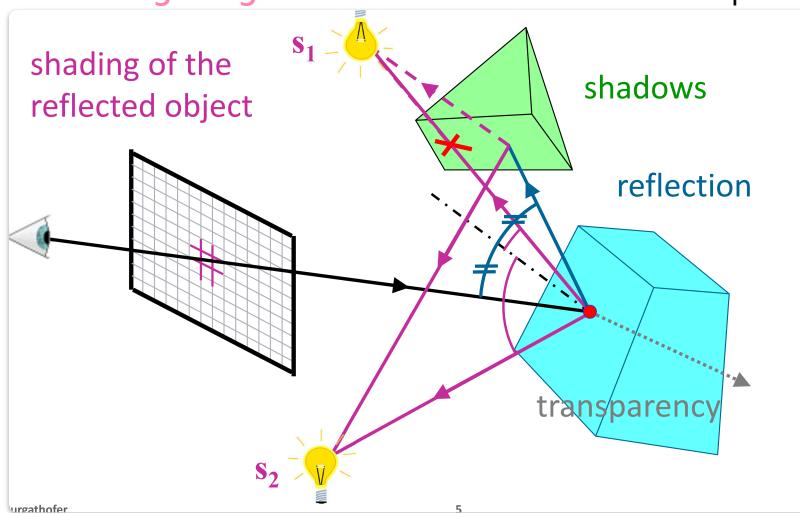


10. Ray-Tracing

- **Ray:** Strahl.
- **to trace:** Eine Spur verfolgen.
- **Ray-Tracing:** "Verfolgen von Strahlspuren".
- **Lichtstrahlen:** Durchlaufen in **verkehrter Richtung** (vom Auge zur Lichtquelle).
- **Aufbauend auf Ray-Casting:** Mächtige Methode zur Simulation wichtiger optischer Effekte:
 - Schattierung
 - Schatten
 - Spiegelbilder
 - Lichtbrechung
- **Einfachheit des Verfahrens:** Ermöglicht Darstellung komplexer Objekte:
 - Freiformflächen
 - Fraktale Oberflächen
 - Mathematische Funktionen aller Art
 - usw.

Das Ray-Tracing Prinzip

- **Basisidee:** Licht, das auf einen Bildpunkt trifft, in umgekehrter Richtung verfolgen.
- **Ziel:** Untersuchen, woher das Licht kommt.
- **Schlussfolgerung:** Daraus das Aussehen dieses Bildpunktes (Pixels) bestimmen.



Korrekte Sichtbarkeit und Schattierung

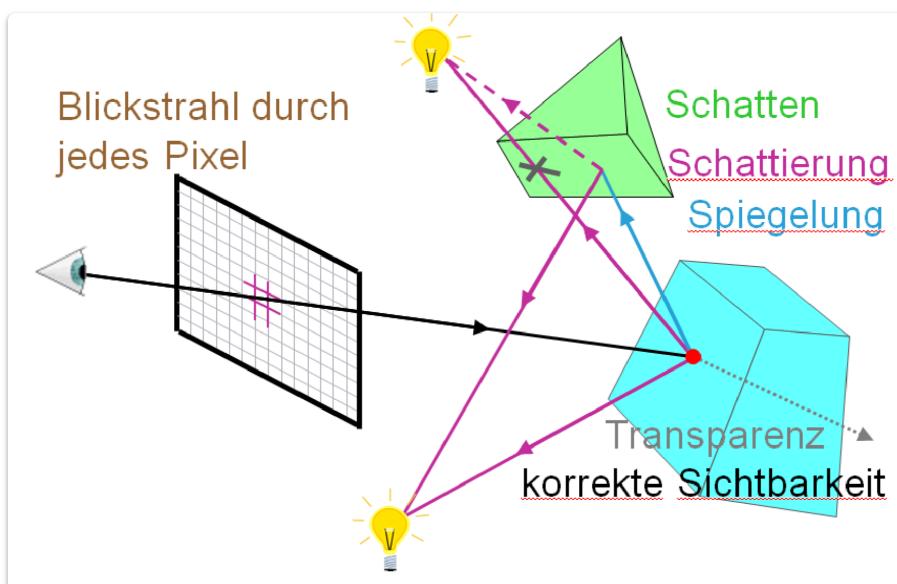
- **Blickstrahl (Primärstrahl):** Durch jeden Bildpunkt legen.
- **Schnitt:** Blickstrahl mit allen Oberflächen der Szene schneiden.
- **Nächster Schnittpunkt:** Denjenigen auswählen, der am nächsten zum Bild liegt.

- **Schattierung:** Schattierung dieses Objektpunktes (aus Blickrichtung) als Pixelwert.
- **Wiederholung:** Für alle Bildpunkte (Pixel).
- **Ergebnis:** Abbildung der Szene mit korrekter Sichtbarkeit.
- **Schattierungsmodell:** Beliebig wählbar (z.B. Phong-Modell).

