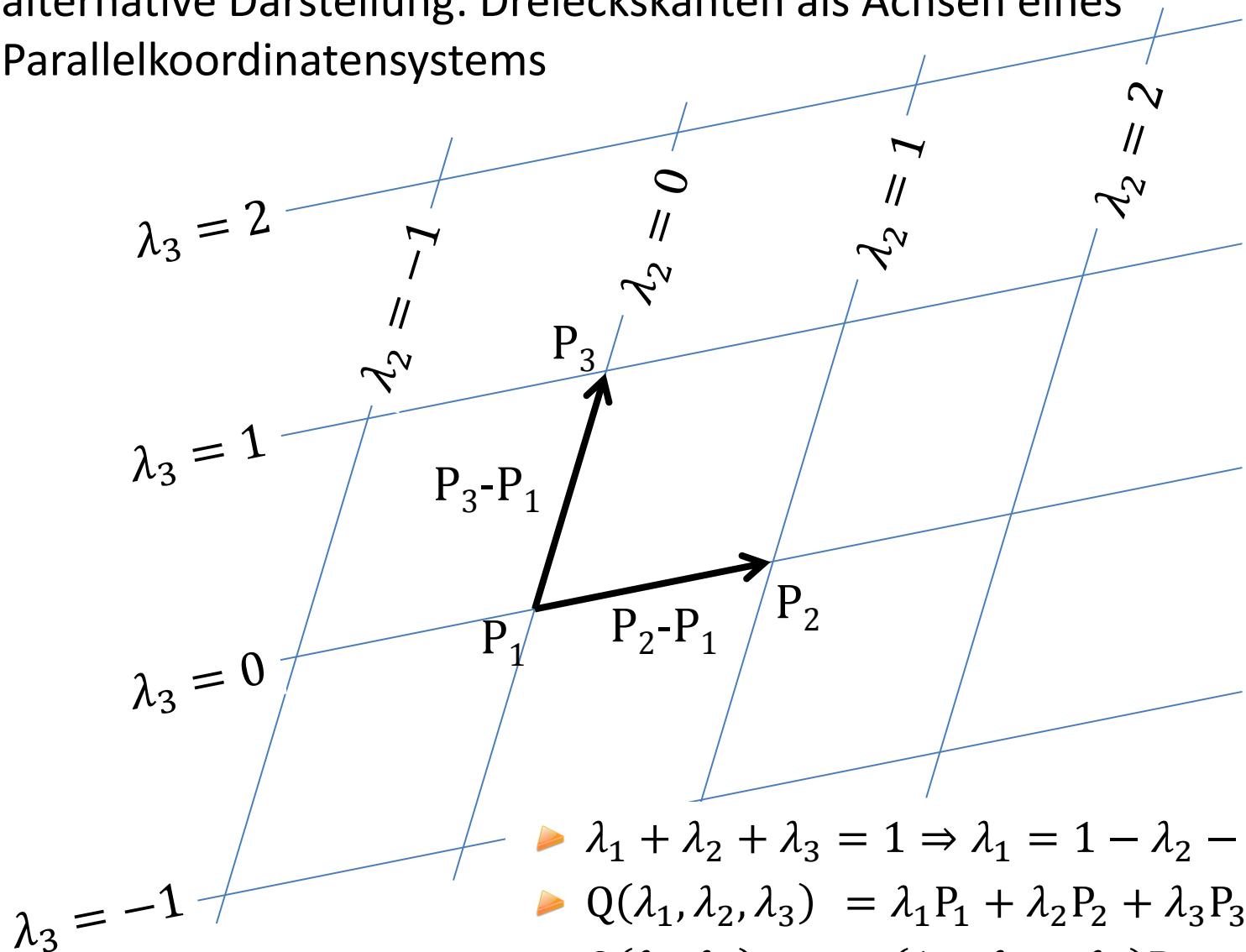
Strahl-Dreieck-Schnitt

- ▶ baryzentrische Koordinaten eines potentiellen Schnittpunkts
 - ▶ $\mathbf{p} = \lambda_1 \mathbf{a} + \lambda_2 \mathbf{b} + \lambda_3 \mathbf{c}$
mit $\lambda_1 + \lambda_2 + \lambda_3 = 1$
 - ▶ $\lambda_3 = \Delta(\mathbf{apb}) / \Delta(\mathbf{acb})$, etc.
 - ▶ Punkt in Dreieck, falls alle λ_i größer als Null
- ▶ möglicher Schnitttest:
 - ▶ Ebene durch Dreieck legen
 - ▶ berechne Schnittpunkt mit Ebene
 - ▶ berechne baryzentrische Koordinaten über 2D-Punkte in Ebene
 - ▶ Projektion in Hauptebene mit max. Normalenkoordinate
 - ▶ teste $\lambda_1, \lambda_2, \lambda_3$ auf Positivität



Zur Erinnerung: Baryzentrische Koordinaten

- alternative Darstellung: Dreieckskanten als Achsen eines Parallelkoordinatensystems



- $\lambda_1 + \lambda_2 + \lambda_3 = 1 \Rightarrow \lambda_1 = 1 - \lambda_2 - \lambda_3$
- $Q(\lambda_1, \lambda_2, \lambda_3) = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3$
