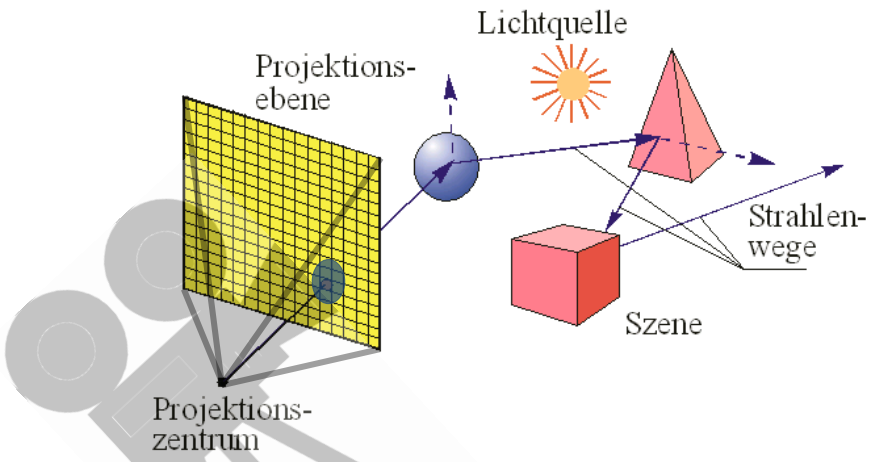
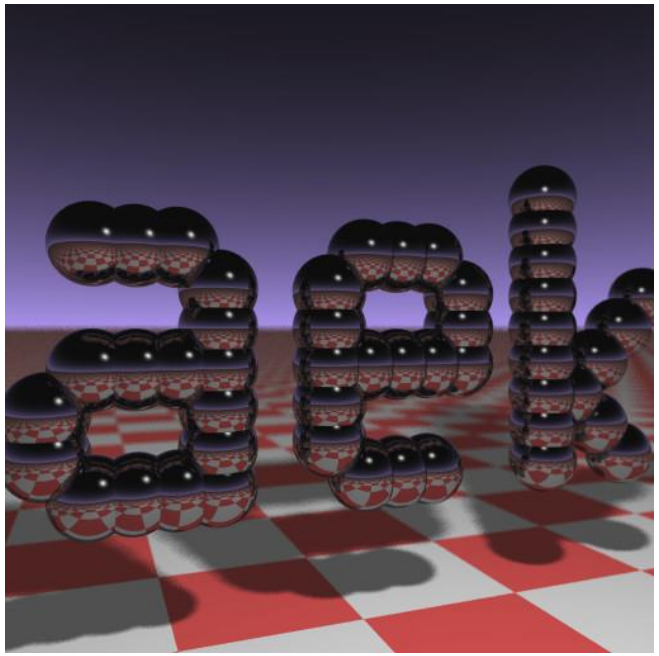
Raytracing


Lichtquelle
Projektions-
ebene
Strahlen-
wege
Szene
Projektions-
zentrum

Visitenkartenraytracer (1984)

Computergraphik und Visualisierung

```
#include <stdlib.h>   // card > aek.ppm
#include <stdio.h>
#include <math.h>
typedef int i;typedef float f;struct v{
f x,y,z;v operator+(v r){return v(x+r.x
,y+r.y,z+r.z);}v operator*(f r){return
v(x*r,y*r,z*r);}f operator%(v r){return
x*r.x+y*r.y+z*r.z;}v(){}v operator^(v r
){return v(y*r.z-z*r.y,z*r.x-x*r.z,x*r.
y-y*r.x);}v(f a,f b,f c){x=a;y=b;z=c;}v
operator!(){return*this*(1/sqrt(*this%*
this));}};i G[]={247570,280596,280600,
249748,18578,18577,231184,16,16};f R(){
return(f)rand()/RAND_MAX;}i T(v o,v d,f
&t,v&n){t=1e9;i m=0;f p=-o.z/d.z;if(.01
<p)t=p,n=v(0,0,1),m=1;for(i k=19;k--;)
for(i j=9;j--;)if(G[j]&1<<k){v p=o+v(-k
,0,-j-4);f b=p%d,c=p%p-1,q=b*b-c;if(q>0
){f s=-b-sqrt(q);if(s<t&&s>.01)t=s,n=!(
p+d*t),m=2;}}return m;}v S(v o,v d){f t
;v n;i m=T(o,d,t,n);if(!m)return v(.7,
.6,1)*pow(1-d.z,4);v h=o+d*t,l=!(v(9+R(
),9+R(),16)+h*-1),r=d+n*(n%d*-2);f b=l%
n;if(b<0||T(h,l,t,n))b=0;f p=pow(l%r*(b
>0),99);if(m&1){h=h*.2;return((i)(ceil(
h.x)+ceil(h.y))&1?v(3,1,1):v(3,3,3))*(b
*.2+.1);}return v(p,p,p)+S(h,r)*.5;}i
main(){printf("P6 512 512 255 ");v g=!v
(-6,-16,0),a=!(v(0,0,1)^g)*.002,b=!(g^a
)*.002,c=(a+b)*-256+g;for(i y=512;y--;)
for(i x=512;x--;){v p(13,13,13);for(i r
=64;r--;){v t=a*(R()-.5)*99+b*(R()-.5)*
99;p=S(v(17,16,8)+t,!(t*-1+(a*(R()+x)+b
*(y+R())+c)*16))*3.5+p;}printf("%c%c%c"
,(i)p.x,(i)p.y,(i)p.z);}}
```



http://fabiensanglard.net/rayTracing_back_of_business_card/

Inhalt

- Rekursives Raytracing
- Strahlverfolgung
  - Primärstrahlen (Supersampling)
  - Sekundärstrahlen (Epsilonproblematik)
  - Schattenstrahlen (Transparenzproblematik)
  - Distribution Raytracing
- Strahlschnitte
  - Ebene
  - Kugel
  - Achsenparallele Box
  - Dreieck
- Beschleunigungsdatenstrukturen
  - Gitter
  - KD-Baum
  - Hüllvolumen

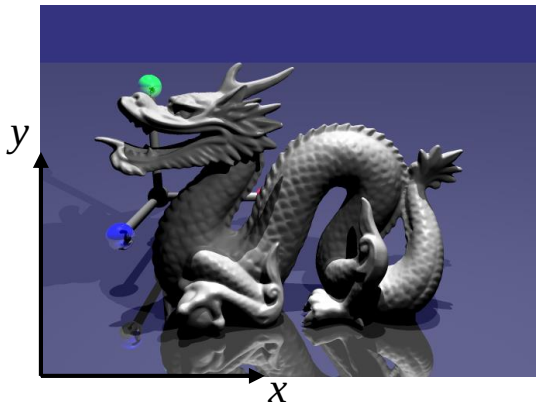
Recursive Raytracing

Computergraphik und Visualisierung

Äußere Schleife des Algorithmus:

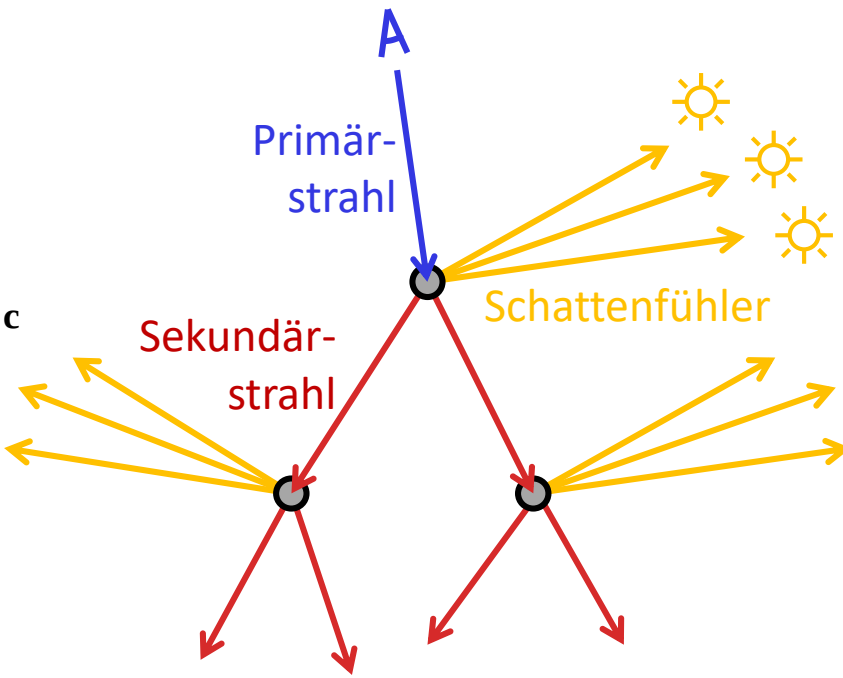
for each pixel (x,y) of camera **do**
    generate a primary ray $R = (\underline{\mathbf{p}},v)$
    set image pixel to color of incoming light:
        $I[x,y] = trace(R)$

Visibility Tracing



Rekursive Berechnung der Pixelfarbe:

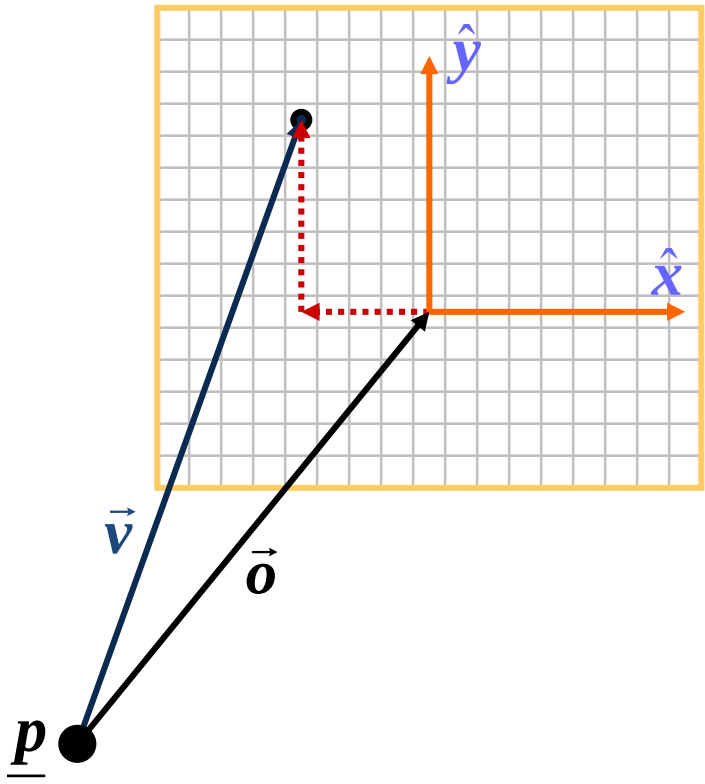
$trace(R = (\underline{\mathbf{p}},v))$
    determine first intersection $\underline{\mathbf{q}}$ of R and scene
    if no intersection exists
        return background color
    initialize resulting color $\mathbf{c}$ to black
    for each light source $\underline{\mathbf{l}}$ **do**
        if scene intersects segment $(\underline{\mathbf{q}},\underline{\mathbf{l}})$ **continue**
        add light from $\underline{\mathbf{l}}$ reflected at $\underline{\mathbf{q}}$ in direction of $-v$ to $\mathbf{c}$
    if surface at $\underline{\mathbf{q}}$ has [partial] mirror reflection
        generate reflected secondary ray R_{refl}
        contribute $trace(R_{refl})$ to $\mathbf{c}$
    if surface at $\underline{\mathbf{q}}$ has [partial] transmission
        generate transmitted secondary ray R_{trans}
        contribute $trace(R_{trans})$ to $\mathbf{c}$
    return $\mathbf{c}$



Strahlverfolgung
Primärstrahlen

- Inkrementelle Berechnung der Strahlen von Augpunkt durch Pixel der virtuellen Kamera:

```
for (x= 0; x < xres; x++)
  for (y= 0; y < yres; y++)
  {
    v = o + 2(x/xres - 0.5)·x
          + 2(y/yres - 0.5)·y;
    v = v/|v|; // Normalize
    col= trace(p, v);
    write_pixel(x,y,col);
  }
```

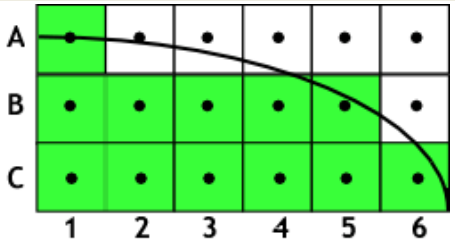


parametrisierter Strahl:

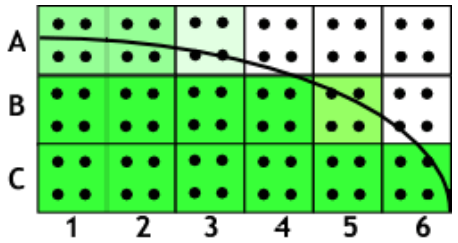
$$\underline{x}(t) = \underline{p} + t\vec{v}$$

Primärstrahlen
Supersampling

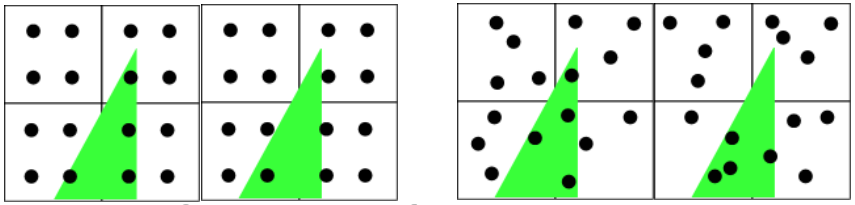
- Um Treppenstufenartefakte an Kanten und Schatten zu vermeiden, bietet sich der Einsatz von Supersampling an
- Im einfachsten Fall wird das Bild mit einer höheren Auflösung berechnet und danach die Auflösung mit einem Filter reduziert
- Meist wird aber pro Pixel ein Supersampling durchgeführt:
  - Gitter … regelmäßig
  - Zufall … ganz zufällig – neigt zur Verklumpung
  - Poisson … zufällig ohne Verklumpung
  - Jitter … zufällig pro Gitterzelle

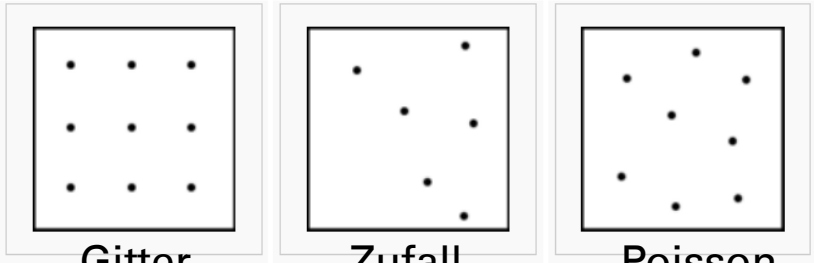


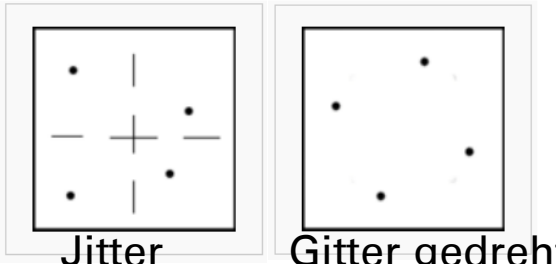
Treppenstufen



Antialiasing mit Gitter

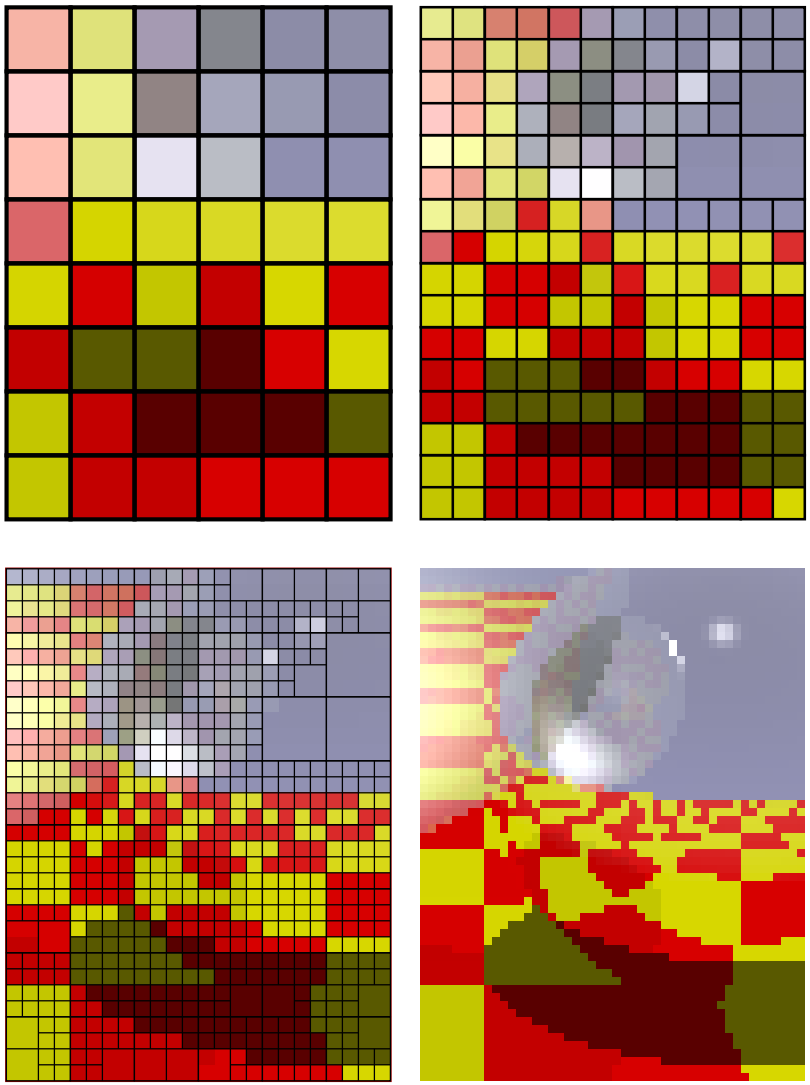

Gitter vs Zufallssampling
@www.andymeneely.com/prog/raytracer/supersampling/index.htm