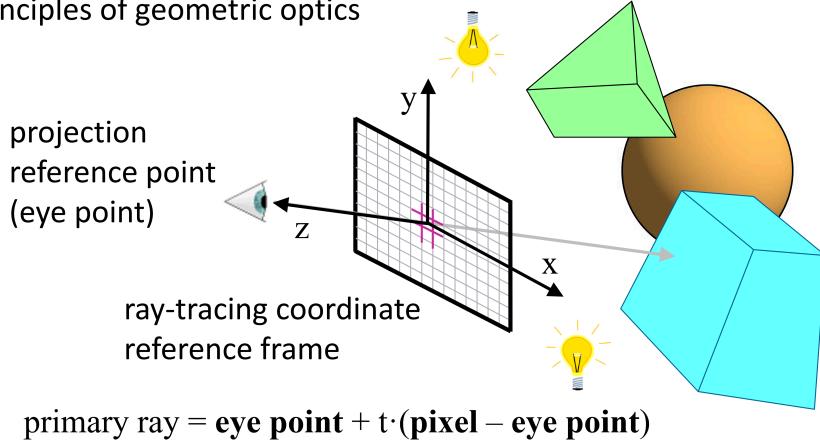


Schatten

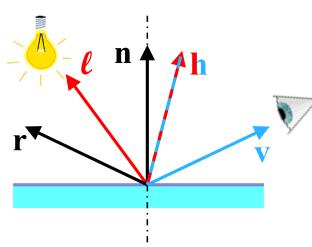
- **Schattierungsberechnung:** Benötigt Oberflächennormale und Richtungen zu allen Lichtquellen.
- **Direkter Lichteinfluss:** Nur wenn Lichteinfall nicht durch andere Objekte verdeckt ist.
- **Schattenfühler (Sekundärstrahl):** Vom zu schattierenden Punkt zur Lichtquelle legen.
- **Schnittprüfung:** Schattenfühler mit allen Objekten der Szene schneiden.
- **Schattenwurf:** Lichtquelle nicht berücksichtigen, wenn Schnittpunkt zwischen Objekt und Lichtquelle besteht.
- **Ergebnis:** Objektteile im Schatten erhalten weniger Lichteinfluss → automatischer Schattenwurf.



principles of geometric optics

ambient light $k_a I_a$ shadow ray along ℓ

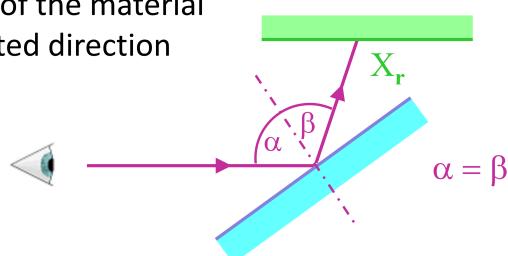
$$I_d = k_a I_a + k_d (\mathbf{n} \cdot \ell) + k_s (\mathbf{h} \cdot \mathbf{n})^p$$

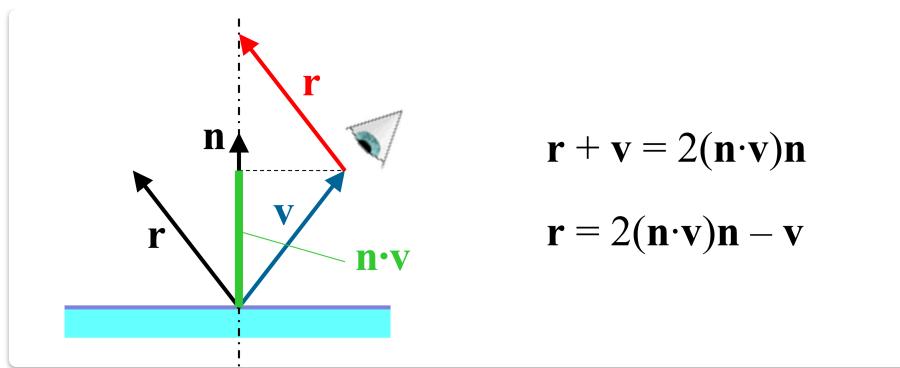
diffuse reflection $k_d (\mathbf{n} \cdot \ell)$ specular reflection $k_s (\mathbf{h} \cdot \mathbf{n})^p$ 

Spiegelbilder

- **Spiegelndes Objekt getroffen:** Nicht das Objekt selbst sichtbar, sondern das, was in Spiegelungsrichtung sichtbar ist.
- **Reflexionsgesetz:** Einfallswinkel = Ausfallwinkel (Symmetrie).
- **Reflexionsstrahl (Sekundärstrahl):** Blickstrahl an der Oberfläche spiegeln und in Spiegelungsrichtung verfolgen.
- **Schnitt:** Reflexionsstrahl mit allen Objekten schneiden.
- **Nächster Schnittpunkt:** Auswählen.
- **Farbe/Schattierung:** Schattierung dieses weiteren Auftreffpunktes (aus Richtung des Reflexionsstrahls) ist die Farbe, die der ursprüngliche Blickstrahl sieht.
- **Lokale Berechnung:** Reflexionsverhalten wird lokal berechnet → einfache Erzeugung gekrümmter Spiegel.

