

Datenstrukturen

Einführung in die Computergrafik
(für Augmented Reality)

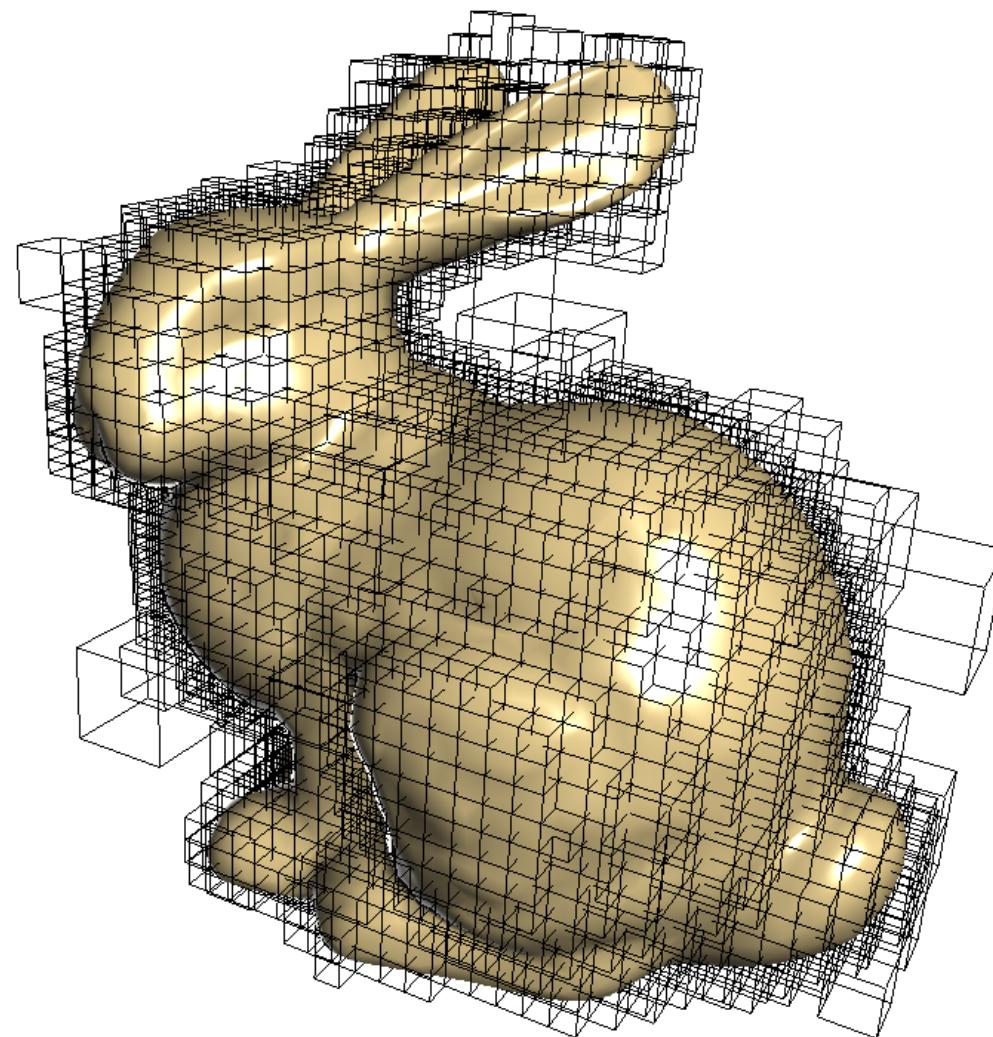
Änderungshistorie

- 22.11.2018
 - auf Folie 20 waren der IF- und ELSE-Zweig vertauscht

Wiederholung

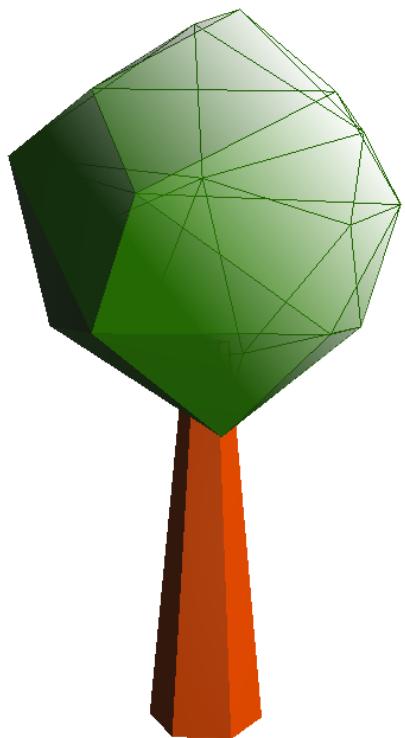
- Interpolation
- Parametrisierung & Basisfunktionen
- Kurventypen
 - Hermite
 - Bezier
- Splines
- Parametrisierte Flächen