Gitter Zufall Poisson


Jitter Gitter gedreht
@http://en.wikipedia.org/wiki/Supersampling

Primärstrahlen
Adaptives Sampling

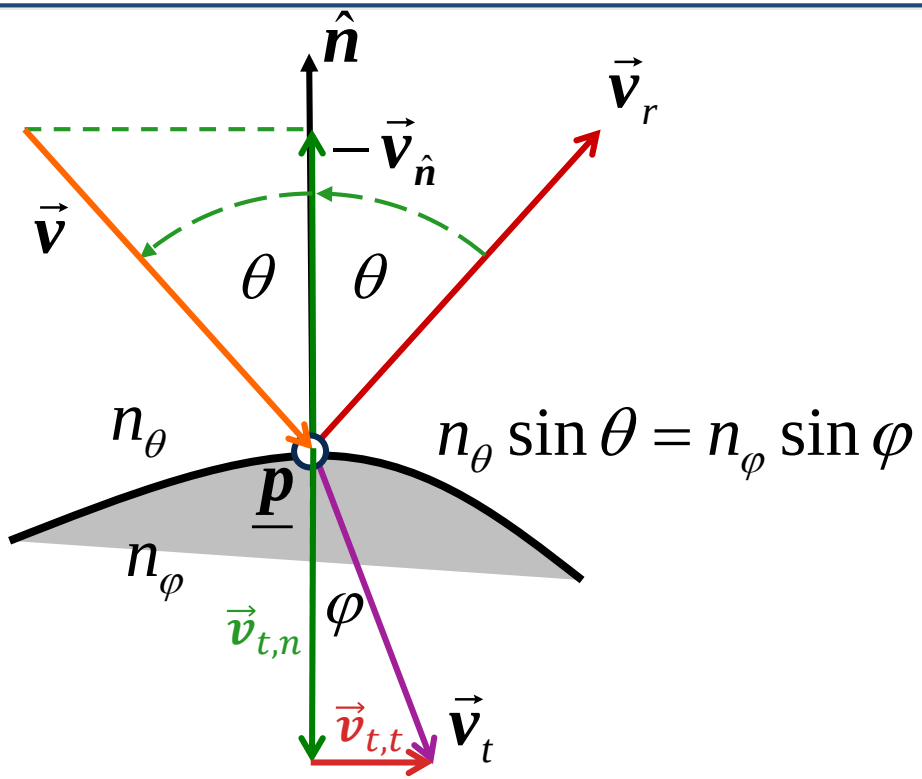
- Idee: generiere Bild in Knoten eines groben Gitters und verfeinere Zellen nur, wenn Farbinterpolation nicht ausreicht
- Bei konstanter Interpolation prüfe ob Farben in Zellecken übereinstimmen
- Bei bilinearer Interpolation trace einen Strahl durch den Zellmittelpunkt und prüfe ob die Farbe mit der interpolierten Farbe übereinstimmt.
- Unterteile rekursiv maximal bis zur Pixelauflösung
- Ist Anfangsauflösung zu grob können kleine Bildelement komplett übersehen werden



Strahlverfolgung
Sekundärstrahlen

- Trifft ein Strahl auf eine Oberfläche, so ergibt sich der Oberflächenpunkt $\underline{\boldsymbol{p}}$ aus dem Strahl-Flächenschnitt.
- Zusätzlich erhält man die Oberflächennormale $\boldsymbol{n}$
- Sekundärstrahlen werden erzeugt wenn das Material spiegelnd oder transparent ist.
- Die Sekundärstrahlen beginnen beide bei $\underline{\boldsymbol{p}}$ und haben die zu berechnenden Richtungen $\boldsymbol{v}_r$ und $\boldsymbol{v}_t$
- Zur Berechnung von $\boldsymbol{v}_t$ werden zusätzlich die Brechungsindizes benötigt. Die Herleitung der Formel basiert auf der Identität:

$$\sin^2\theta + \cos^2\theta = 1$$



Berechnung der Sekundärstrahlen

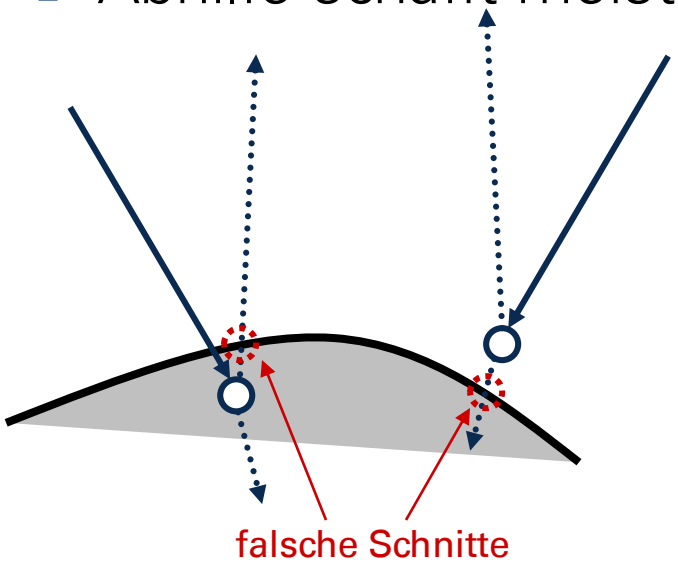
$$\vec{v}_r = \vec{v} - 2\langle \hat{n}, \vec{v} \rangle \hat{n}$$

in Klausur gegeben

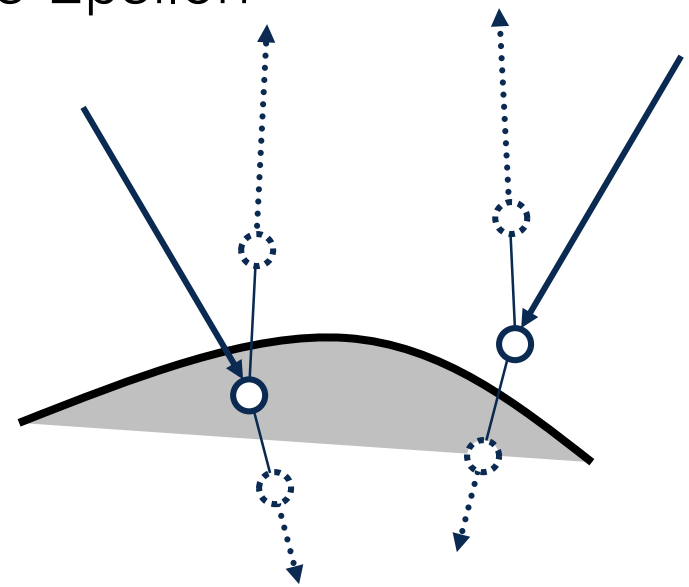
$$\vec{v}_t = \underbrace{\frac{n_\theta}{n_\phi}(\vec{v} - \langle \hat{n}, \vec{v} \rangle \hat{n})}_{\vec{v}_{t,t}} - \underbrace{\sqrt{1 - \left(\frac{n_\theta}{n_\phi}\right)^2 (1 - \langle \hat{n}, \vec{v} \rangle^2)} \cdot \hat{n}}_{\vec{v}_{t,n}}$$

Sekundärstrahlen
Numerische Ungenauigkeiten

- Beim Verfolgen von Sekundärstrahl ausgehend von Schnitten mit Oberflächen, kann es durch numerische Ungenauigkeiten zu erneutem Schnitt mit derselben Oberfläche kommen.
- Abhilfe schafft meist ein globales Epsilon



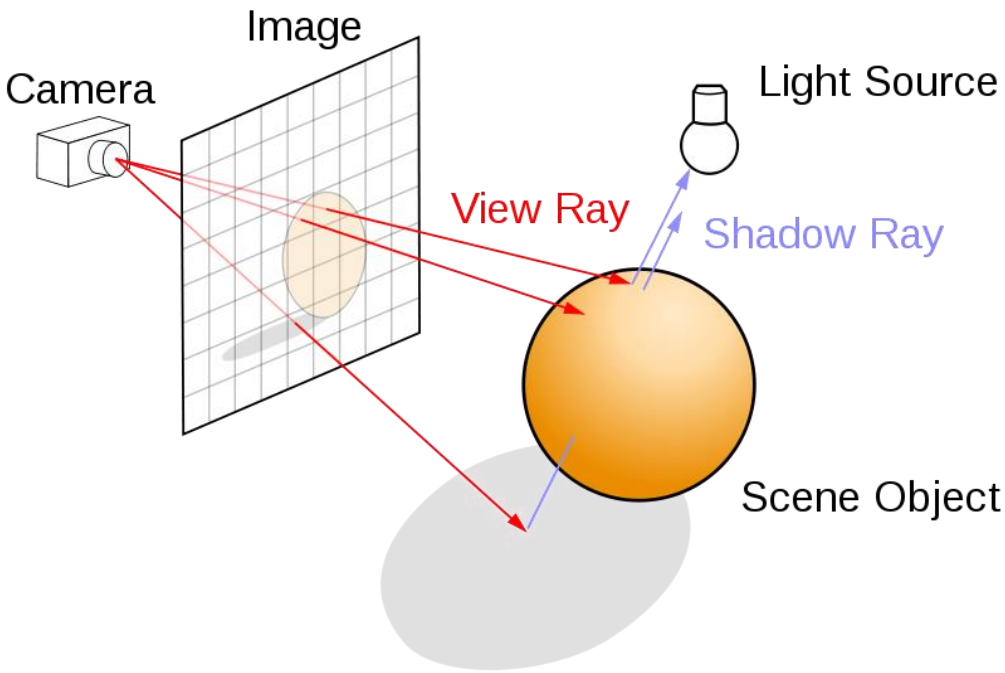
Problem: numerische Ungenauigkeiten



Abhilfe: richtungsabhängiges Offset im Sekundärstrahl

Strahlverfolgung
Schattenfühler

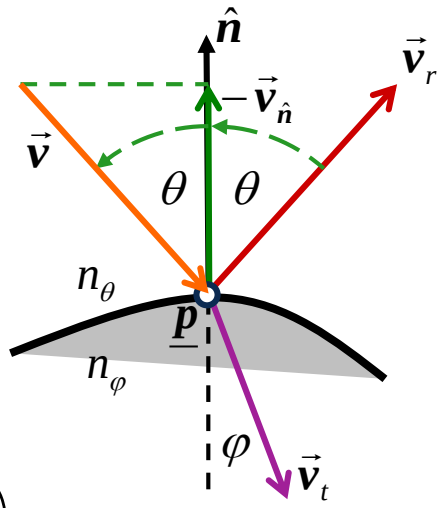
- Bei jedem Schnitt $\boldsymbol{p}$ mit einer Oberfläche wird zu jeder Lichtquelle $\boldsymbol{l}_i$ ein Schattenstrahl geschickt und geprüft, ob ein anderes Objekt zwischen aktuell betrachtetem Oberflächenpunkt und Lichtquelle ist.
- Hierzu wird nur ein beliebiger Schnitt benötigt, nicht unbedingt der erste. Deshalb werden meist zwei Schnittfunktionen implementiert:
  - `find_first_intersection` ($\boldsymbol{p}$,$\boldsymbol{v}$) … erster Schnitt vor $\boldsymbol{p}$ in Richtung $\boldsymbol{v}$
  - `is_blocked` ($\boldsymbol{p}$,$\boldsymbol{q}$) … Test, ob zwischen $\boldsymbol{p}$ & $\boldsymbol{q}$ irgendein Objekt liegt



Schattenfühler – geblockt und ungeblockt

Strahlverfolgung
Berechnung der Leuchtdichte

- Im einfachsten Fall werden die Materialparameter um einen skalaren Reflexions- , einen skalaren Refraktionskoeffizienten und einen Brechungsindex erweitert:
  - $\boldsymbol{r}_a$ … ambienter Reflektions-farbkoeffizient
  - $\boldsymbol{r}_d$ … diffuser Reflektions-farbkoeffizient
  - $\boldsymbol{r}_s$ … spekularer Reflektions-farbkoeffizient
  - s … shininess
  - r_r … Prozentsatz des ideal spiegelnden Lichts
  - r_t … Prozentsatz des refraktierten Lichts
  - n … Brechungsindex
  - n_{scene} … Brechungsindex der Szene
  - n_{ray} … Brechungsindex des Strahls



in Klausur gegeben

$$\ddot{\boldsymbol{L}}_{\text{out}}\left(\underline{\boldsymbol{p}},-\vec{\boldsymbol{v}}\right)=$$

$$\left(\ddot{\boldsymbol{L}}_{\text{scene},a}+\sum_{i=1}^{m}\ddot{\boldsymbol{L}}_{i,a}\right)\otimes\ddot{\boldsymbol{r}}_a+$$

Sekundärstrahlen

$$r_r\cdot\ddot{\boldsymbol{L}}_{\text{in}}\left(\underline{\boldsymbol{p}},\vec{\boldsymbol{v}}_r\right)+r_t\cdot\ddot{\boldsymbol{L}}_{\text{in}}\left(\underline{\boldsymbol{p}},\vec{\boldsymbol{v}}_t,n,n_{\text{ray}}\right)+$$

$$\sum_{i=1}^{m}V\left(\underline{\boldsymbol{p}},\underline{\boldsymbol{l}}_i\right)\ddot{\boldsymbol{L}}_{\text{refl},d\&s}\left(\ddot{\boldsymbol{L}}_{i,d\&s},\ddot{\boldsymbol{r}}_{d\&s},s\right)$$

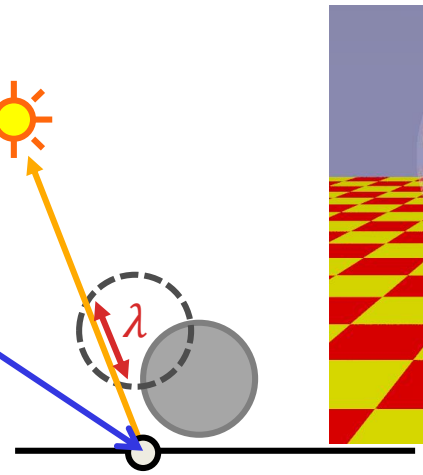
Visibilität — Lokales Beleuchtungsmodell

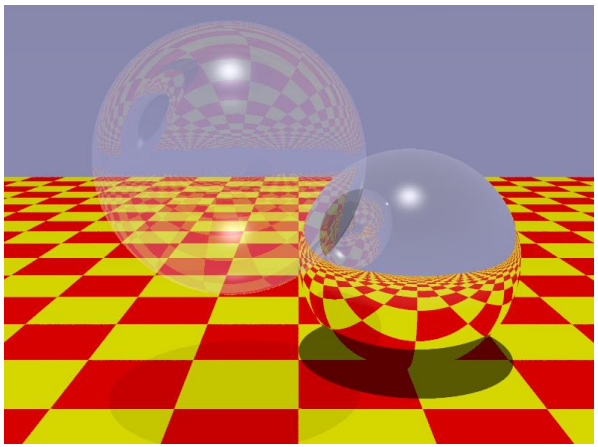
Strahlverfolgung
Schatten & Transparenz

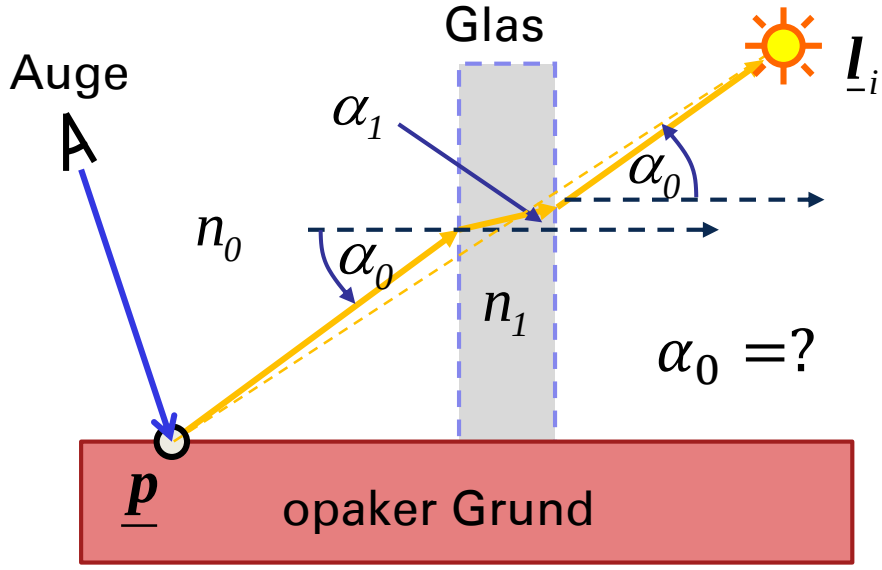
- Der Visibilitätsfaktor ist 0 oder 1 je nachdem, ob der Schattenfühler geblockt wird oder nicht
- Blockiert ein transparentes Objekt den Schattenfühler, so kann man auch einen Visibilitätsfaktor berechnen, der angibt wie viel Prozent des Lichts durch das transparente Objekt kommt.
- Für transparente Objekte definiert man die Lichtabsorptionsrate pro Länge σ
- Aus der Weglänge λ_o im Objekt o berechnet sich der Visibilitätsfaktor aus

$$V_o = r_{t,o}^2 \cdot \exp(-\lambda_o \sigma_o)$$

- Trifft Schattenfühler mehreren transp. Objekte multipliziere Visibil.faktoren.
- Das ist allerdings nur eine Approximation, weil der Schattenfühler an der Oberfläche des transparenten Objekts zwei mal gebrochen wird. Die Brechung ist zu kompliziert um berücksichtigt zu werden