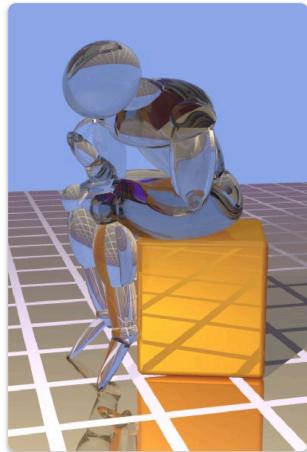
$$I_r = k_r \cdot X_r$$

 I_r ... illumination caused by reflection k_r ... reflection coefficient of the material X_r ... shading in the reflected direction



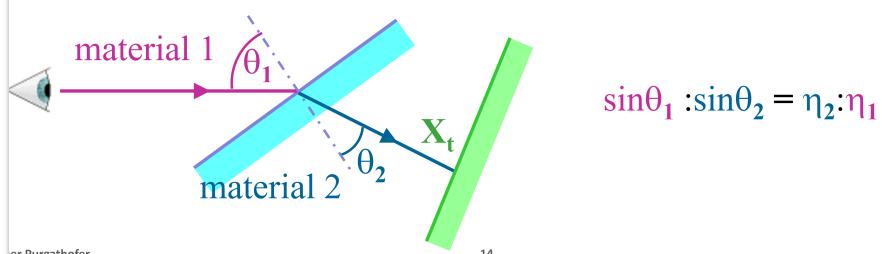
Transparenz

- **Transparentes Objekt getroffen:** Man sieht, was der durch das Objekt verlaufende Transparenzstrahl trifft.
- **Transparenzstrahl (Sekundärstrahl):** Vom Auftreffpunkt durch das transparente Objekt verfolgen.
- **Brechungsgesetz:** Richtung des Transparenzstrahls so legen, dass das Material das Licht bricht.
- **Schnitt:** Transparenzstrahl mit allen Objekten schneiden.
- **Nächster Schnittpunkt:** Auswählen.
- **Farbe/Schattierung:** Schattierung dieses weiteren Auftreffpunktes (aus Richtung des Transparenzstrahls) ist, was der ursprüngliche Blickstrahl sieht.



$$I_t = k_t \cdot X_t$$

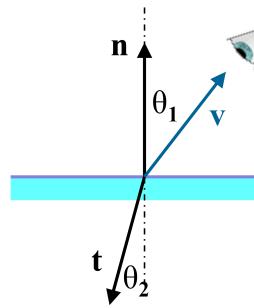
I_t ... illumination caused by transparency
 k_t ... transparency coefficient of the material
 X_t ... shading in the transparency direction



calculation of transparency ray

$$\frac{\sin \theta_2}{\sin \theta_1} = \frac{\eta_1}{\eta_2} \quad \sin \theta_2 = \frac{\eta_1}{\eta_2} \sin \theta_1$$

$$\mathbf{t} = -\frac{\eta_1}{\eta_2} \mathbf{v} - (\cos \theta_2 - \frac{\eta_1}{\eta_2} \cos \theta_1) \mathbf{n}$$

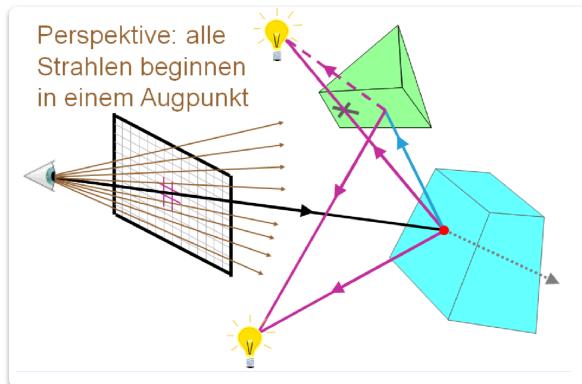


Rekursion

- **Gleichwertigkeit der Strahlen:** Jeder Strahl (außer Schattenfühler) ist gleichwertig (Primär- oder Sekundärstrahl).
- **Aktion am Auftreffpunkt:** Unabhängig davon, ob Primär- oder Sekundärstrahl.
- **Ermöglicht:**
 - Mehrfachspiegelungen.
 - Spiegelungen hinter transparentem Material.
 - Usw.

Perspektive

- **Erzeugung primärer Blickstrahlen:** Bestimmt die Abbildung der Szene auf die Bildebene.
- **Parallele Strahlen (normal zur Bildebene):** Orthogonale Parallelprojektion.
- **Strahlen von fiktivem Augpunkt:** Perspektivische Projektion (natürliche Entstehung ohne Mehraufwand).



Ray-Tracing Implementierung

Einen Ray-Tracer zu schreiben ist also ganz einfach. Man braucht eine Funktion, die eine Gerade mit allen Objekten schneidet und den vordersten Schnittpunkt zurückliefert.

Ray-Tracing Pseudocode:

```
FOR alle Pixel p0 DO
```

1. lege Blickstrahl vom Auge e aus durch p0,
schneide mit allen Objekten und wähle den nächsten Schnittpunkt p
2. FOR alle Lichtquellen s DO
schneide Schattenfühler p → s mit allen Objekten
IF kein Schnittpunkt zwischen p, s THEN Schattierung += Einfluss von s
3. IF Oberfläche von p ist spiegelnd
THEN verfolge Sekundärstrahl; Schattierung += Einfluss der Reflexion
4. IF Oberfläche von p ist transparent
THEN verfolge Sekundärstrahl; Schattierung += Einfluss der Transparenz

Viewing-Koordinatensystem und Strahldarstellung

- **Viewing-Koordinatensystem (Standard):**

- xy-Ebene = Bildebene.
- Hauptblickrichtung = negative z-Achse.

- **Strahlen (parametrisierte Form):**

$$p(t) = p_0 + t \cdot d$$

- p_0 : Startpunkt.
- t : Parameter.
- d : Richtungsvektor.