- $Q(\lambda_2, \lambda_3) = (1 - \lambda_2 - \lambda_3)P_1 + \lambda_2 P_2 + \lambda_3 P_3$
 $= P_1 + \lambda_2(P_2 - P_1) + \lambda_3(P_3 - P_1)$

Schnittpunktberechnung mit Dreieck



Direkte Berechnung über baryzentrische Koordinaten

- ▶ Strahlgleichung: $\mathbf{r}(t) = \mathbf{e} + t\mathbf{d}, |\mathbf{d}| = 1$
- ▶ Gleichsetzen:
 - ▶ $\mathbf{r}(t) = Q(\lambda_2, \lambda_3)$
 - ▶ $\mathbf{e} + t\mathbf{d} = P_1 + \lambda_2(P_2 - P_1) + \lambda_3(P_3 - P_1)$
 - ▶ Schnittpunkt, wenn $\lambda_2 + \lambda_3 < 1 \wedge \lambda_2 > 0 \wedge \lambda_3 > 0$
- ▶ 3 Gleichungen, 3 Unbekannte
 - ▶ $e_x + t d_x = P_{1,x} + \lambda_2(P_{2,x} - P_{1,x}) + \lambda_3(P_{3,x} - P_{1,x}), \dots$
- ▶ Lösung mittels Cramerscher Regel (Determinanten)
 - ▶ baryzentrische Koord. müssen noch auf Positivität getestet werden
 - ▶ direkte Umsetzung im Programmcode
 - ▶ effizient, ohne Umweg über Ebenengleichung