Beispiel Schattenfühler mit Brechung

Strahlverfolgung
Komplexität

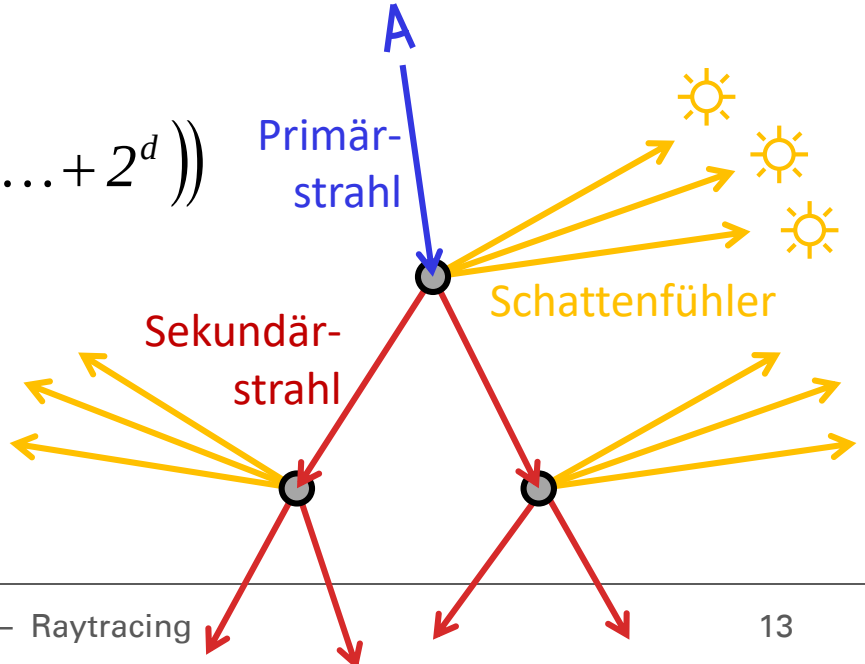
- Kenngrößen
  - w, h … Breite und Höhe des Ausgabebildes in Pixel
  - s … supersampling Faktor
  - n … Anzahl der Objekte in der Szene
  - m … Anzahl der Lichtquellen
  - d … maximale Rekursionstiefe
- worst case
  - Anzahl Strahlen:

$$\#_{\text{rays}} = \mathrm{O}\left(w \cdot h \cdot s \cdot (1+m) \cdot \left(1+2+4+\ldots+2^{d}\right)\right)$$

$$= \mathrm{O}\left(w \cdot h \cdot s \cdot (1+m) \cdot 2^{d+1}\right)$$

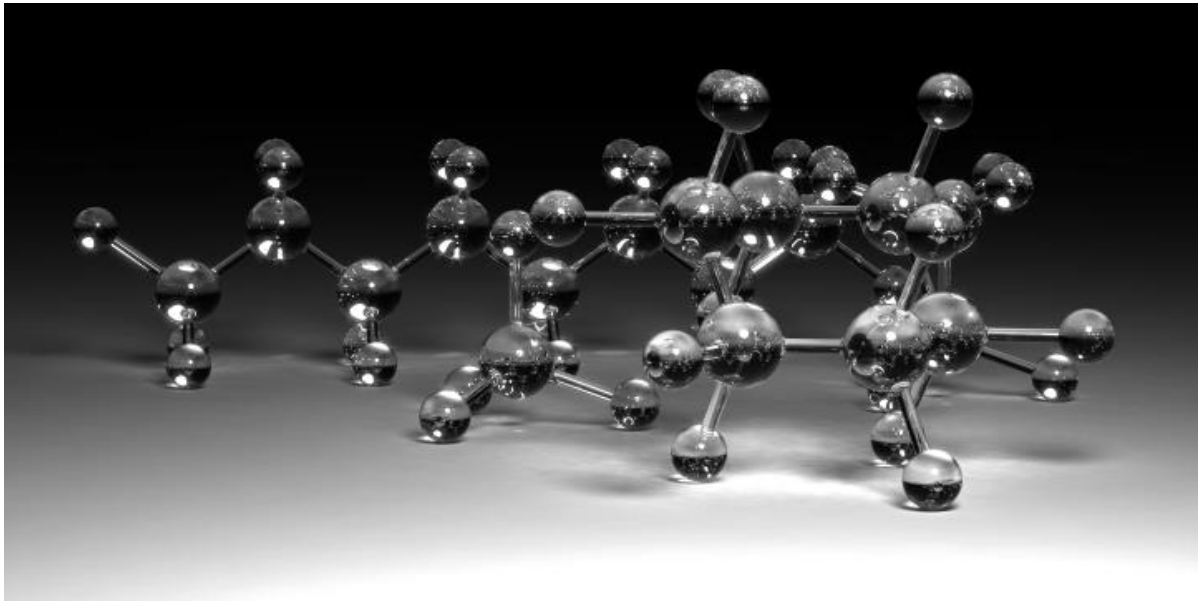
  - Anzahl Berechnungen:

$$\#_{\text{comp}} = \mathrm{O}\left(n \cdot \#_{\text{rays}}\right)$$



Strahlverfolgung
Komplexität – Reduktion

- Zur Reduktion der Komplexität beschränkt man typischerweise die maximale Rekursionstiefe
- Außerdem werden Beschleunigungsdatenstrukturen verwendet, um die Schnittberechnung zwischen einem Strahl und allen Objekten in der Szene zu beschleunigen. Damit kann die Schnittberechnung Strahl-Szene auf $\log n$ reduziert werden
- Resultierende Laufzeit: $\#_{\text{comp}} = \mathrm{O}(w \cdot h \cdot s \cdot m \cdot \log n)$

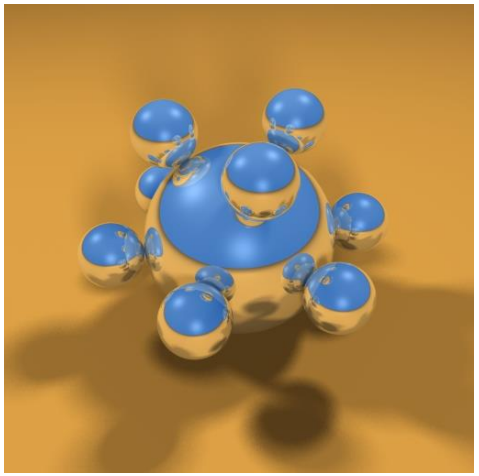


maximale Rekursions-tiefe: 9

© Purpy Pupple

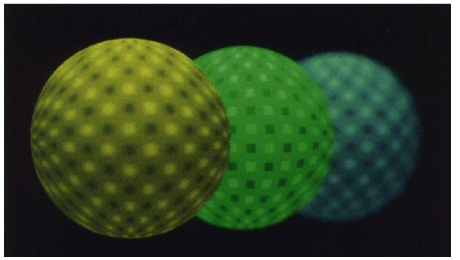
Strahlverfolgung
Distribution Raytracing

- Das Supersampling eines Pixels kann auch verwendet werden, um weiche Schatten, Motion Blur und Tiefenunschärfe zu modellieren
- Dazu werden für jedes Sample eines Pixels zufällig die jeweiligen Parameter ausgewählt:
  - weiche Schatten … Position auf der Flächenlichtquelle
  - Motion Blur … Zeit aus dem zu integrierenden Zeitintervall
  - Tiefenunschärfe … Position auf der ausgedehnten Blendenöffnung
- Dieses Verfahren nennt man Distribution Raytracing

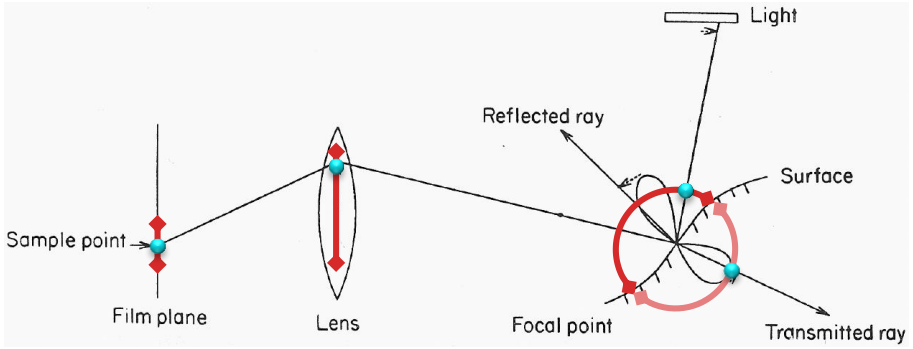


weiche Schatten

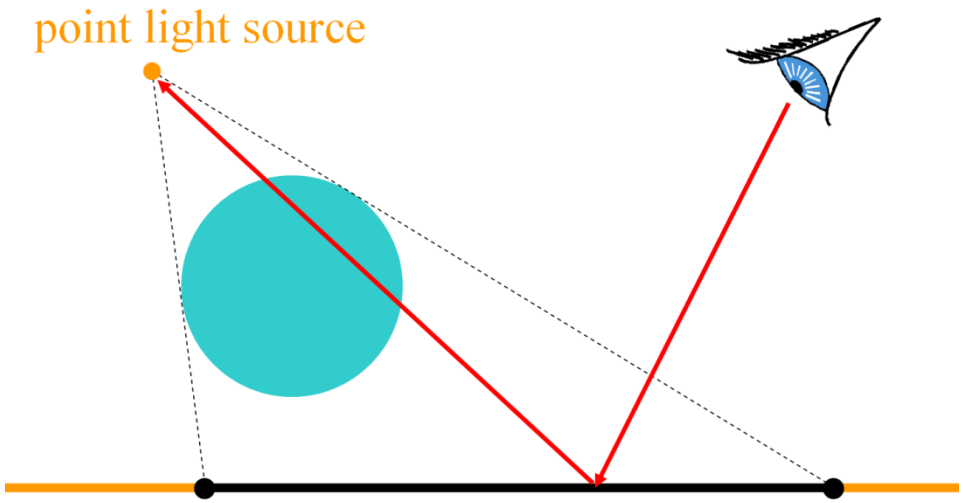
Bewegungsunschärfe

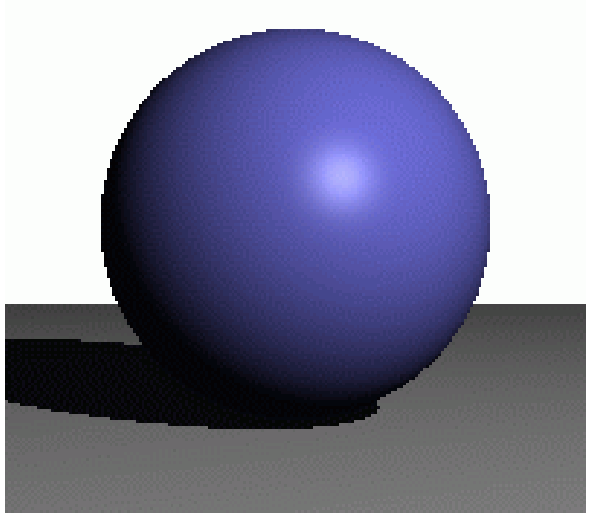


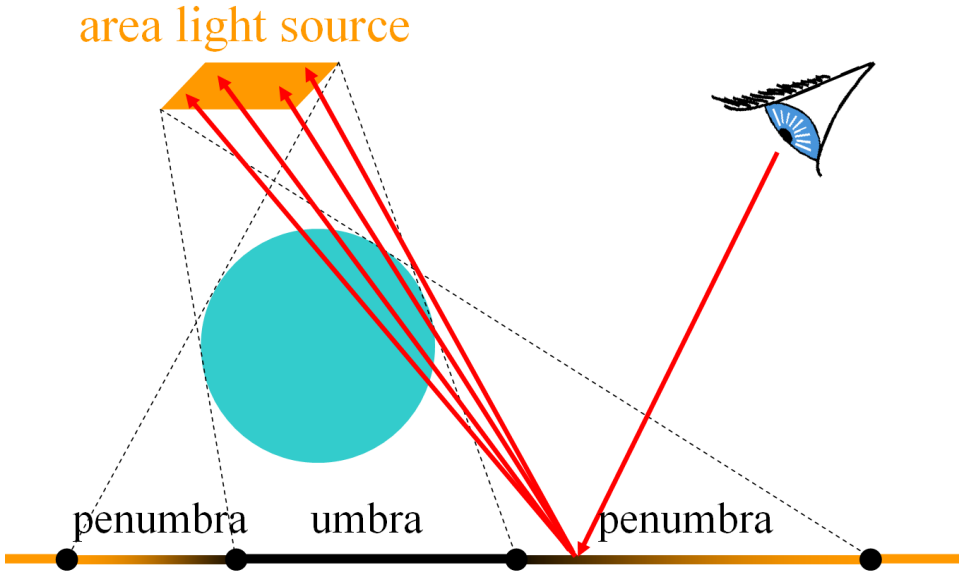
Tiefenunschärfe

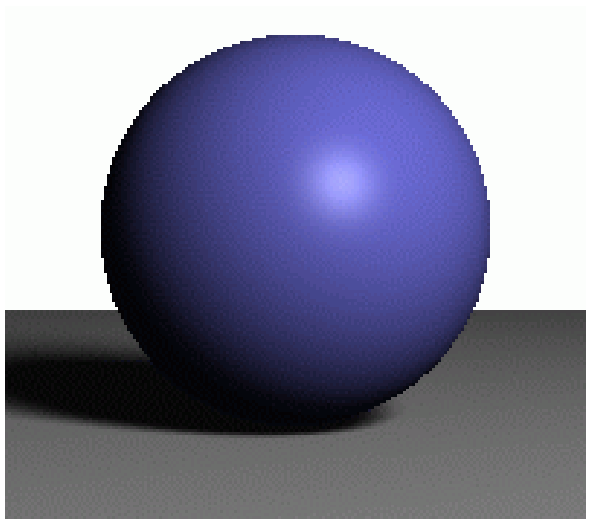


Distribution Raytracing
Weiche Schatten



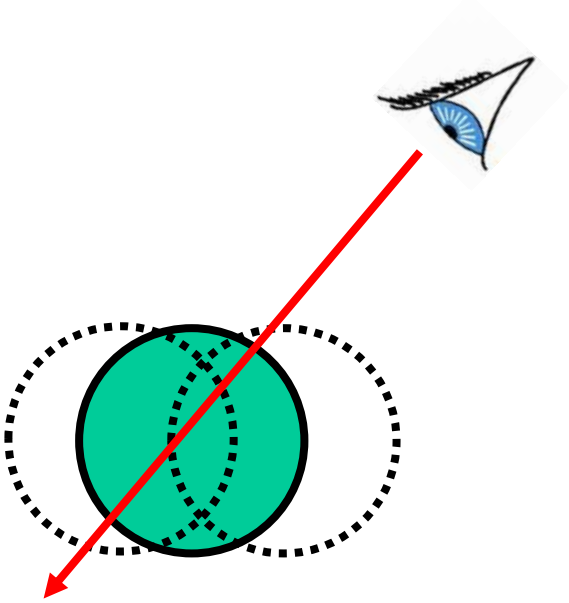






Distribution Raytracing
Motion Blur

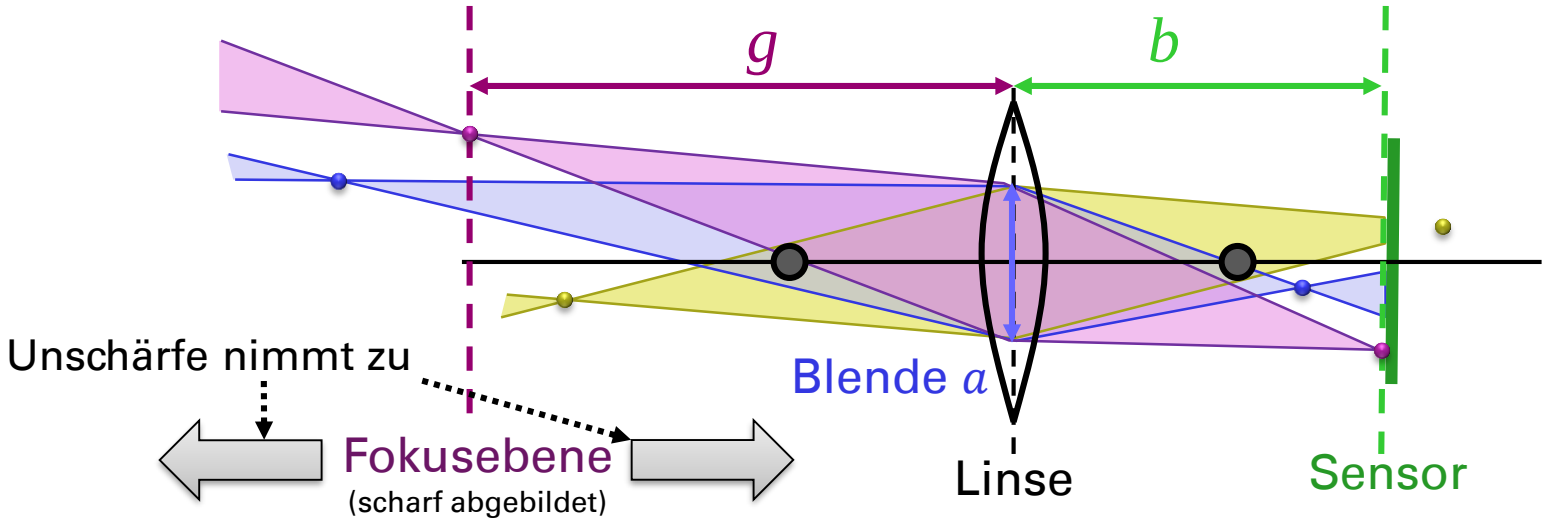
- Für jeden Strahl wird zufällig ein Zeitpunkt in dem darzustellenden Zeitintervall gewählt.
- Vorsicht: hier müssen auch die bewegten Objekte pro Strahl neu positioniert werden.