Motivation



Agenda

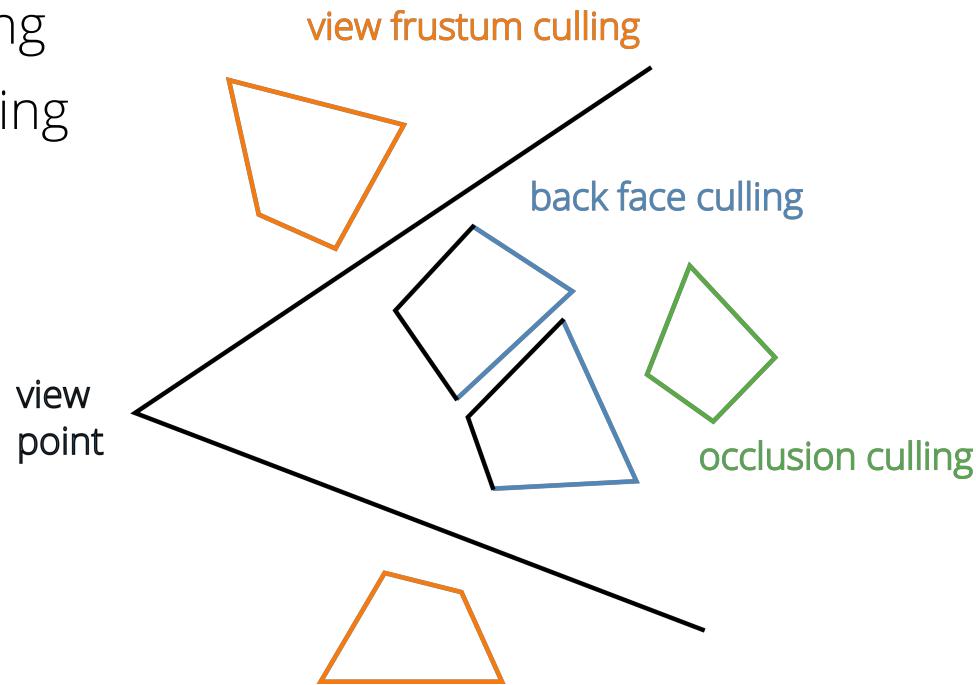
- Sichtbarkeit
- Hüllkörper
- Hüllkörperhierarchien
 - Octree
 - Binary Space Partition