Weitere Schnitttests



► <http://realtimerendering.com/intersections.html>

	ray	plane	sphere	cylinder	cone	triangle	AABB	OBB	frustum	polyhedron	
ray	Gems p. 304; SG; TOS; RTCD p. 198; SoftSurfer; RTK2 p. 618; RTR3 p. 781	IRT p. 80, 88; SG; GTCG p. 482; TOS; RTCD p. 179; SoftSurfer (more)	IRT p. 39, 91; Gems p. 388; Held p. 2(4); GTWeb; 3DG p. 16; GTCG p. 501; TOS; RTCD p. 127, 177; RTK2 p. 568; RTR3 p. 738	IRT p. 91; Gems IV p. 356; Held p. 2(4); GTWeb; 3DG p. 16; GTCG p. 507; TOS; RTCD p. 194	IRT p. 91; Gems IV p. 356; Held p. 2(4); GTWeb; 3DG p. 16; GTCG p. 501; TOS; RTCD p. 194	Moller-Trumbore (pt 2)(1); IRT p. 33, 102; Gems IV p. 24; Held p. 2(4); GTWeb; 3DG p. 17; Moller; GTCD p. 485; TOS; RTCD p. 133, 184; Loftson (pt 10)(2); Chirkov (pt 10)(3); Lagae (pt 10)(4); SoftSurfer; RTK2 p. 578; RTR3 p. 746; Havel TVCg June 2009	IRT p. 65, 104; Gems IV p. 395; Smits; 3DG p. 20; TOS; RTCD p. 104; Loftson (pt 10)(2); Chirkov (pt 10)(3); Lagae (pt 10)(4); SoftSurfer; RTK2 p. 572; RTR3 p. 742	(IRT p. 104; Gems II p. 247); GTWeb; Gomes; GTCG p. 630; TOS; RTCD p. 179; RTK2 p. 572; RTR3 p. 743	(IRT p. 104; Gems II p. 247); GTCG p. 493; Platti (pt 8)(4); RTCD p. 198; SoftSurfer	IRT p. 104; Gems II p. 247; GTCG p. 493; Platti (pt 8)(4); RTCD p. 198; SoftSurfer	
plane	IRT p. 80, 88; SG; GTCG p. 482; TOS; RTCD p. 179; SoftSurfer (more)	GTWeb; SG; GTCG p. 529; RTCD p. 207	distance of center to plane <= radius; GTWeb; Gomes; GTCG p. 548; TOS; RTCD p. 160, 219	GTWeb; Gomes; GTCG p. 563; RTCD p. 164	GTWeb; Gomes; GTCG p. 563; RTCD p. 164	Check if all vertices are on one side of the (pt 2)(2); GTCD p. 534; SoftSurfer	Gems IV p. 74; GTWeb p. 634; TOS; RTCD p. 161, 222; RTK2 p. 587; RTR3 p. 735	GTWeb; Gomes; GTCG p. 635; TOS; RTCD p. 161; RTK2 p. 588; RTR3 p. 737		plane	
sphere	IRT p. 39, 91; Gems p. 388; Held p. 2(4); GTWeb; 3DG p. 16; GTCG p. 501; TOS; RTCD p. 127, 177; RTK2 p. 568; RTR3 p. 738	IRT p. 39, 91; Gems IV p. 356; Held p. 2(4); GTWeb; 3DG p. 16; GTCG p. 507; TOS; RTCD p. 160, 219	distance of center to plane; Held p. 2(4); GTWeb; Gomes; GTCG p. 548; TOS; RTCD p. 88, 215, 223; RTR3 p. 763	If radii A+B >= center/center distance; Held p. 2(4); GTWeb; Gomes; GTCG p. 602; TOS; RTCD p. 114	If radii A+B >= center/axis distance; Held p. 2(4); GTWeb; Gomes; GTCG p. 602, 646; TOS; RTCD p. 114	If radii A+B >= axis/axis distance; Held p. 2(4); GTWeb; Gomes; GTCG p. 602, 646; TOS; RTCD p. 114	Held p. 2(4); GTWeb; Karaboasi (pt 4)(1); TOS; RTCD p. 133, 167, 226	Gems p. 335; Gomes; GTWeb p. 644; TOS; RTCD p. 133, 167, 226	TOS; RTCD p. 132, 166	3DG p. 462; RTCD p. 142	sphere
(capped) cylinder	IRT p. 91; Gems IV p. 227; Held p. 2(4); GTWeb; GTCG p. 507; TOS; RTCD p. 194	GTWeb; GTCG p. 551; TOS;	If radii A+B >= center/axis distance; Held p. 2(4); GTWeb; Gomes; GTCG p. 602, 646; TOS; RTCD p. 114	(GTWeb p. 602)	Held p. 2(4); GTWeb;	TOS	TOS	GTWeb; Gomes; GTCG p. 380		(capped) cylinder	
(capped) cone	IRT p. 91; Gems V p. 227; Held p. 2(4); GTWeb; GTCG p. 512	GTWeb; RTCD p. 164	GTWeb; (GTWeb p. 602)	(GTWeb p. 602)	Held p. 2(4); GTWeb;					(capped) cone	