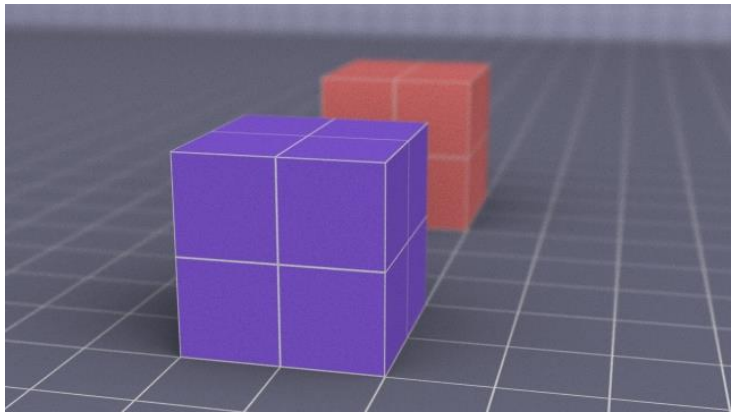


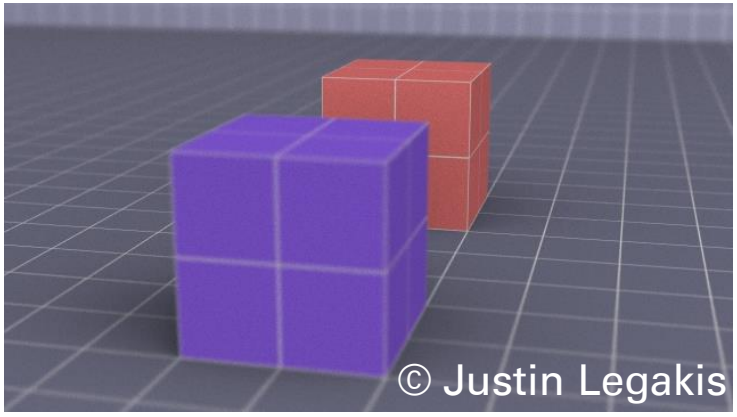
Rob Cook

Distribution Raytracing Tiefenunschärfe

- Strahlen werden von Pixelzentrum durch einen zufällig gewählten Punkt auf der Blendenöffnung konstruiert und an Linse abgebildet

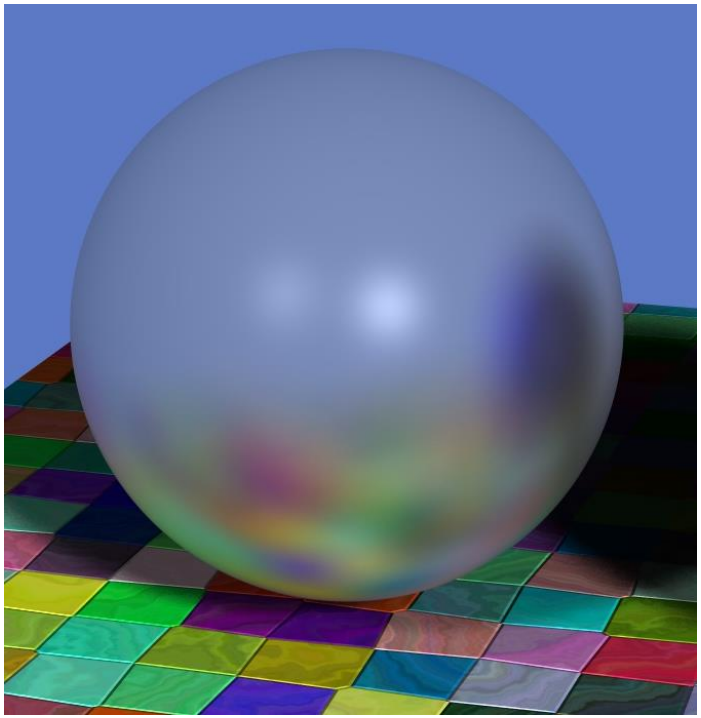




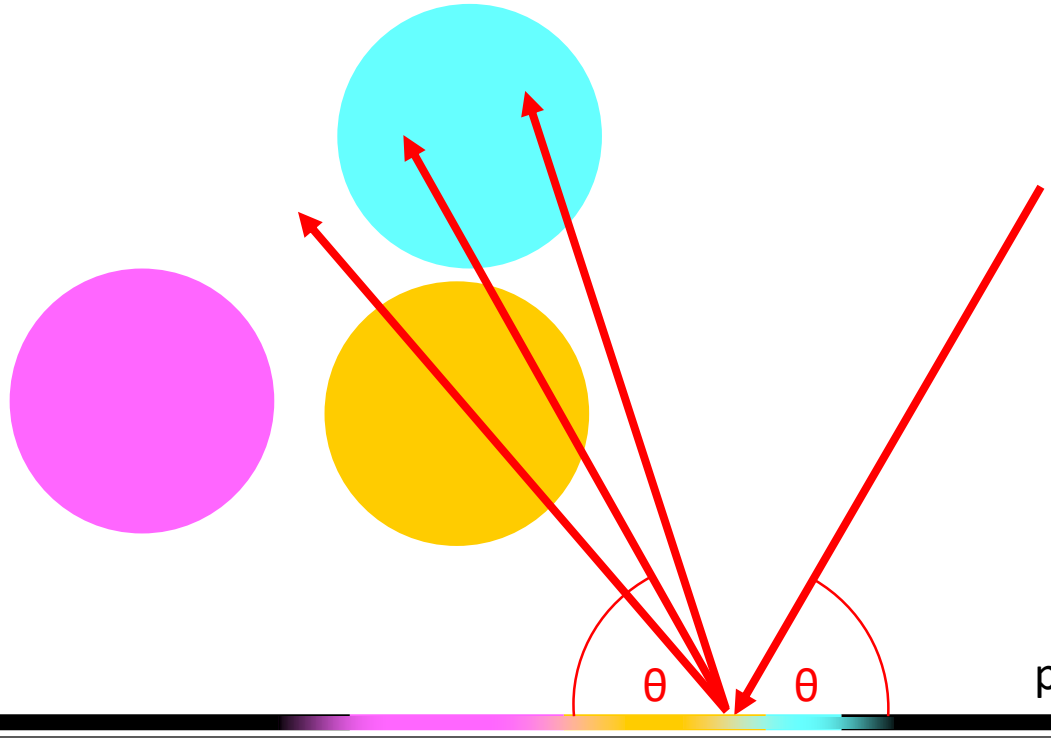


Distribution Raytracing diffuse Spiegelung

- Zusätzlich zu spiegelnden Oberflächen werden auch an gerichtet diffus reflektierenden Oberflächen ein Sekundärstrahl erzeugt, dessen Richtung zufällig um die ideal Spiegelnde Richtung gewählt wird



© Justin Legakis



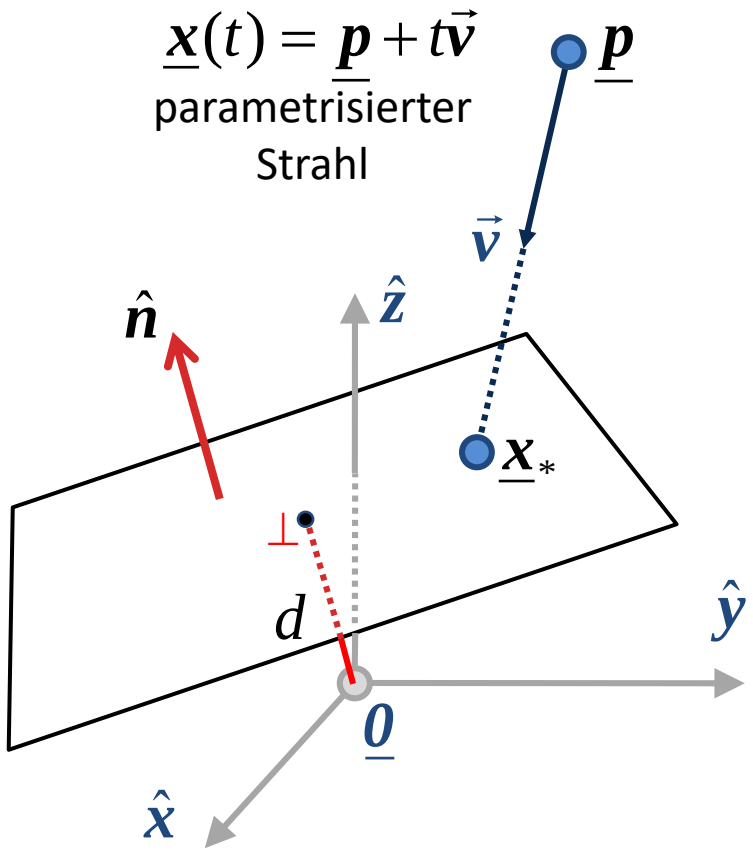
Schnittberechnung Ebene

- Bei vielen Strahl-Primitiv-Schnittberechnungen setzt man die parametrische Form des Strahls in eine implizite Darstellung des Primitives und berechnet den Strahlparameter:

$$\left\langle \hat{\boldsymbol{n}}, \underline{\boldsymbol{p}} + t_* \vec{\boldsymbol{v}} \right\rangle = d \;\Rightarrow t_* = \frac{d - \left\langle \hat{\boldsymbol{n}}, \underline{\boldsymbol{p}} \right\rangle}{\left\langle \hat{\boldsymbol{n}}, \vec{\boldsymbol{v}} \right\rangle}$$

- Als Ergebnis erhält man keinen, einen oder mehrere Schnittparameter (bei Ebene maximal einen). Daraus sucht man sich den mit dem kleinsten positiven t_{fst} aus
- wurde ein t_{fst} gefunden, berechnet man den Schnittpunkt und die Normale des Primitives am Schnittpunkt. Im Falle der Ebene:

$$\underline{\boldsymbol{x}}_{\text{fst}} = \underline{\boldsymbol{p}} + t_{\text{fst}} \vec{\boldsymbol{v}}, \quad \hat{\boldsymbol{n}}_{\text{fst}} = \hat{\boldsymbol{n}}$$



$$\left\langle \hat{\boldsymbol{n}}, \underline{\boldsymbol{x}} \right\rangle = d$$

Hesse'sche Normalform

Schnittberechnung Kugel

- Beim Schnitt von Strahl mit Kugel geht man nach demselben Prinzip vor:

$$\left\|\left(\underline{\boldsymbol{p}}+t_*\vec{\boldsymbol{v}}\right)-\underline{\boldsymbol{c}}\right\|^2=\left\|t_*\vec{\boldsymbol{v}}+\left(\underline{\boldsymbol{p}}-\underline{\boldsymbol{c}}\right)\right\|^2=r^2$$

$$\Rightarrow \left\|\vec{\boldsymbol{v}}\right\|^2 t_*^2+2\left\langle\vec{\boldsymbol{v}},\underline{\boldsymbol{p}}-\underline{\boldsymbol{c}}\right\rangle t_*+\left\|\underline{\boldsymbol{p}}-\underline{\boldsymbol{c}}\right\|^2-r^2=0$$

$$at_*^2+bt_*+c=0$$

- Numerisch kann dies mit folgender Fallunterscheidung stabil gelöst werden:

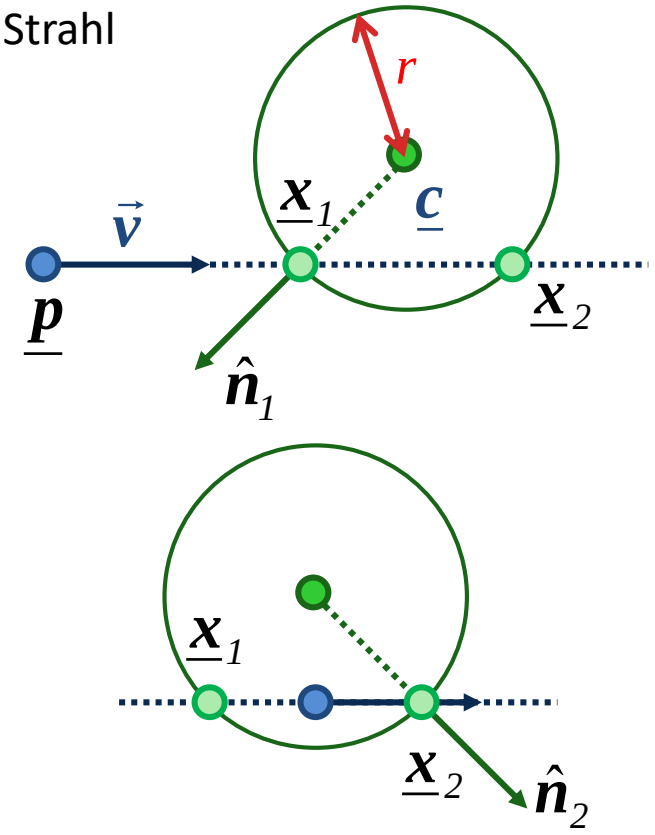
$$t_1=\frac{q}{a},t_2=\frac{c}{q},\ \text{mit}$$

$$q=\begin{cases}-\frac{1}{2}\left(b-\sqrt{b^2-4ac}\right)\ldots\ b<0\\ -\frac{1}{2}\left(b+\sqrt{b^2-4ac}\right)\ldots\text{sonst}\end{cases}$$

- Nach Auswahl von t_{fst} ergibt sich die Normale zu

$$\hat{\boldsymbol{n}}=\left(\underline{\boldsymbol{x}}_{\text{fst}}-\underline{\boldsymbol{c}}\right)/\left\|\left(\underline{\boldsymbol{x}}_{\text{fst}}-\underline{\boldsymbol{c}}\right)\right\|$$

$\underline{\boldsymbol{x}}(t)=\underline{\boldsymbol{p}}+t\vec{\boldsymbol{v}}$
parametrisierter Strahl

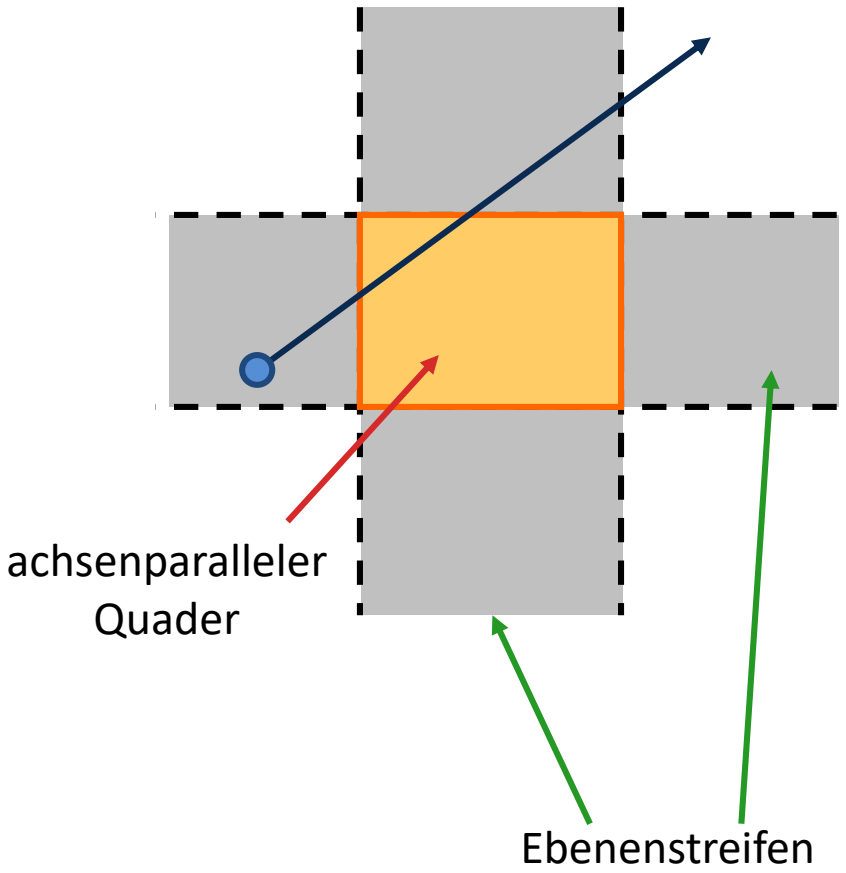


$$\left\|\underline{\boldsymbol{x}}-\underline{\boldsymbol{c}}\right\|^2=r^2$$

implizite Darstellung der Kugel

Schnittberechnung Achsenparalleler Quader

- Quader sind besonders wichtig für Hierarchien von Hüllvolumen.
- Idee: spalte Berechnung in Schnitt mit Ebenenstreifen und diese wiederum in Schnitt mit achsenparallelen Ebenen.
- Bilde Schnittmenge aus resultierenden Intervallen des Strahlparameters t

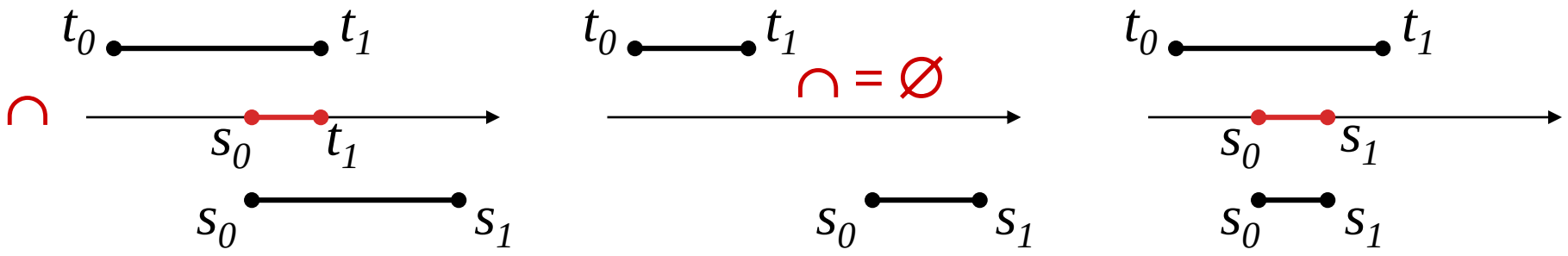


Schnittberechnung
Schnittmenge von Intervallen

- Motivation: der Schnitt zwischen einem Strahl und einem beliebigen Objekt führt auf Intervalle im Parameter t.
- Parameterintervall: $T = [t_0, t_1]$ mit $t_0 \leq t_1$ oder $T = \varnothing$