- **Primärstrahlen:**

$$e + t \cdot (p_0 - e)$$

e: Augpunkt.

p0: Pixelkoordinate.

- **Schattenfühler:**

$$p + t \cdot (s - p)$$

- p : Oberflächenpunkt.
- s : Lichtquellenposition.

- **Reflexionsstrahlen:**

$$p + t \cdot r$$

r: Reflexionsrichtung des Blickstrahls v.

$$r = (2n \cdot v)n - v$$

(aus Reflexionsgesetz).

n: Oberflächennormale.

v: Richtung des einfallenden Strahls.

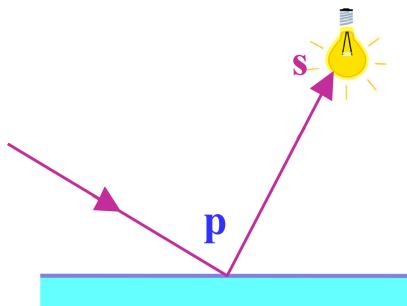
$|r| = 1$ (garantiert durch Berechnung).

ray = intersection point + $t \cdot$ vector to light source

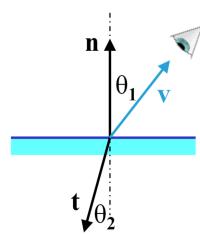
$$\text{ray} = \mathbf{p} + t \cdot (\mathbf{s} - \mathbf{p})$$

\mathbf{p} ... intersection point

\mathbf{s} ... light source position



a light source influences the result only if there is no intersection with $0 < t < 1$

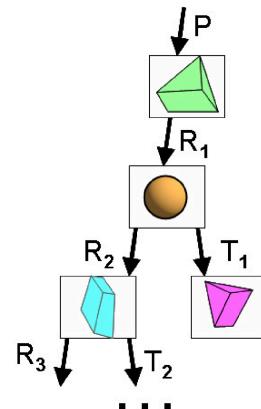
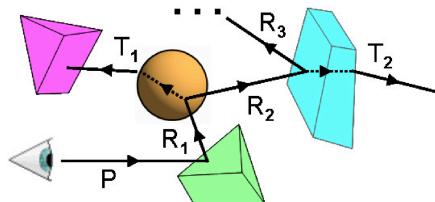


Transparenzstrahlen sind $p + t \cdot t$, wobei t sich aus dem Snellius'schen Brechungsgesetz $\sin \theta_1 : \sin \theta_2 = \eta_2 : \eta_1$ berechnet (η_i ist der Brechungsindex des Materials i):

$$t = -\frac{\eta_1}{\eta_2}v - (\cos \theta_2 - \frac{\eta_1}{\eta_2} \cos \theta_1)n$$

Auch der Vektor t hat wieder die Länge 1 (siehe linke Skizze).

Wenn man nun solcherart die Lichtstrahlen in verkehrter Richtung „verfolgt“, so entsteht eine rekursive Aufruffolge (siehe Skizze unten), die einem Strahlenbaum entspricht (rechte Skizze). Normalerweise wird dieser Baum aber nicht in dieser Form gespeichert, sondern ist nur eine symbolische Darstellung der Rekursionsaufruffolge.



Schnitte zwischen Strahlen und Objekten (Ray-Tracing)

- **Bedingungen für darzustellende Objekte:**
 - Schnittpunkt mit Gerade muss berechenbar sein.
 - Oberflächennormale am Schnittpunkt muss bekannt sein.
 - Materialeigenschaften am Schnittpunkt müssen vorhanden sein.
- **Erfüllung der Bedingungen:**
 - BReps (Boundary Representations): Einfach.
 - CSG-Bäume (Constructive Solid Geometry): Durch rekursive Evaluation.
 - Viele andere Datenformate (z.B. Freiformflächen).
- **Notwendigkeit:** Funktionen zur Schnittberechnung mit Strahl für jede Primitivart.
- **Beispiele (folgen):**
 - Kugel
 - Polygon

Schnitt Strahl-Kugel

Kugelgleichung: $|p - c|^2 - R^2 = 0$

In diese setzt man den Strahl ein: $|(e + td) - c|^2 - R^2 = 0$

Dann wird zur besseren Lesbarkeit Δp eingeführt: $\Delta p = c - e$

Und man erhält eine quadratische Gleichung in t : $t^2 - 2(d \cdot \Delta p)t + (|\Delta p|^2 - R^2) = 0$

Die 2 Lösungen entsprechen den beiden Schnittpunkten mit der Kugel:

$$t = d \cdot \Delta p \pm \sqrt{(d \cdot \Delta p)^2 - |\Delta p|^2 + R^2}$$

In Fällen, wo $R^2 \ll |\Delta p|^2$ ist (das ist durchaus häufig), entstehen in dieser Formel Rundungsfehler. Um diese zu vermeiden, kann man $d^2 = 1$ ausnutzen und die Formel umformen, so dass Rundungsfehler unwahrscheinlicher werden:

$$t = d \cdot \Delta p \pm \sqrt{|\Delta p|^2 - (d \cdot \Delta p)^2 - R^2}$$

ray equation:

$$\mathbf{p}(t) = \mathbf{p}_0 + t \cdot \mathbf{d}$$

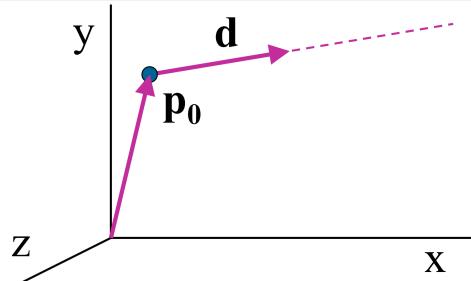
for primary rays:

$$\mathbf{d} = \frac{\mathbf{p}_0 - \mathbf{e}}{|\mathbf{p}_0 - \mathbf{e}|}$$

for secondary rays:

$$\mathbf{d} = \mathbf{r}$$

$$\mathbf{d} = \mathbf{t}$$



\mathbf{p}_0 ... initial-position vector
 \mathbf{d} ... unit direction vector
 \mathbf{e} ... eye-point vector
 \mathbf{r} ... reflection vector
 \mathbf{t} ... transparency vector

parametric ray equation
inserted into sphere equation

$$|\mathbf{p} - \mathbf{c}|^2 - R^2 = 0$$

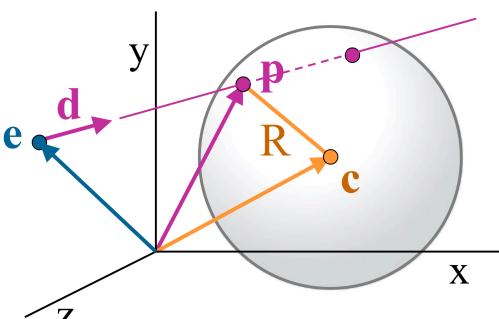
$$|(\mathbf{e} + t\mathbf{d}) - \mathbf{c}|^2 - R^2 = 0$$

$$\Delta \mathbf{p} = \mathbf{c} - \mathbf{e}$$

$$t^2 - 2(\mathbf{d} \cdot \Delta \mathbf{p})t + (|\Delta \mathbf{p}|^2 - R^2) = 0$$

$$(\mathbf{d}^2 = 1)$$

$$t = \mathbf{d} \cdot \Delta \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \Delta \mathbf{p})^2 - |\Delta \mathbf{p}|^2 + R^2}$$



if discriminant is negative \Rightarrow no intersections

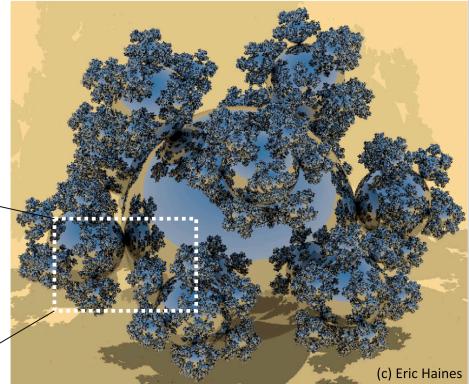
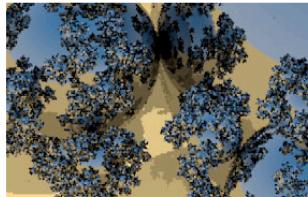
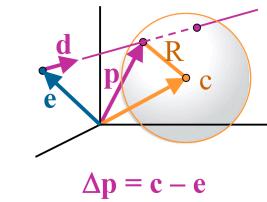
$$t = \mathbf{d} \cdot \Delta \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \Delta \mathbf{p})^2 - |\Delta \mathbf{p}|^2 + R^2}$$

$$\Rightarrow t = \mathbf{d} \cdot \Delta \mathbf{p} \pm \sqrt{R^2 - |\Delta \mathbf{p} - (\mathbf{d} \cdot \Delta \mathbf{p})\mathbf{d}|^2}$$

$$= |\Delta \mathbf{p}|^2 - 2\mathbf{d}^2|\Delta \mathbf{p}|^2 + (\mathbf{d} \cdot \Delta \mathbf{p})^2\mathbf{d}^2$$

$$\quad \quad \quad \mathbf{d}^2 = 1 \quad \quad \quad \mathbf{d}^2 = 1$$

(to avoid roundoff errors
when $R^2 \ll |\Delta \mathbf{p}|^2$)



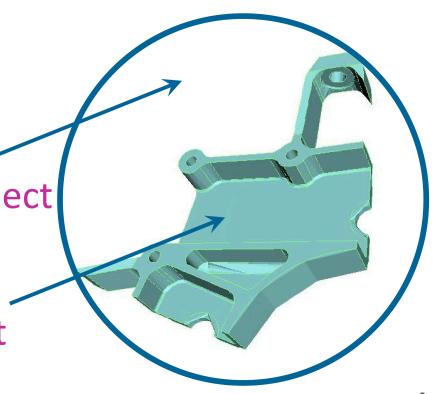
use **bounding sphere** to eliminate easy cases

ray does not hit bounding sphere
 \rightarrow no intersection with object

further investigation necessary

ray hits bounding sphere but no intersection with object

ray hits bounding sphere and intersection with object



Schnitt Strahl-Polygon

- **Ziel:** Schnittpunkt eines Strahls mit einem Polygon bestimmen.
- **Schritt 1: Backface Detection**
 - Überprüfen, ob das Polygon in die richtige Richtung schaut.
- **Schritt 2: Strahlengleichung**
 - Strahl definiert als: $\mathbf{p} = \mathbf{p}_0 + t \cdot \mathbf{d}$
 - \mathbf{p} : Punkt auf dem Strahl
 - \mathbf{p}_0 : Ursprung des Strahls
 - \mathbf{d} : Richtung des Strahls
 - t : Parameter entlang des Strahls
- **Schritt 3: Ebenengleichung des Polygons**
 - Form: $Ax + By + Cz + D = 0$
 - Kann auch in Normalenform geschrieben werden: $\mathbf{n} \cdot \mathbf{p} = -D$
 - $\mathbf{n} = (A, B, C)$: Normalenvektor der Ebene
- **Schritt 4: Schnittpunkt mit der Ebene berechnen**