Sichtbarkeit

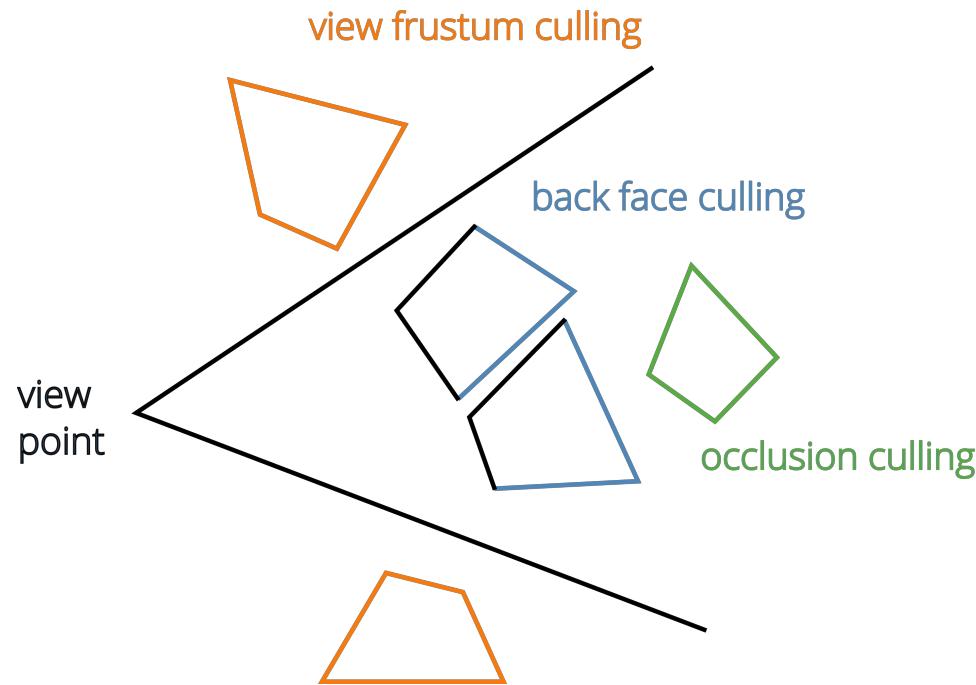
Culling

- Ignorieren von Objekten, die vom Beobachter (virtuelle Kamera) aus nicht sichtbar sind.
 - View-Frustum-Culling
 - Backface-Culling
 - Occlusion-Culling



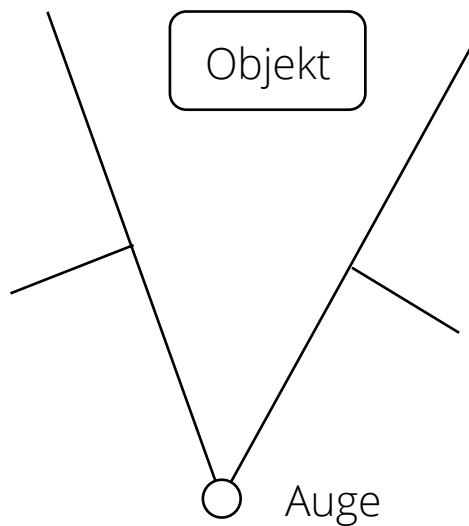
View-Frustum-Culling

- Objekte außerhalb des Sichtkegels
- Schnitt mit Sichtbarkeitsvolumen

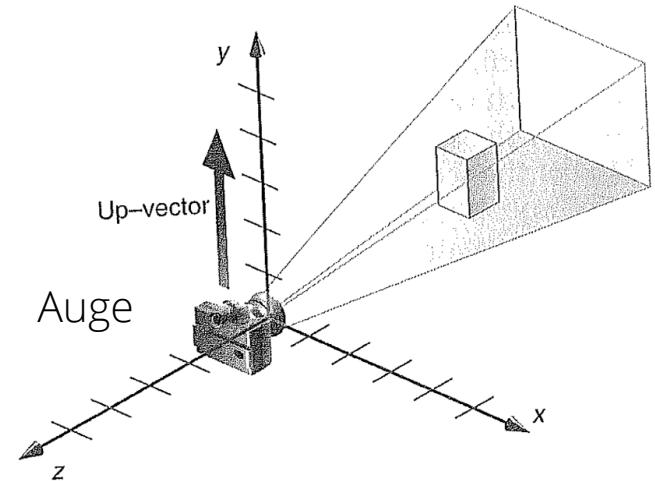


Berechnung

- Darstellung durch Begrenzungsebenen
 - 2D: 2 Linien mit Normalen + 2 für near-/far-clipping-Ebene
 - 3D: 4 Ebenen + 2 für near-/far-clipping-Ebene