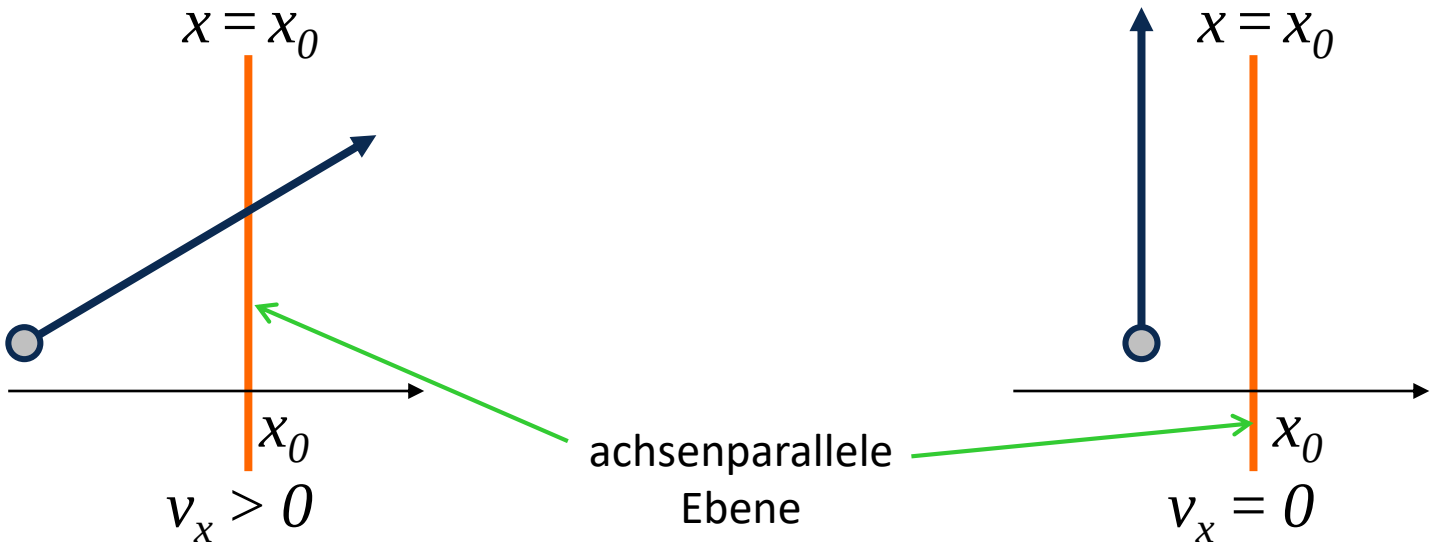
- Schnittmengenoperation:

$$[t_0, t_1] \cap [s_0, s_1] = \begin{cases} \emptyset, & \text{wenn } \max\{t_0, s_0\} > \min\{t_1, s_1\} \\ [\max\{t_0, s_0\}, \min\{t_1, s_1\}], & \text{sonst} \end{cases}$$



Schnittberechnung
Achsenparallele Ebene

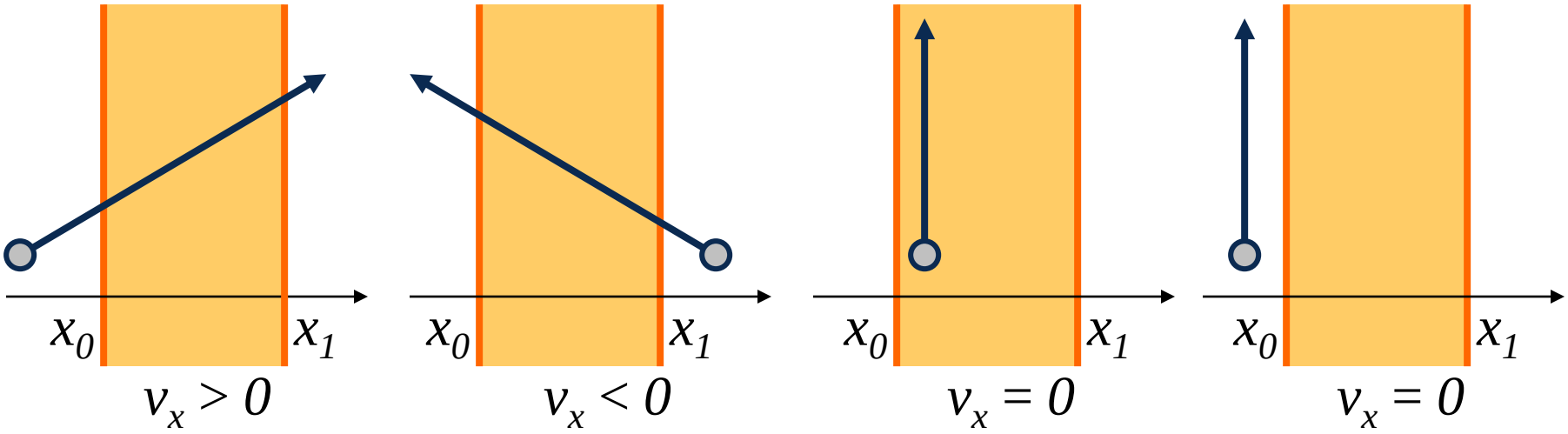
- Das Schnittproblem lässt sich auf eine Dimension reduzieren: der Strahl beginnt bei $x(t=0) = p_x$ und die x-Komponente wächst mit der Geschwindigkeit $\partial x/\partial t = v_x$
- Aus $x_0 = p_x + t_* \cdot v_x$ ergibt sich der Schnittparameter t_* zu: $t_* = \begin{cases} \dfrac{x_0 - p_x}{v_x} : v_x \neq 0 \\ \text{undef} : v_x = 0 \end{cases}$



Schnittberechnung Achsenparalleler Ebenenstreifen

- Intervall wird abhängig vom Vorzeichen von v_x berechnet
- Weitere Fallunterscheidung bei Parallelität

$$[t_0, t_1] = \begin{cases} \left[\frac{x_0 - p_x}{v_x}, \frac{x_1 - p_x}{v_x}\right] & : v_x > 0 \\ \left[\frac{x_1 - p_x}{v_x}, \frac{x_0 - p_x}{v_x}\right] & : v_x < 0 \\ [-\infty, +\infty] & : v_x = 0 \wedge p_x \in [x_0, x_1] \\ \emptyset & : \text{sonst} \end{cases}$$

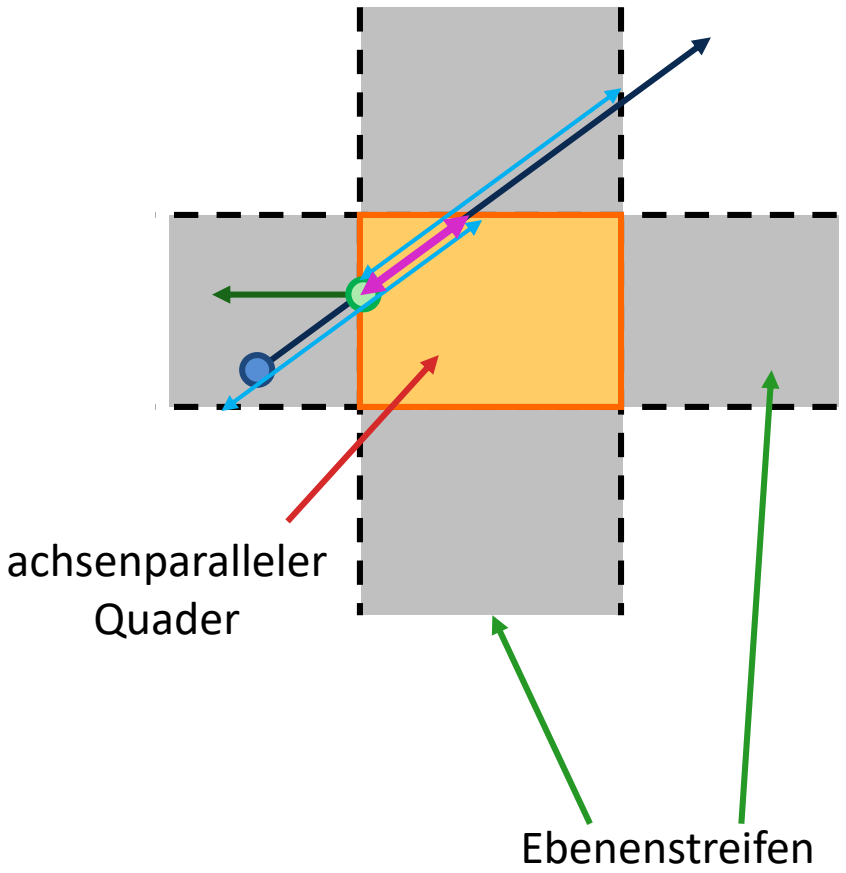


Schnittberechnung Achsenparalleler Quader

- Gegeben: Quader, definiert durch zwei Punkte aus den minimalen und maximalen Koordinaten: $\underline{\boldsymbol{x}}_{\min}, \underline{\boldsymbol{x}}_{\max}$
- Berechne Schnittintervall $T_{x/y/z}$ mit den drei Ebenenstreifen gemäß voriger Folie.
- Schnittintervall T_Q mit Quader ergibt sich aus dem Schnitt über alle drei Intervalle:

$$T_Q = T_x \cap T_y \cap T_z$$

- prüfe abschließend ob es ein $0 < t_{\text{fst}} \in T_Q$ gibt.
- die Normale ergibt sich entlang einer Koordinatenachse



Schnittberechnung Dreieck

- Setzt man $\sigma_2=1-\sigma_0-\sigma_1$, kann man den Strahlparameter t_* sowie σ_{0*} und σ_{1*} durch Gleichsetzen der parametrischen Strahldarstellung und der baryzentrischen Dreiecksdarstellung berechnen.
- Anschließend testen, ob alle $\sigma_i>0$ sind und prüfen, ob $t_*>0$
- falls ja, $\underline{\boldsymbol{x}}_{\text{fst}}$ und $\underline{\boldsymbol{n}}_{\text{fst}}$ berechnen:

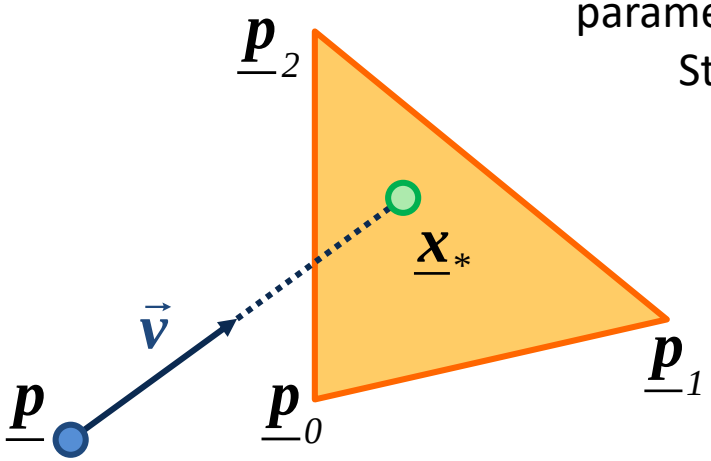
$$\hat{\boldsymbol{n}}_{\text{fst}} = \vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2) \,/\, \left\|\vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2)\right\|$$

mit der Definition

$$\vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2) = \left(\underline{\boldsymbol{p}}_1 - \underline{\boldsymbol{p}}_0\right) \times \left(\underline{\boldsymbol{p}}_2 - \underline{\boldsymbol{p}}_0\right)$$

$$\underline{\boldsymbol{x}}(t) = \underline{\boldsymbol{p}} + t\vec{\boldsymbol{v}}$$

parametrisierter Strahl



$$\underline{\boldsymbol{x}} = \sigma_0\,\underline{\boldsymbol{p}}_0 + \sigma_1\,\underline{\boldsymbol{p}}_1 + \sigma_2\,\underline{\boldsymbol{p}}_2, \forall i : \sigma_i \geq 0$$

Dreieck in baryzentrischer Darstellung

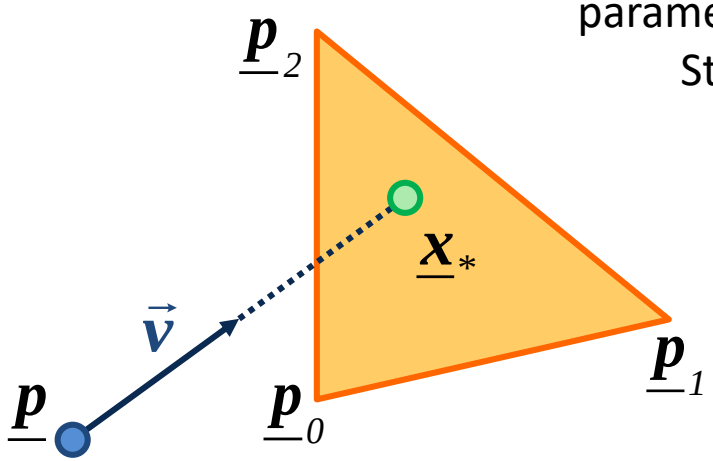
Schnittberechnung Dreieck II

- Man kann optional auch den Schnittparameter t_* wie beim Schnitt mit einer Ebene bestimmen und die σ_{0*} direkt aus $\underline{\boldsymbol{x}}_*$ und den $\underline{\boldsymbol{p}}_i$ berechnen:

$$t_* = \frac{\left\langle \vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2), \underline{\boldsymbol{p}}_0 - \underline{\boldsymbol{p}} \right\rangle}{\left\langle \vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2), \vec{\boldsymbol{v}} \right\rangle}$$

$$\underline{\boldsymbol{x}}(t) = \underline{\boldsymbol{p}} + t\vec{\boldsymbol{v}}$$

parametrisierter Strahl



$$\sigma_0 = f \cdot \left\langle \vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2), \vec{\boldsymbol{n}}(\underline{\boldsymbol{x}}_*\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2) \right\rangle$$

$$\sigma_1 = f \cdot \left\langle \vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2), \vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{x}}_*\,\underline{\boldsymbol{p}}_2) \right\rangle$$

$$\sigma_2 = f \cdot \left\langle \vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2), \vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{x}}_*) \right\rangle \text{ mit } f = \frac{1}{\left\| \vec{\boldsymbol{n}}(\underline{\boldsymbol{p}}_0\,\underline{\boldsymbol{p}}_1\,\underline{\boldsymbol{p}}_2) \right\|^2}$$

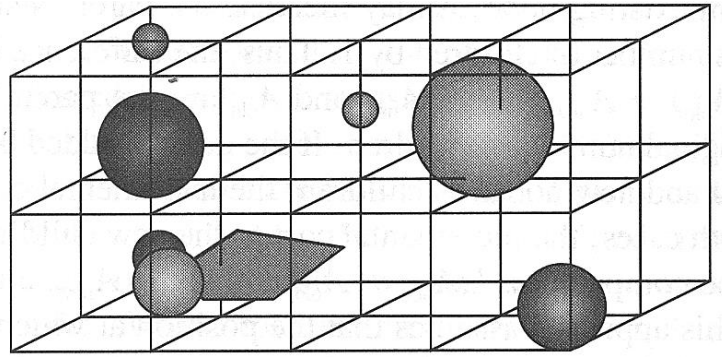
$$\underline{\boldsymbol{x}} = \sigma_0\,\underline{\boldsymbol{p}}_0 + \sigma_1\,\underline{\boldsymbol{p}}_1 + \sigma_2\,\underline{\boldsymbol{p}}_2, \forall i : \sigma_i \geq 0$$

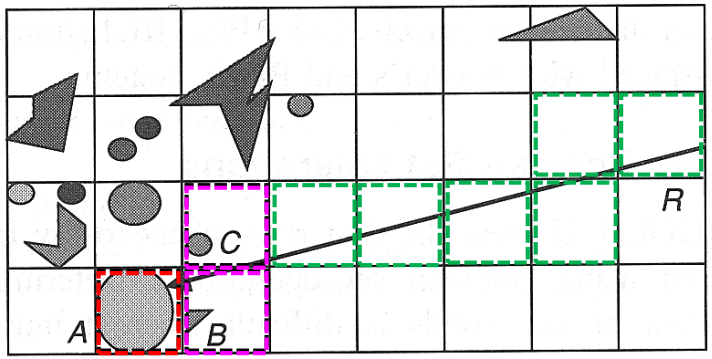
Dreieck in baryzentrischer Darstellung

- Beim Vergleich $\sigma_i \geq 0$ kann f auch ignoriert werden

Beschleunigungstechniken
Gitter

- Unterteile Bounding Box in regelmäßiges Voxel
- Auflösung: oft $\sqrt[3]{n}$
- Objekte einfügen
  - füge in alle Voxel ein, die das Objekt überlappen
  - leicht optimierbar
- Traversierung
  - verfolge Voxel entlang des Strahls
  - schneide jeweils mit enthaltenen Objekten
  - terminiere wenn erster Schnitt gefunden wurde





Gitter
Strahlverfolgung

- verfolge Zeitpunkte der Voxelübergänge für jede Achse getrennt
- berechne Startzelle aus:

$$i = \text{floor}\left(\frac{p_x - x_{\min}}{\Delta x}\right), \Delta x = \frac{x_{\max} - x_{\min}}{n}$$

- und erste Schnittzeiten $t_{\alpha,0}$ aus:

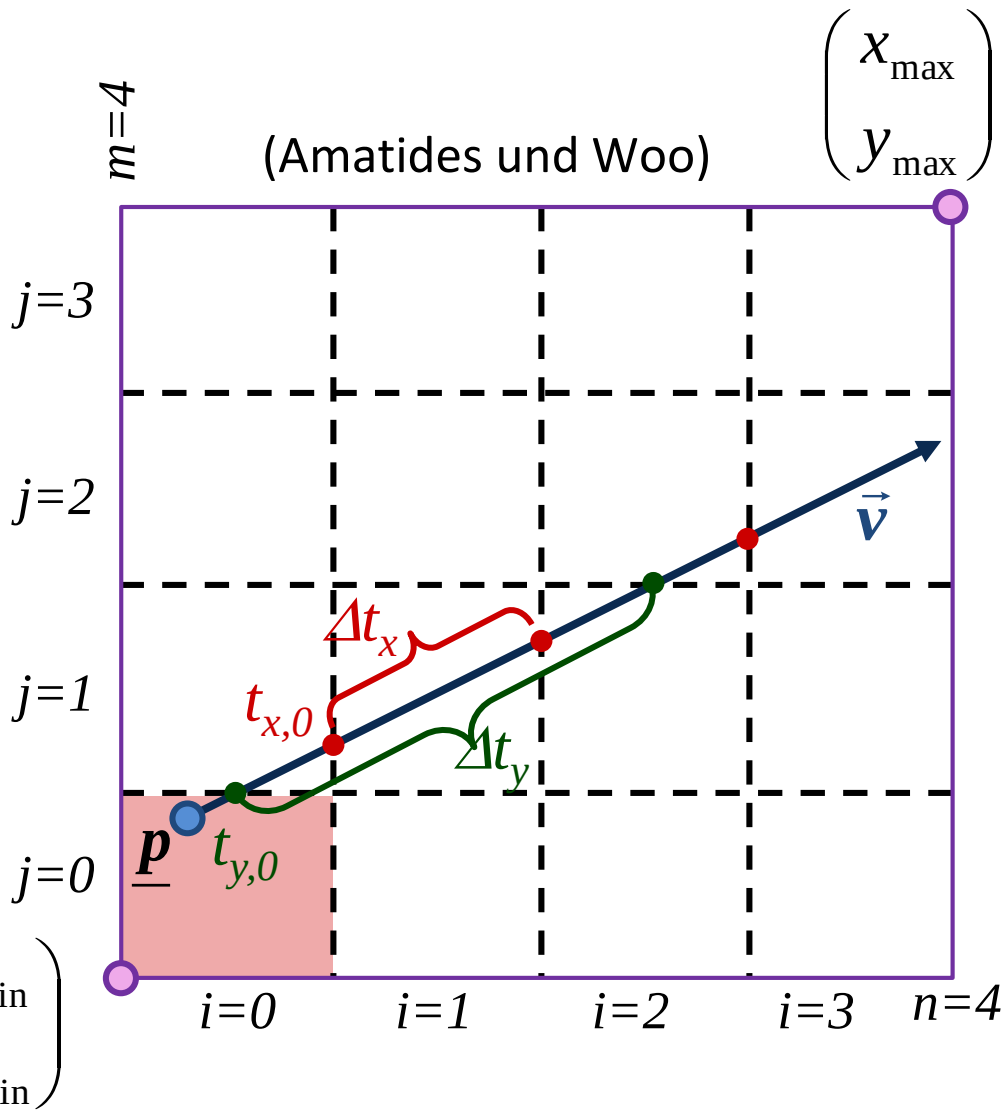
$$x_{\min} + (i+1)\Delta x = p_x + v_x t_{x,0}$$

- Zeitschritte Δt_α zum nächsten Wechsel für alle $\alpha \in \{x,y,z\}$

$$\Delta x = v_x \Delta t_x$$

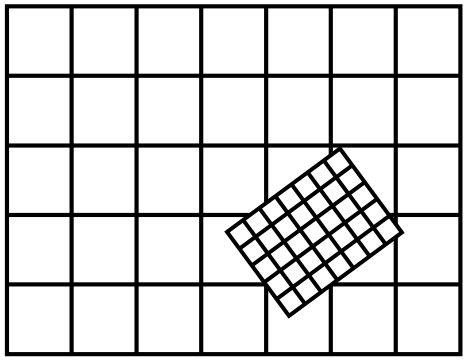
- verfolge Strahl entlang t und führe nächsten Wechsel durch

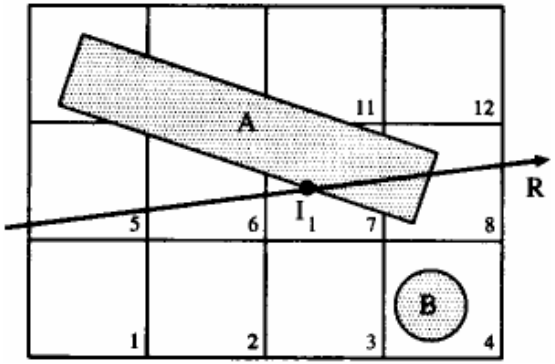
$$\begin{pmatrix} x_{\min} \\ y_{\min} \end{pmatrix}$$

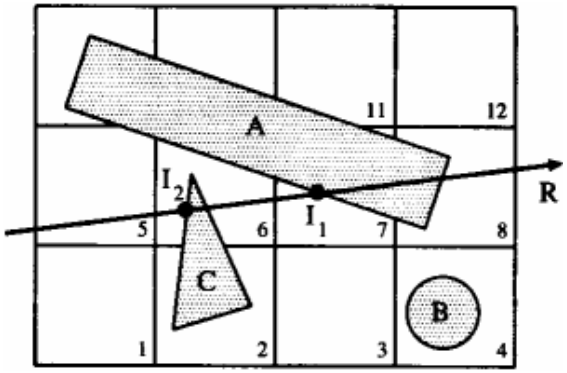


Gitter
Diskussion

- Gitterauflösung
  - hängt stark von Szene ab
  - keine Adaption an einen Fußball in einem Fußballstadium
  - mögliche Abhilfe: geschachtelte Gitter
- **Mailbox Technik**
  - Einsatz: wenn Objekte von mehreren umgebenden Volumen referenziert werden können, wie z.B. beim Gitter
  - Gebe jedem Strahl eine eindeutige ID und jedem Objekt eine Mailbox für letzte Schnittberechnung (ID, Schnittinfo)
  - Prüfe vor Schnittberechnung Mailbox des Objektes und speichere nach evtl. Neuberechnung Ergebnis in Mailbox

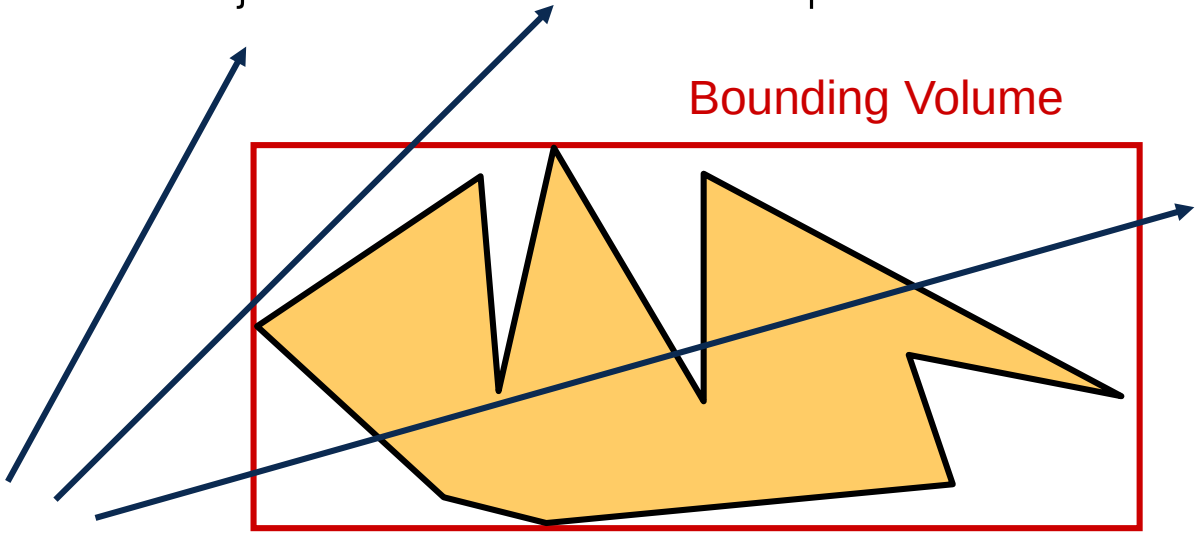






Beschleunigungstechniken
Hüllvolumen

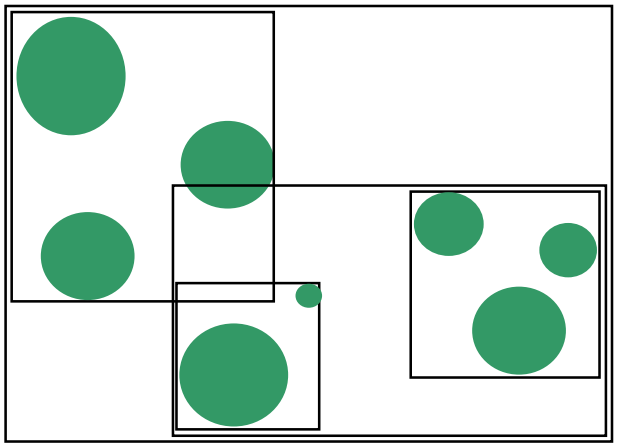
- Grundidee: Das Ziel besteht darin, Kosten bei den Schnittpunkttests eines Strahls mit den Objekten der Teilszene zu sparen.

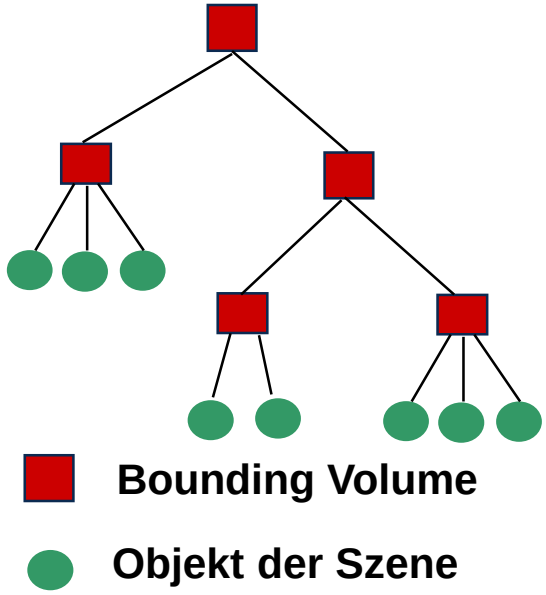


- Verfehlt der Strahl das Hüllvolume, so kann man auf den Schnitt mit der Teilszene verzichten
- Oft bietet es sich an, aus der Szene alle geometrischen Primitive (z.B. alles in Dreiecke zerlegen) in einem globalen Koordinatensystem zu extrahieren und darauf die Beschleunigungstechniken anzuwenden

Hüllvolumen Hierarchie

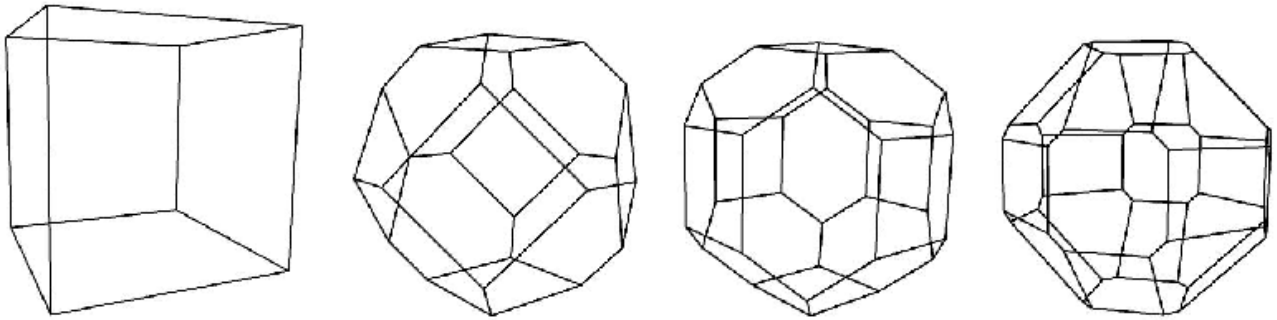
- Idee: Umschließe die Objekte in einer Hierarchie von sich zum Teil überlappenden Hüllvolumen (BVH)
- Teilbäume: Objekte nahe in der Szene beieinander
- minimiere Oberflächeninhalte der Hüllvolumen
- Fokus auf obere Knoten, weil hier das Abschneiden eines Teilbaumes mehr spart
- Berechnungszeit für Bild durch Raytracing sehr viel geringer trotz zusätzlicher Vorverarbeitung





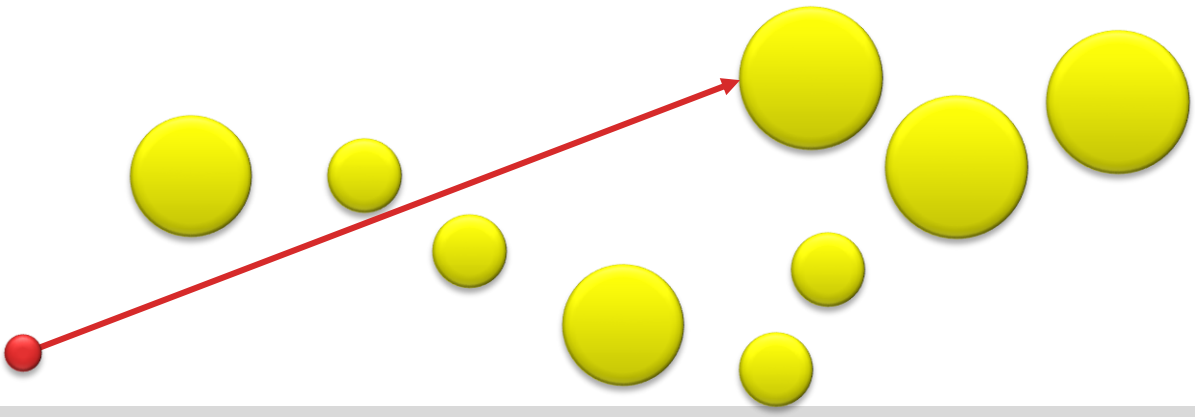
Hüllvolumen
Entwurfskriterien

- Hüllvolumen werden so konstruiert, dass Teilobjekte komplett umschlossen werden ⇒ keine Mailbox-Techniken nötig
- Man muss abwägen zwischen
  - einfache Hüllvolumen (z.B. Kugel, Ellipsoid, Quader) mit kleinen Schnittkosten haben relativ hohe Strahltrefferzahlen
  - enganliegende Hüllvolumen (z.B. konvexe Hülle) mit kleiner Strahltrefferzahl haben hohe Schnittkosten.
- Einen guten Kompromiss ergeben k-Dops:



6-DOP 14-DOP 18-DOP 26-DOP

Hüllvolumenhierarchie (BVH) nicht beschleunigt



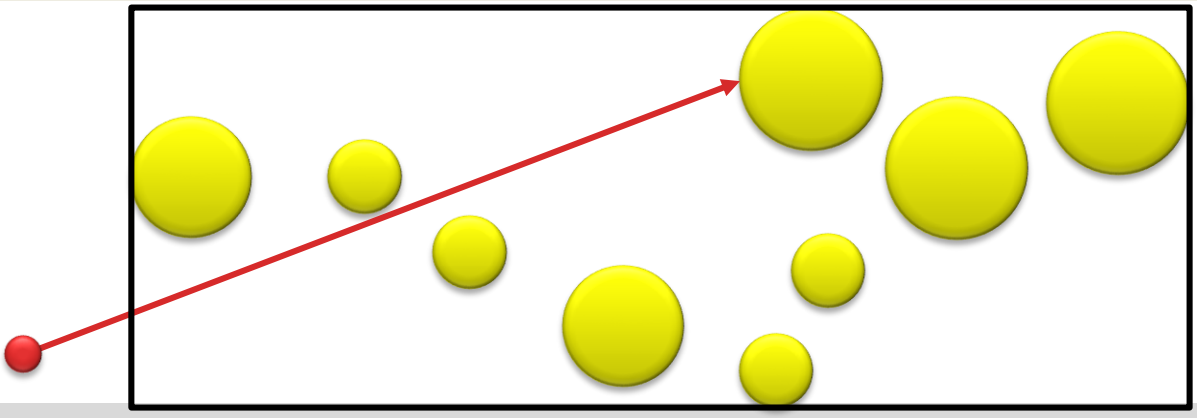
```
[t*, hit] find_first_hit(ray, prims, n)
    t* = ∞;
    for (i=0; i<n; ++i)
        [t, hit_i] = prims[i].first_intersection(ray);
        if (t > 0 && t < t*)
            [t*, hit] = [t, hit_i]
    return [t*, hit]
```

$O(c_P)$

- Der nicht beschleunigte Schnitt von Strahl mit n Primitiven hat eine Laufzeit von $T_P = n \cdot c_P$

  c_P … Zeit für Primitiv-Strahl-Schnittberechnung)