- Setze die Strahlengleichung in die Ebenengleichung ein:
 - $\mathbf{n} \cdot (\mathbf{p}_0 + t \cdot \mathbf{d}) = -D$
 - $\mathbf{n} \cdot \mathbf{p}_0 + t(\mathbf{n} \cdot \mathbf{d}) = -D$
- Löse nach t auf:
 - $t(\mathbf{n} \cdot \mathbf{d}) = -(D + \mathbf{n} \cdot \mathbf{p}_0)$
 - $t = -\frac{D + \mathbf{n} \cdot \mathbf{p}_0}{\mathbf{n} \cdot \mathbf{d}}$
- **Schritt 5: Überprüfung des Schnittpunkts**
 - Liegt der Schnittpunkt innerhalb des Polygons?
 - Oder neben dem Polygon?
 - Kann durch Projektion auf eine Hauptebene in 2D überprüft werden.

1. use bounding sphere to eliminate easy cases

2. locate front faces $\mathbf{d} \cdot \mathbf{n} < 0$

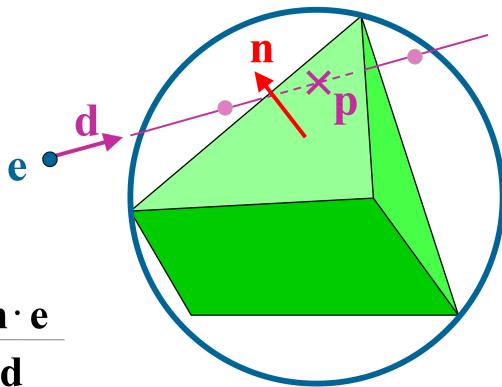
3. solve plane equation

$$Ax + By + Cz + D = 0$$

$$\mathbf{n} = (A, B, C)$$

$$\mathbf{n} \cdot \mathbf{p} = -D$$

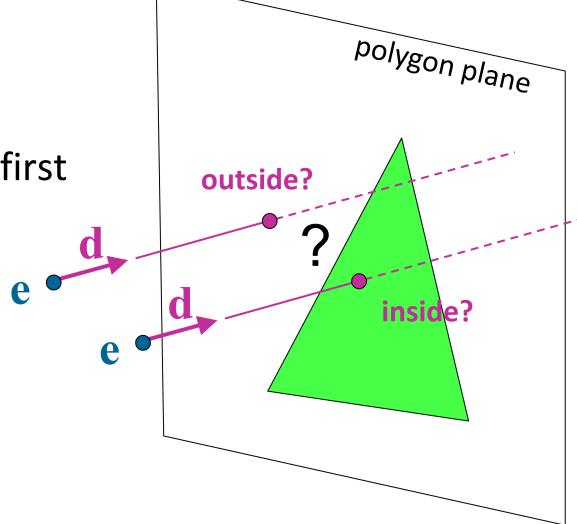
$$\mathbf{n} \cdot (\mathbf{e} + t\mathbf{d}) = -D \quad \Rightarrow \quad t = -\frac{D + \mathbf{n} \cdot \mathbf{e}}{\mathbf{n} \cdot \mathbf{d}}$$



4. intersection point inside polygon boundaries?

perform inside-outside test

→ smallest t to an inside point is first intersection point of polyhedron



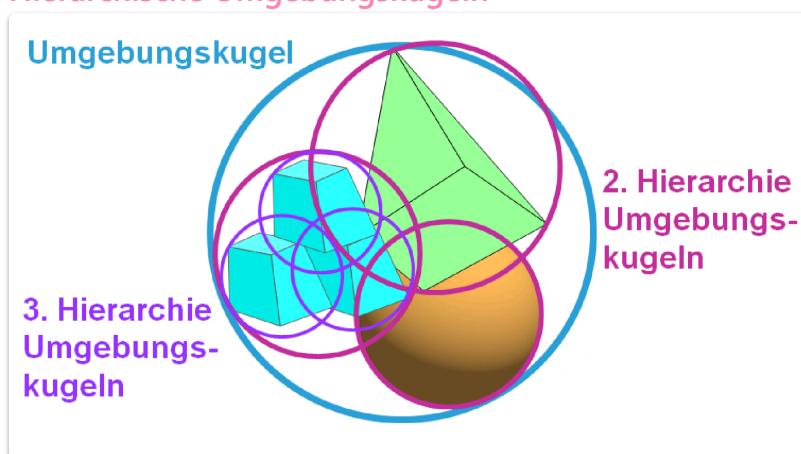
Ray-Tracing Beschleunigung

- **Problem:** Ray-Tracing ist rechenintensiv.
 - Beispiel: Szene mit 1000 Polygonen auf 1000x1000 Pixel Fläche $\Rightarrow 10^9$ Schnittberechnungen (nur für Primärstrahlen, ohne Optimierungen).
- **Notwendigkeit:** Signifikante Beschleunigung des Verfahrens ist erforderlich.