triangle (polygon)	Moller (pt 2)(1); IRT p. 53, 102; Gems p. 24; Held p. 2(4); GTWeb; 3DG p. 17; Moller; GTCG p. 485; TOS; RTCD p. 133, 184; Loftson (pt 10)(2); Chirkov (pt 10)(3); Lagae (pt 10)(4); SoftSurfer; RTK2 p. 578; RTR3 p. 746	Check if all vertices are on one side of the (pt 2)(2); GTCD p. 534; SoftSurfer	Held p. 2(4); GTWeb; Karaboasi (pt 4)(1); TOS; RTCD p. 135, 167, 226	Held p. 2(4); GTWeb;	Held p. 2(4); GTWeb;		Gems p. 235; Gomes; GTWeb p. 393; GTCD p. 539; TOS; RTCD p. 149, 169; Shen (pt 8)(1); Guigue (pt 8)(1); Eck (pt 8)(1); RTK2 p. 590; RTR3 p. 757	Gems III p. 236; Gems V p. 375; TOS; RTCD p. 149;	GTWeb; homogeneous clipping; Gems p. 84	generalized clipping	triangle (polygon)
axis-aligned bounding box	IRT p. 65, 104; Gems p. 395; Smits; 3DG p. 20; Tardiman (optimized Woo); Schroeder; GTWeb p. 629; TOS; RTCD p. 179; RTK2 p. 572; RTR3 p. 742	Gems IV p. 74; GTWeb p. 634; TOS; RTCD p. 161, 222; RTK2 p. 587; RTR3 p. 735	Gems p. 335; Gomes; GTWeb p. 644; TOS;	RTCD p. 130, 165, 228; RTK2 p. 599; RTR3 p. 763	TOS		Gems III p. 236; Gems V p. 375; TOS; RTCD p. 149, 169; Moller; RTK2 p. 590; RTR3 p. 760	(Gems V p. 378); (Gems V p. 378); Gomes; GTWeb p. 644; TOS; RTCD p. 160, 220; RTK2 p. 602; RTR3 p. 777	(Gems V p. 378); (Gems V p. 378); Assarsson; 3DG p. 302;(1; GTWeb p. 624); RTK2 p. 612; RTR3 p. 774	Gems IV p. 74; Gems V p. 378	axis-aligned bounding box
oriented bounding box	IRT p. 104; Gems I p. 247; GTWeb; Gomes; GTCG p. 630; TOS; RTCD p. 179; RTK2 p. 572; RTR3 p. 743	GTWeb; Gomes; GTCG p. 635; TOS; RTCD p. 161; RTK2 p. 588; RTR3 p. 737	TOS; RTCD p. 132, 166	TOS	GTWeb; TOS		(Gems V p. 378); (Gems V p. 378); Gomes; GTWeb p. 639; TOS; RTCD p. 101; RTK2 p. 602; RTR3 p. 767	GTWeb; Gomes; 3DG p. 449; GTWeb p. 639;(1; TOS; RTCD p. 101; RTK2 p. 612; RTR3 p. 777	(Gems IV p. 83); (Gems IV p. 83)	oriented bounding box	
viewing frustum	(IRT p. 104; Gems II p. 247)		GTWeb; Assarsson; Gomes; GTCG p. 633; 3DG p. 302; RTK2 p. 609; RTR3 p. 774	GPG p. 380		homogeneous clipping; Gems p. 84	(Gems IV p. 83); (Gems V p. 378); (Gems V p. 378); Assarsson; 3DG p. 302;(1; GTWeb p. 624); RTK2 p. 612; RTR3 p. 771	(Gems IV p. 83); (Gems IV p. 83)	(Gems IV p. 83)	viewing frustum	
convex polyhedron	IRT p. 104; Gems II p. 247; GTCG p. 493; Platti (pt 8)(4); RTCD p. 198; SoftSurfer		3DG p. 462; RTCD p. 142		generalized clipping		Gems IV p. 74; Gems V p. 378	(Gems IV p. 83)	(Gems IV p. 83)	convex polyhedron	

Ray Tracing Pseudocode

```

▶ for ( y = 0; y < height; y++ ) {
    for ( x = 0; x < width; x++ ) {
        u = 1 + (r-1) * (x+0.5) / width;
        v = t + (b-t) * (y+0.5) / height;
        s = ...;
        d = normalize( s );
    }

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
        t' = intersect( object, e, d );
        if ( t' > 0 && t' < t ) {
            intersection = object;
            t = t';
        }
    }
    ...
}
}

```



Raytracing ist
„embarrassingly parallel“:

#pragma omp parallel for

- ▶ Achtung: uns interessieren nur Schnittpunkte vor der Kamera ($t' > 0$), und von diesen der nächste in Strahlrichtung ($t' < t$)