Hüllvolumenhierarchie (BVH) ein Hüllvolumen

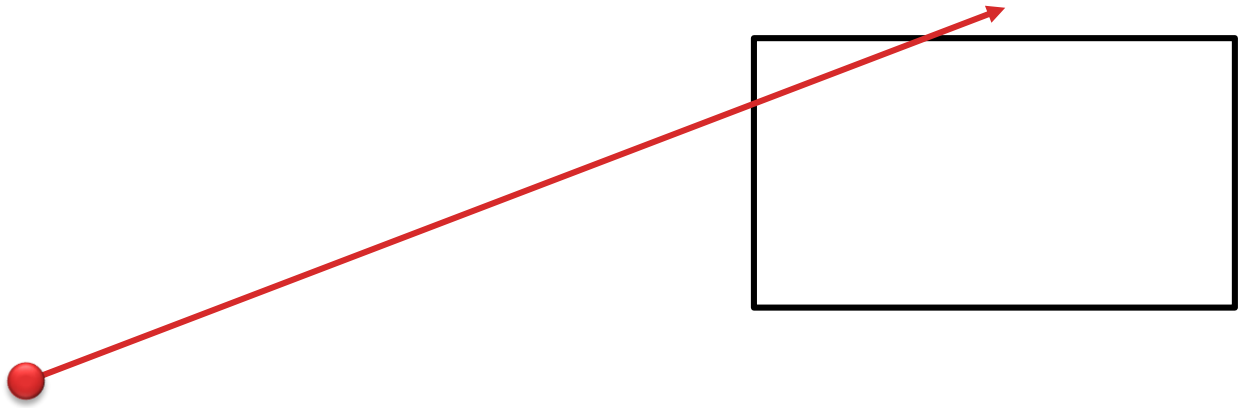


```
[found,hit] find_first_hit_BV(ray, prims, BV, n)
    if (BV.intersection_interal(ray)== ∅)      ... O(c_B)
        return [false, ∅];
    else
        return find_first_hit(ray, prims, n)   ... O(n · c_P)
```

- Mit einem Hüllvolumen (Bounding Volume) hängt Laufzeit von Wahrscheinlichkeit p_{BV} ab, dass ein Strahl das Hüllvolumen trifft: $T = c_{BV} + p_{BV} \cdot n \cdot c_P$

  c_{BV} … Zeit für Hüllvolumen-Strahl-Schnittberechnung)

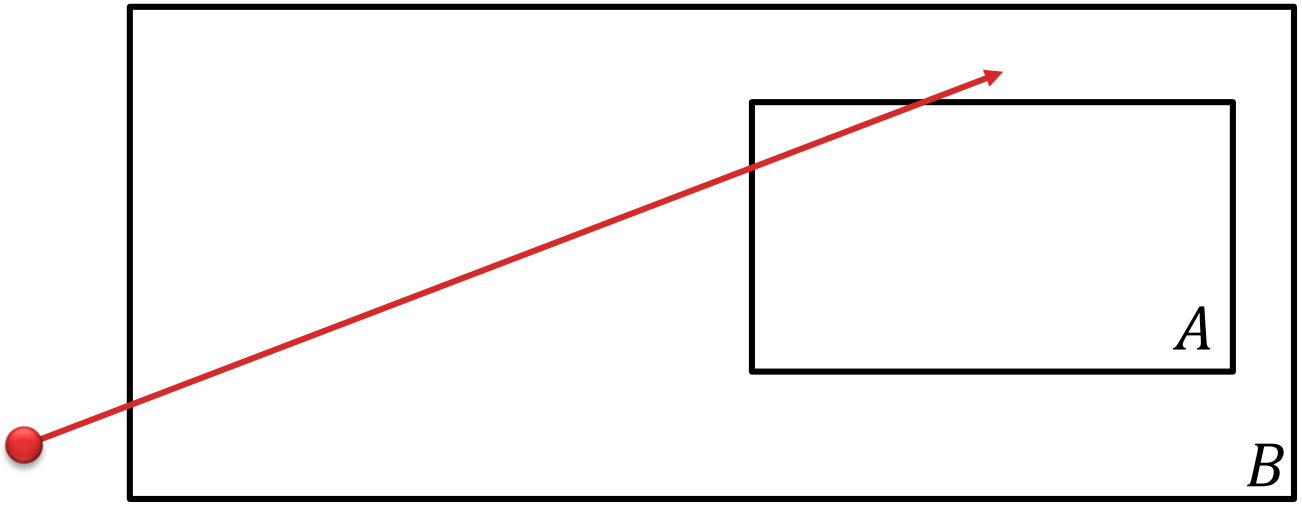
Hüllvolumenhierarchie (BVH)
Surface Area Heuristics (SAH)



- Für konvexe Hüllvolumen wird im Rendering meist die Surface Area Heuristic verwendet, die besagt, dass p_{BV} proportional zur Oberflächeninhalt ist: $p_{BV} \propto SA_{BV}$

  SA_{BV} … Oberfläche des Hüllvolumens
  - $2(ab + bc + ca)$ für Quader der Kantenlängen a, b, c
  - $4\pi r^2$ für Kugel mit Radius r

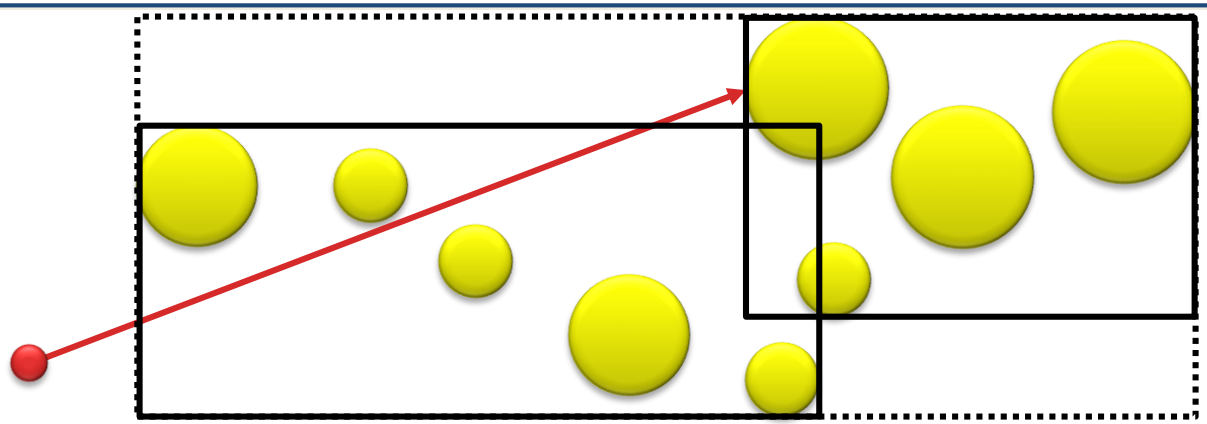
Hüllvolumenhierarchie (BVH)
Surface Area Heuristics (SAH)



- Weil Proportionalitätskonstante für p_{BV} schwer zu ermitteln ist, betrachtet man die bedingte Wahrscheinlichkeit $p_{A|B}$, dass Strahlen, die Hüllvolumen B getroffen haben auch Hüllvolumen A treffen. Dann gilt

$$p_{A|B} = SA_A / SA_B$$

Hüllvolumenhierarchie (BVH)
Aufteilen der Primitive



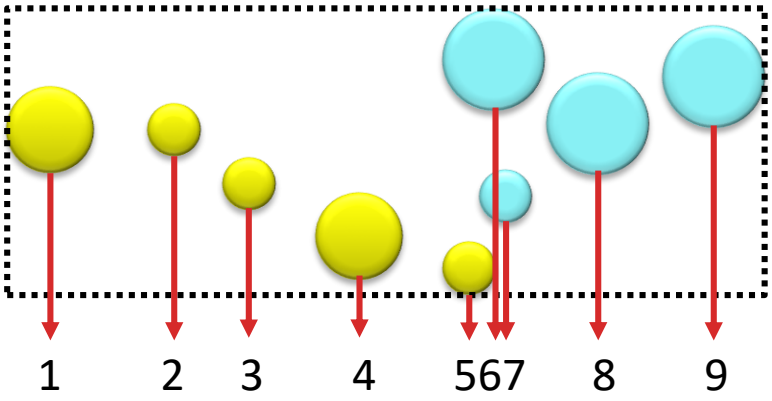
- Für das Erstellen einer Hüllvolumenhierarchie (Bounding Volume Hierarchy) werden die Primitive in Teilmengen partitioniert (hier immer in zwei Teilmengen)
- Für jede Teilmenge berechnet man getrennte Hüllvolumen, die sich überlappen können.
- Primitive müssen nicht zerteilt oder mehrfach eingetragen werden (keine Mailbox-Technik benötigt)

Hüllvolumenhierarchie (BVH)
Aufteilen der Primitive

Computergraphik und Visualisierung

Mögliche Strategien zum Teilen:

- Medianprimitiv (Primitive, das in Mitte der Sortierung entlang einer Hauptrichtung steht)
  Parameter: Hauptrichtung x,y,z
- BV-Median (Primitive die links/rechts einer mittigen Unterteilung des Hüllvolumens entlang Hauptrichtung liegen)
  Parameter: Hauptrichtung x,y,z



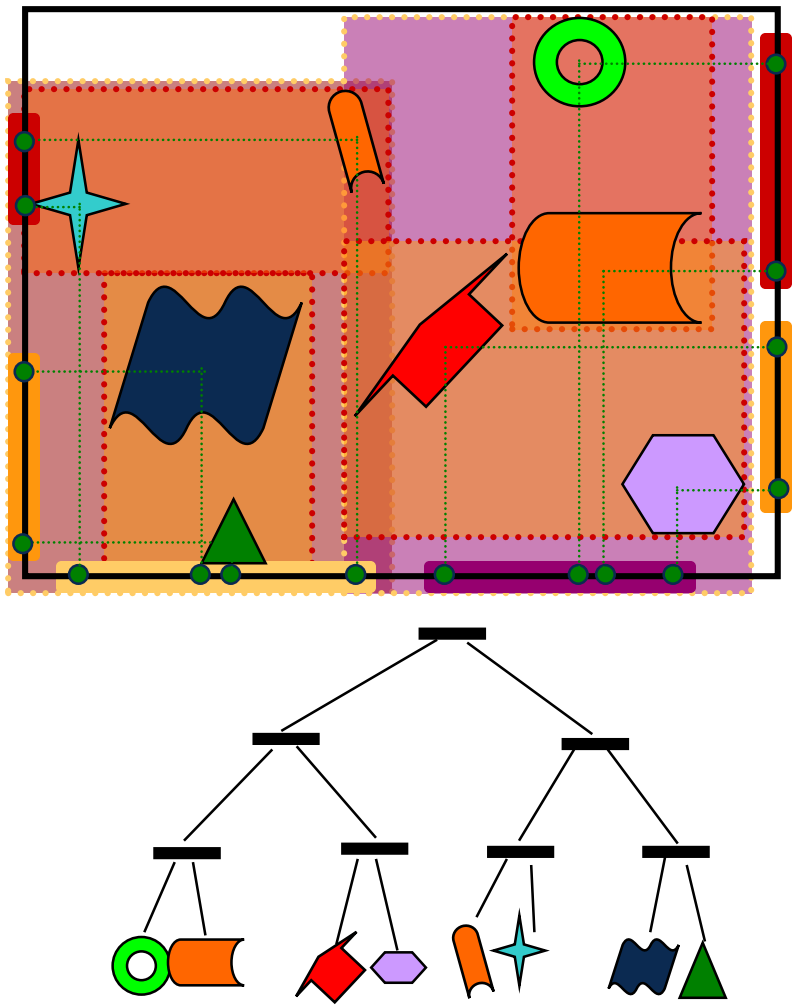
- Minimiere erwartete Laufzeit mittels SAH über alle Partitionierungen der Primitivmenge P in Teilmengen L, R:

$$L \cup R = \min_{\forall L \cup R = P} T_{SAH}\,, T_{SAH} = c_{BV} + \left(p_{L|P} \cdot |L| \cdot + p_{R|P} \cdot |R|\right) c_P$$

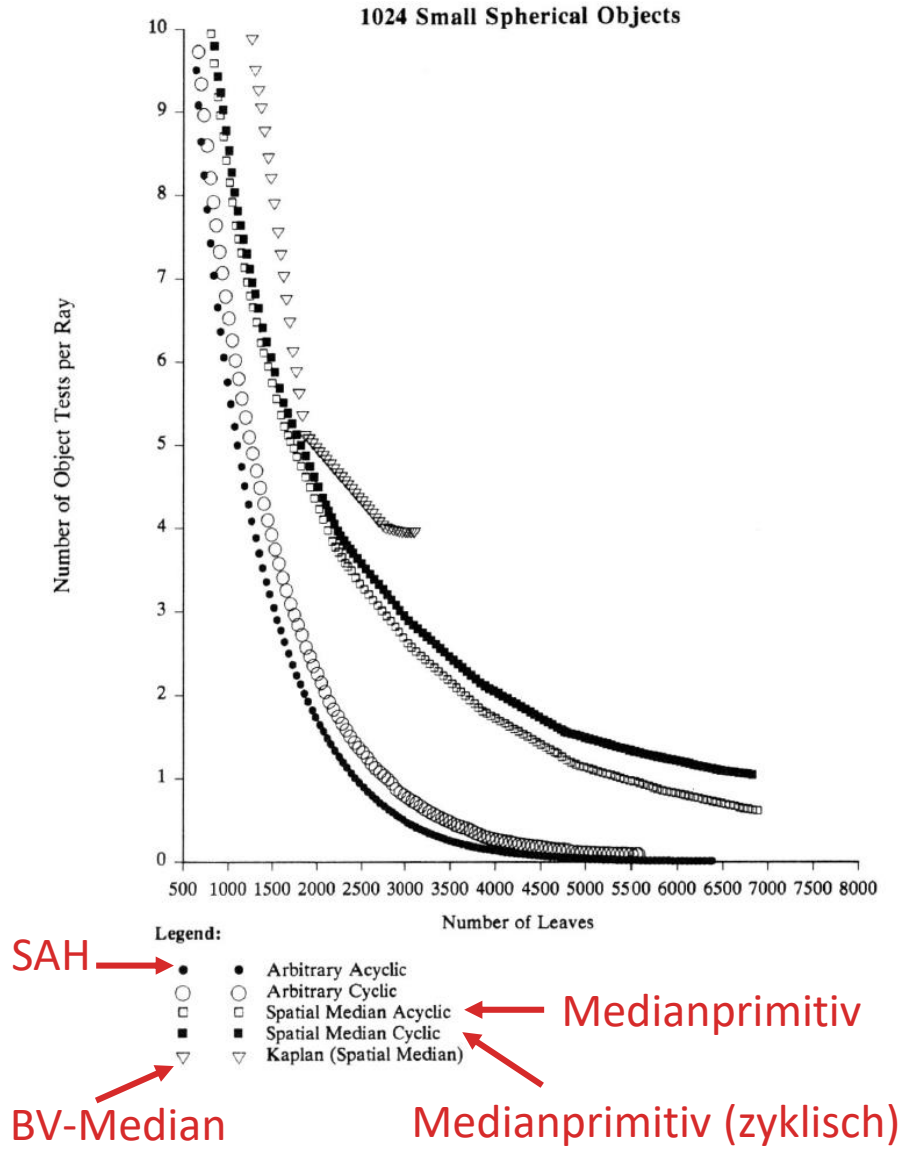
Hüllvolumenhierarchie (BVH)
Aufteilen Medianprimitiv (zyklisch)

Aufbau (top-down):

- Umschließe alle Objekte mit axis aligned bounding box (AABB) der Szene
- Spalte Box entlang erster Koordinatenrichtung und verteile Objekte gleichmäßig
- berechne Kindboxen so, dass sie die enthaltenen Objekte ganz umschließen
- dadurch können sich die Kind-boxen gegenseitig überlagern
- Rekursion mit nächster Koordinatenrichtung bis die Anzahl der Objekte in den Kindboxen klein genug



Hüllvolumenhierarchie (BVH)
Aufteilen der Primitive

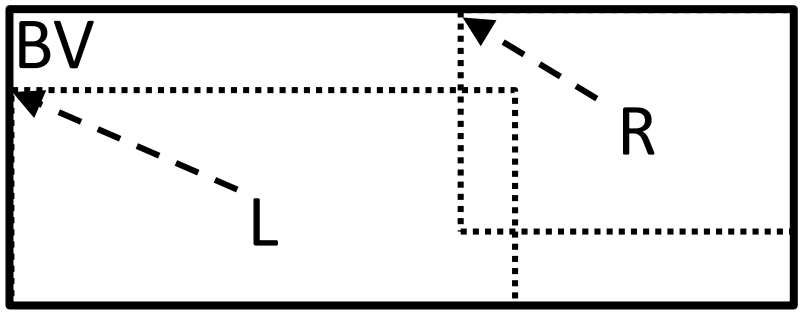


- SAH Optimierung erfolgt nicht über alle Partitionierungen sondern über in einer der 3 Hauptrichtungen sortierten Unterteilungen
- *MacDonald und Booth* (pdf) untersuchten 1989 unterschiedliche Aufteilungsstrategien und fanden heraus, dass SAH zu den performantesten Hierarchien führt
- Man kann die Unterteilung stoppen wenn $T_{SAH} > T_P$

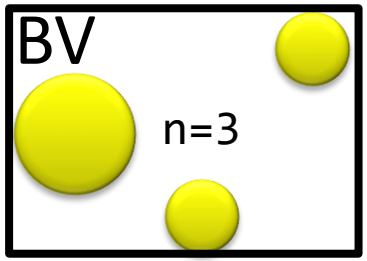
Hüllvolumenhierarchie (BVH)
Hierarchieknotentypen