- **Wichtigste Methode:** Reduktion der Anzahl notwendiger Schnittberechnungen.
 - Ansatz: Ausnutzung von Kohärenz.

Objektumgebungen

- **Problem:** Direkter Schnitt komplexer Objekte mit Strahlen ist ineffizient.
- **Idee:** Umgebungen (Bounding Volumes) für Objekte definieren, um schnelle Vorabtests durchzuführen.
- **Umgebungskugeln (Bounding Spheres)**
 - Einfache geometrische Form (Kugel).
 - Umschließt das gesamte Objekt.
 - **Vorteil:** Schnelle Schnittprüfung mit dem Strahl.
 - **Funktionsweise:**
 - Trifft der Strahl die Umgebungskugel?
 - **Nein:** Dann trifft er auch nicht das eingeschlossene Objekt \Rightarrow keine detaillierte Schnittberechnung notwendig (Performancegewinn).
 - **Ja:** Detaillierte Schnittprüfung mit dem Objekt erforderlich.
- **Hierarchische Umgebungskugeln**

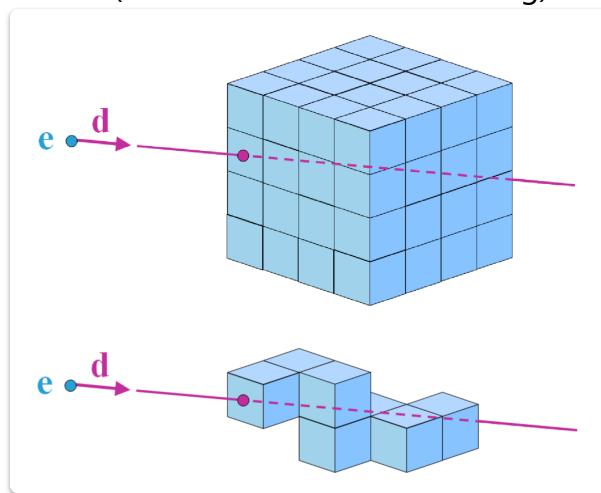


- Komplexe Objekte können hierarchisch in kleinere Teilobjekte mit eigenen Umgebungskugeln unterteilt werden.
- **Vorteil:** Verfeinerte Tests ermöglichen früheres Ausschließen von irrelevanten Teilen der Szene.
- **Prinzip:**
 1. Große Umgebungskugel für das gesamte Objekt.
 2. Unterteilung in kleinere Gruppen mit eigenen Umgebungskugeln (2. Hierarchie).
 3. Weiter Unterteilung bis zu einfachen Objekten (3. Hierarchie).
- **Komplexität:**
 - Ohne Umgebungskugeln: $O(n)$ Schnittversuche (wobei n die Anzahl der Objekte/Teile ist).
 - Mit hierarchischen Umgebungskugeln: Reduktion auf etwa $O(\log n)$.
- **Alternative Objektumgebungen:**

- Statt Umgebungskugeln können auch andere Formen verwendet werden, z.B. Umgebungsquader (Bounding Boxes).
- **Abwägung:** Mehraufwand für komplexere Formen vs. genauere Umschließung (engeres Anliegen) und potenzieller Gewinn bei der Schnittprüfung.

Raumteilungs-Methoden

- **Alternative zu Objektumgebungen:** Aufteilung des gesamten Raumes, in dem sich die Szene befindet.
 - Unabhängig von der Objektverteilung.
 - **Methoden:**
 - Regelmäßiges Raster (Array von Würfeln/Voxeln).
 - Octree (hierarchische Raumauftteilung).



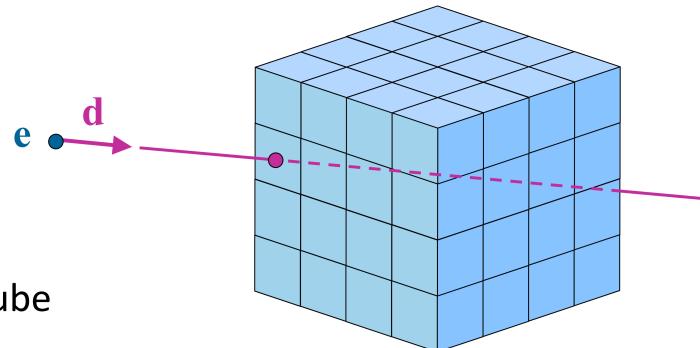
- **Vorteile:**
 - Man muss nur die Objekte in den Teilräumen betrachten, durch die der Strahl geht.
 - Schnelle Berechnung des nächsten Teilraums auf dem Strahlpfad.
 - Sobald ein Schnittpunkt innerhalb eines Teilwürfels gefunden wurde, kann die Suche beendet werden.
- **Algorithmen:** Ähnlich dem 3D-Bresenham-Verfahren zur schnellen Durchquerung der Teilräume.
- **Vorgehensweise für einzelne Teilwürfel:**
 - Strahlgleichung: $\mathbf{p}(t) = \mathbf{p}_0 + t \cdot \mathbf{d}$
 - Eintrittspunkt \mathbf{p}_{in} des Strahls in den Teilwürfel.
 - Normalenvektoren der Würfelflächen sind $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$, $(0, 0, \pm 1)$.
 - Für drei Flächen mit $\mathbf{d} \cdot \mathbf{n} > 0$ (d.h. der Strahl bewegt sich auf die Fläche zu), bestimmt man den Schnittpunkt mit dem Strahl und wählt den vordersten Schnittpunkt (kleinstes t) aus.
 - Diese Methode funktioniert auch, wenn die Würfel unterschiedlich groß sind (z.B. in einem Octree).
- **Berechnung des Austrittspunkts und des zugehörigen t -Wertes:**
 - Austrittspunkt: $\mathbf{p}_{out} = \mathbf{p}_{in} + t_k \mathbf{d}$