Computergraphik und Visualisierung

```
struct node_type {
   virtual bool is_leaf() = 0;
};
struct inner_node_type :
   public node_type
{
   bool is_leaf() { return false; }
   BV_type BV;
   node_type *L, *R;
};
struct leaf_node_type :
   public node_type
{
   bool is_leaf() { return true; }
   BV_type BV;
   size_t n;
   primitive_type* prims;
}
```



innerer Hierarchieknoten



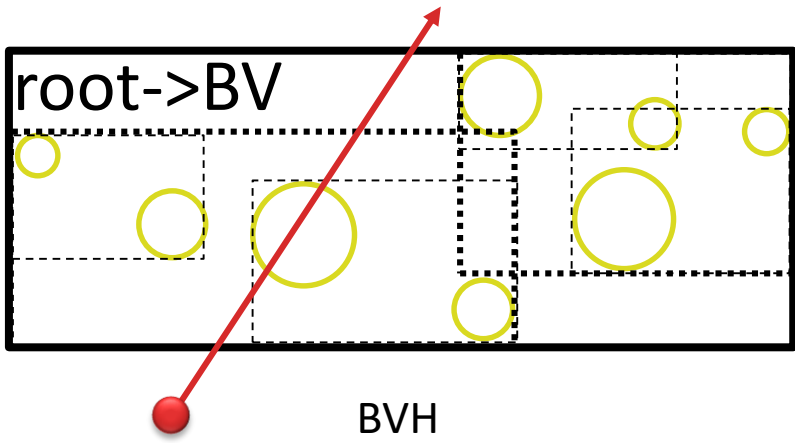
Blattknoten

Hüllvolumenhierarchie (BVH) Aufbau


```
[node] build_BVH(prims, n)
    T_P = n*c_P;
    [T_SAH*,i*,axis*] = [∞,0,'x'];
    BV_P = compute_BV(prims, n);
    // sort along x and check all sorted splits
    prims_x = sort_x(prims);
    for (i=1; i<n; ++i)
        BV_L = compute_BV(prims,i);
        BV_R = compute_BV(prims+i,n-i);
        p_L_P = BV_L.SA()/BV_P.SA();
        p_R_P = BV_R.SA()/BV_P.SA();
        T_SAH = c_BV + (p_L_P*i + p_R_P*(n-i))*c_P;
        if (T_SAH < T_SAH*)
            [T_SAH*,i*,axis*] = [T_SAH, i, 'x'];
    // repeat checks for y and z directions
    if (T_SAH* < T_P)
        return construct_inner_node(prims, n, i*, axis*);
                (rekursively calls build_BVH for child nodes)
    else
        return construct_leaf(prims, n)
```

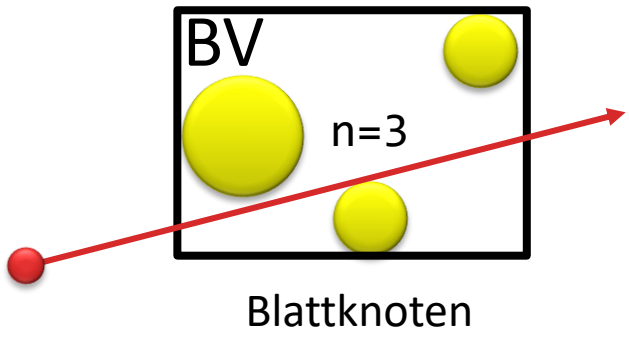
Hüllvolumenhierarchie (BVH) Strahlschnitt

- Strahlschnitt prüft auf Schnitt mit BV der Hierarchiewurzel und terminiert falls kein Schnitt
- Dann wird rekursive Funktion mit Strahlparameter $t^* = \infty$ des ersten Schnitts aufgerufen:

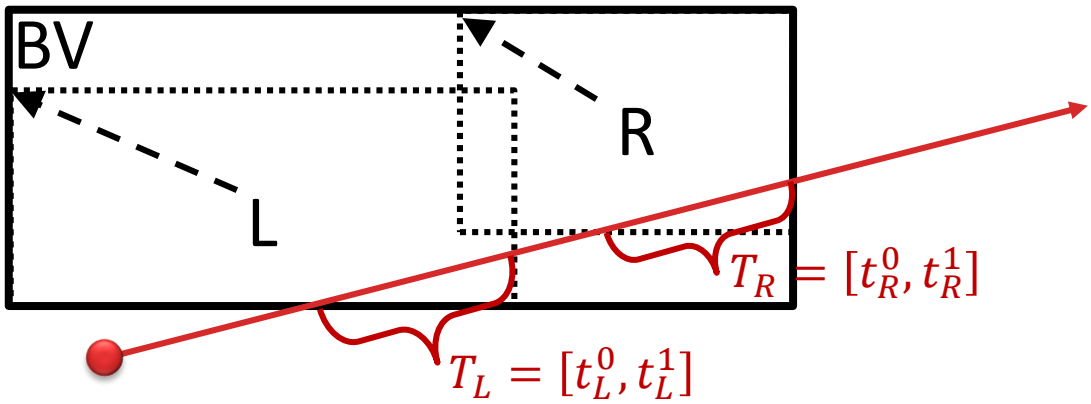



```
[found,hit,t*] find_first_hit_BVH(ray, node, t*)
```

- In rekursiver Funktion wird je nach Knotentype unterschiedlich verfahren:
- Blattknotten: Schnittberechnung mit allen Primitiven des Blatts (Implementierung wie bei nicht beschleunigten Fall)



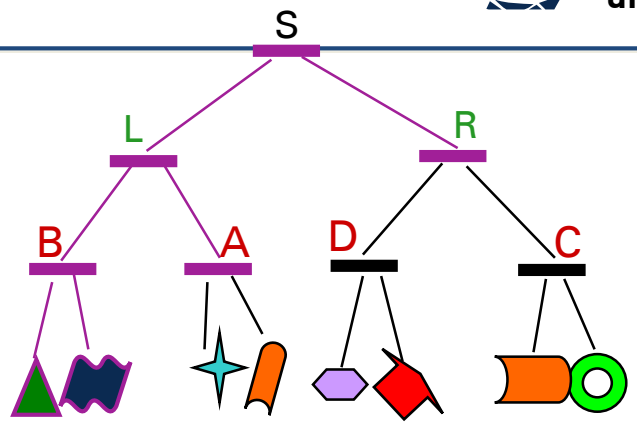
Hüllvolumenhierarchie (BVH)
Strahlschnitt – Innerer Knoten

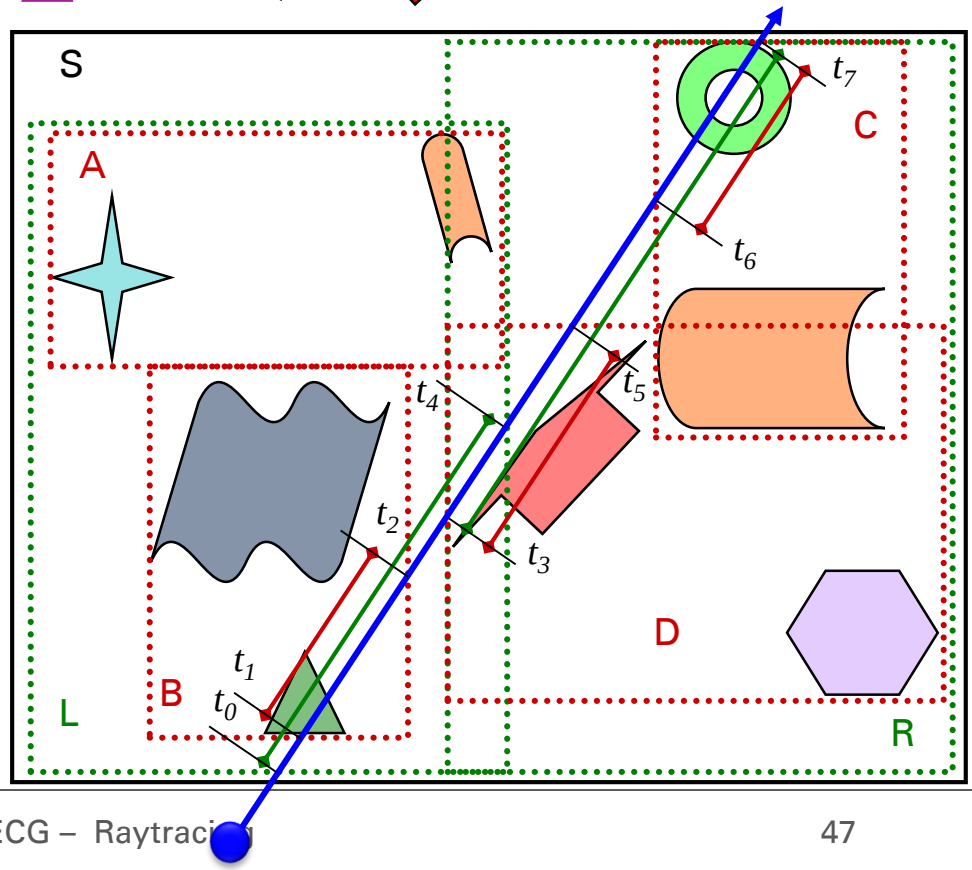


- Schnitt mit `L->BV` und `R->BV` ergibt Schnittparameterintervalle $T_L = [t_L^0, t_L^1]$ und $T_R = [t_R^0, t_R^1]$, wobei Eintrittszeiten $t_{L|R}^0 \geq 0$ garantiert werden
- Kindknoten ohne besseren Schnitt ($T = \emptyset$ oder $t^0 > t^*$) werden direkt verworfen
- zuerst Rekursion auf Kindknoten mit kleinerer Eintrittszeit, dabei Update von t^*
- Rekursion auf zweitem Kindknoten nur, falls Eintrittszeit kleiner neuem t^*

Hüllvolumenhierarchie (BVH) Strahlschnitt – Beispiel

- Im abgebildeten Beispiel werden beim BVH Strahlschnitt nur Schnitte mit den BVs S,L,R,A und B berechnet sowie mit dem grünen Dreieck, wo der erste Schnitt auftritt, und dem Wellblech:





Schnitt mit S gefunden

find_first_hit_BVH(ray,S,∞)

$T_L = [t_0, t_4]$, $T_R = [t_3, t_{t_7}]$

find_first_hit_BVH(ray,L,∞)

$T_A = \emptyset$, $T_B = [t_1, t_2]$

find_first_hit_BVH(ray,B,∞)

Schnitt Dreieck ergibt t_1

return $t^* = t_1$

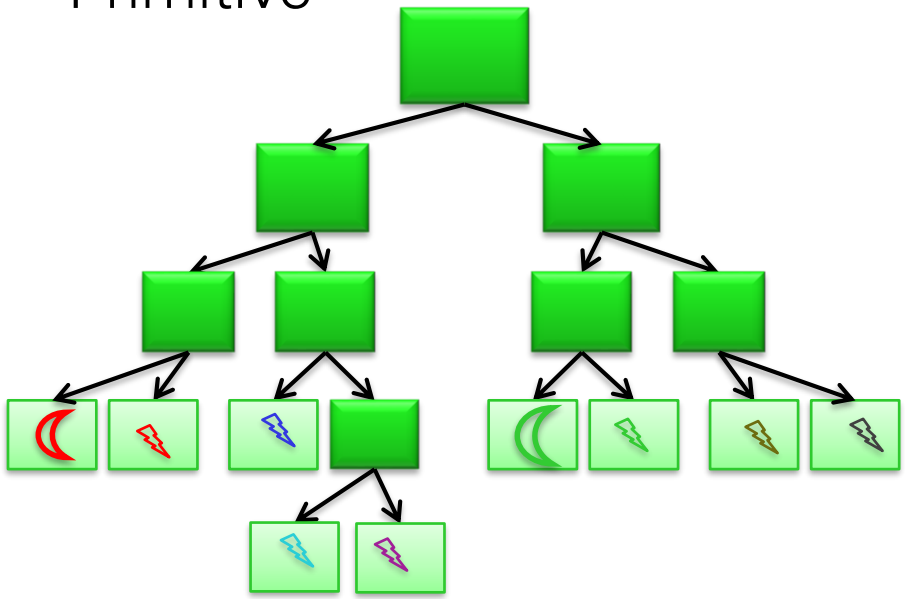
return $t^* = t_1$

R verwerfen $(t_R^0 = t_3 > t^* = t_1)$

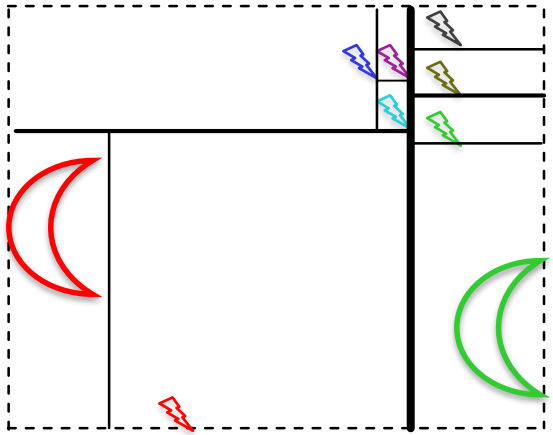
return $t^* = t_1$

KD-Baum
Definition

- Ein KD-Baum (KD steht für k-dimensional) ist ein Binärbaum, der einen k-dimensionalen (bei uns k=3) Raum an achsenparallelen Ebenen unterteilt
- In jedem inneren Knoten wird eine Trennebene gespeichert und in den Blattknoten ein oder mehrere Primitive



hierarchy view on kd tree

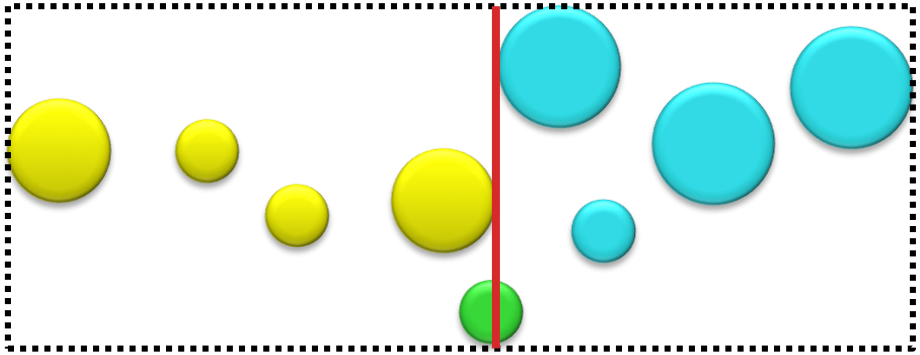


spatial view on kd tree

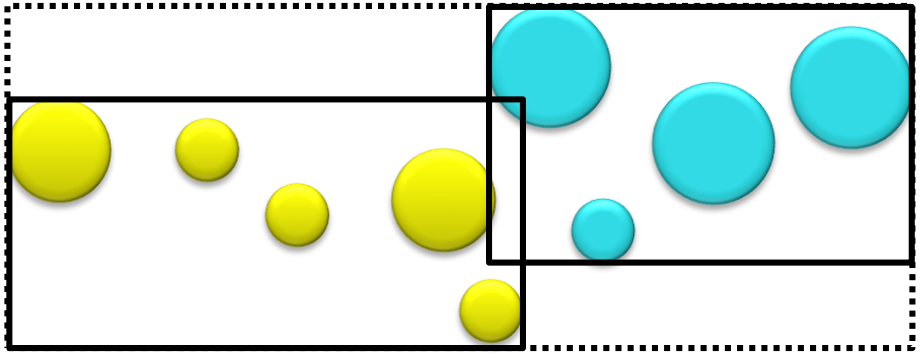
KD-Baum
Aufbau mittels Raumteilung

Computergraphik und Visualisierung

- Raumteilung ergibt 2 BVs; Optimierung wieder über SAH
- Beim Aufteilen des Raums können Primitive (siehe grüner Kreis in linker Abbildung) von Trennebene geschnitten werden.
- Umgang mit geschnittenen Primitiven:
  - referenzieren in beiden Kindknoten (Anwendung: Ray-Tracing)
  - spalten des Primitives (Anwendung: transparente Darstellung)
  - speichern im inneren Knoten (Anwendung: Modellierung)



Raumunterteilung bei KD-Baum

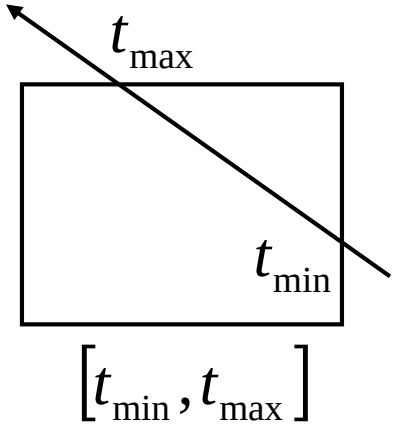


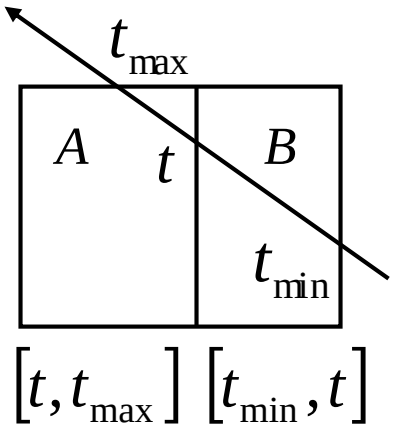
Aufteilen der Primitive bei BVH

KD-Baum
Traversierung beim Schnitttest

- Bei der Traversierung müssen die Blattknoten nach aufsteigend sortiertem Strahlparameter geprüft werden, bis der erste Schnitt gefunden ist:

1. Anfangs wird das Schnittintervall mit der Szenen-AABB berechnet und auf Abbruch geprüft.
2. In den Blättern werden Tests mit den Primitiven durchgeführt und die Mailboxtechnik angewandt
3. In inneren Knoten wird der Schnittparameter der Trennfläche berechnet, um das aktuelle Parameterintervall in zwei Teile zu zerlegen.
4. Der vom Strahl zuerst durchwanderte Kindknoten wird zuerst rekursive traversiert.





5. Nur wenn im zuerst traversierten Teilbaum kein Schnitt vor der Trennebene gefunden wurde, wird auch der zweite Kindknoten rekursive traversiert.

Tube Audio Amplifier

Pebbles Beach

Highway


Paskire
Bire-Bire
Bodabida


NATIONAL
HIGHWAYS
AUTHORITY
OF INDIA
SLOW
TOLL PLAZA AHEAD
1500 Mtr.

Dolphine[s]

Bonsai