- Ebene der Austrittsfläche: $\mathbf{n}_k \cdot \mathbf{p} = -D_k$
- Da \mathbf{p}_{out} auf der Ebene liegt: $\mathbf{n}_k \cdot \mathbf{p}_{out} = -D_k$
- Einsetzen der Gleichung für \mathbf{p}_{out} : $\mathbf{n}_k \cdot (\mathbf{p}_{in} + t_k \mathbf{d}) = -D_k$
- Auflösen nach t_k :
 - $\mathbf{n}_k \cdot \mathbf{p}_{in} + t_k(\mathbf{n}_k \cdot \mathbf{d}) = -D_k$
 - $t_k(\mathbf{n}_k \cdot \mathbf{d}) = -D_k - \mathbf{n}_k \cdot \mathbf{p}_{in}$
 - $t_k = \frac{-D_k - \mathbf{n}_k \cdot \mathbf{p}_{in}}{\mathbf{n}_k \cdot \mathbf{d}}$

Slides:

space-subdivision methods

- regular grid
- octree

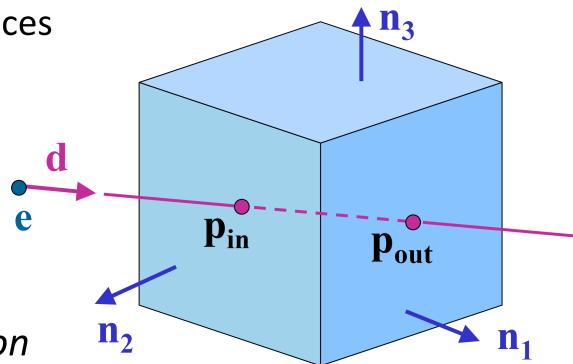


preprocess:

find object data in each cube

space-subdivision methods: incremental grid traversal

- 3D Bresenham
- processing of potential exit faces



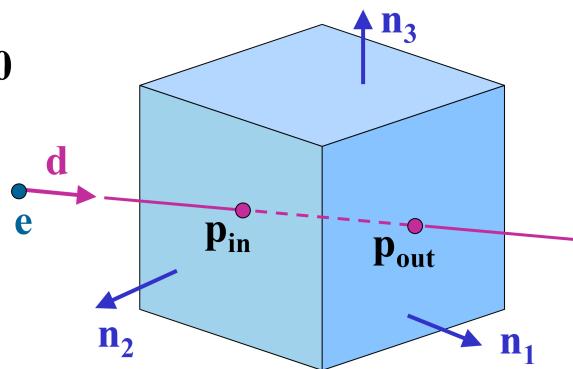
*ray traversal through a sub-region
of a cube enclosing a scene*

ray direction \mathbf{d} & ray entry position \mathbf{p}_{in}

→ potential exit faces $\mathbf{d} \cdot \mathbf{n}_k > 0$

→ normal vectors

$$\mathbf{n}_k = \begin{cases} (\pm 1, 0, 0) \\ (0, \pm 1, 0) \\ (0, 0, \pm 1) \end{cases}$$



→ check signs of components of \mathbf{d}

calculation of exit positions, select smallest t_k

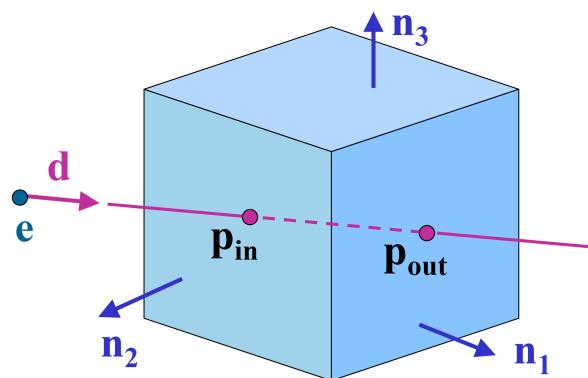
$$\mathbf{p}_{out,k} = \mathbf{p}_{in} + t_k \mathbf{d}$$

$$\mathbf{n}_k \cdot \mathbf{p}_{out,k} = -D_k$$

$$t_k = \frac{-D_k - \mathbf{n}_k \cdot \mathbf{p}_{in}}{\mathbf{n}_k \cdot \mathbf{d}}$$

example: $\mathbf{n}_k = (1, 0, 0)$

$$x_k = -D_k \Rightarrow t_k = \frac{x_k - x_0}{x_d}$$



variation: trial exit plane

perpendicular to largest component of \mathbf{d}

- (a) exit point in 0 ⇒ done
- (b) {1, 2, 3, 4} ⇒ side clear
- (c) {5, 6, 7, 8} ⇒ extra calc.

*sectors of the trial
exit plane*