2D Sichtvolumen



3D Sichtvolumen

2D Beispiel

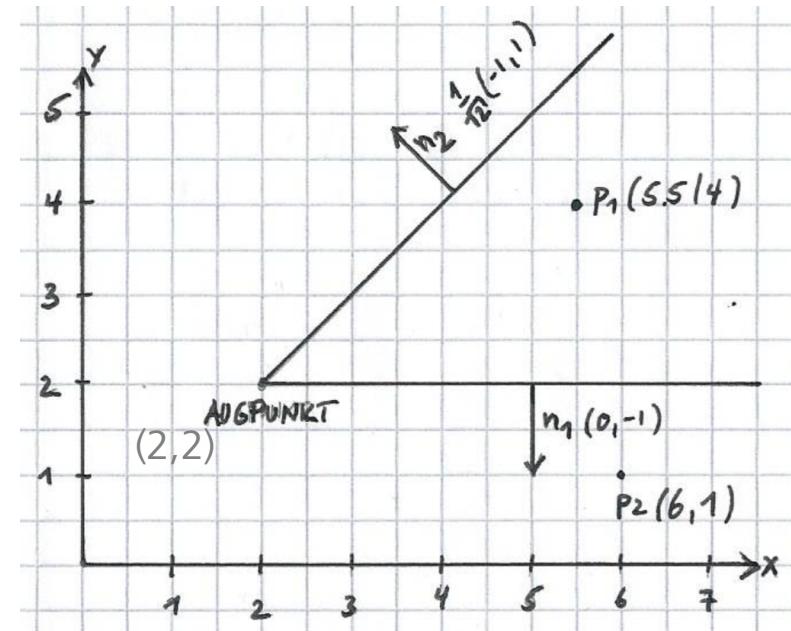
$$E_1 : (\vec{x} \cdot \begin{pmatrix} 0 \\ -1 \end{pmatrix} - \begin{pmatrix} 2 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ -1 \end{pmatrix}) = 0$$

$$E_1 : \vec{x} \cdot \begin{pmatrix} 0 \\ -1 \end{pmatrix} + 2 = 0$$

$$E_2 : \frac{1}{\sqrt{2}} \vec{x} \cdot \begin{pmatrix} -1 \\ 1 \end{pmatrix} = 0$$

$$E_1(p_1) : -4 + 2 = \boxed{-2}$$

$$E_2(p_1) : \frac{-5.5 + 4}{\sqrt{2}} = \frac{-1.5}{\sqrt{2}} \approx \boxed{-1.06}$$



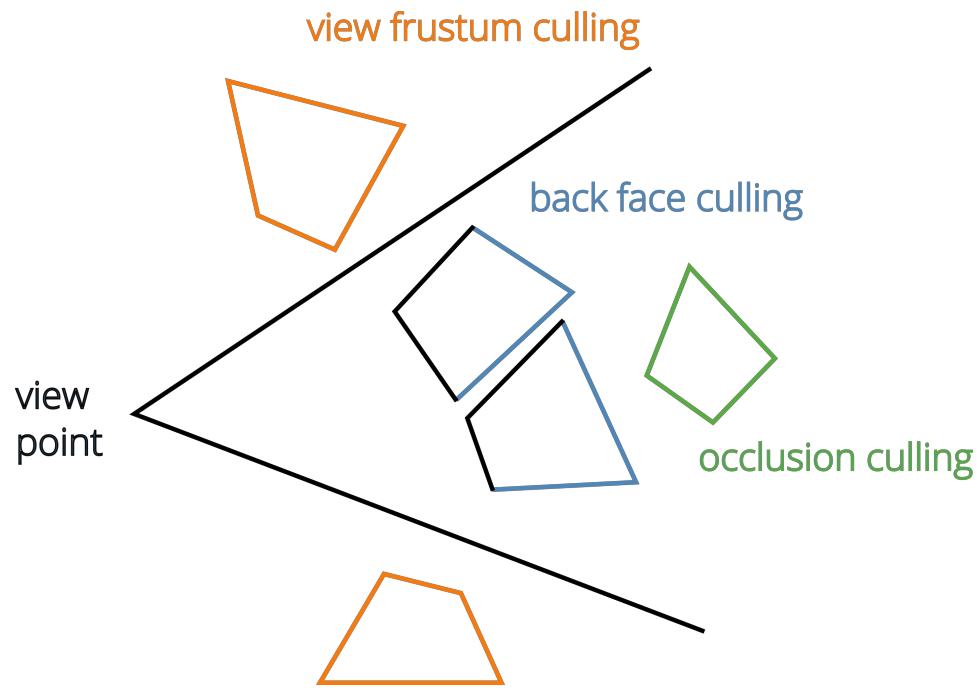
$$E_1(p_2) : -1 + 2 = \boxed{1}$$

$$E_2(p_2) : \frac{-6 + 1}{\sqrt{2}} = \frac{-5}{\sqrt{2}} \approx \boxed{-3.54}$$

eine Entfernung positiv, daher
außerhalb

Backface-Culling

- Ignorieren von Facetten, die vom Beobachter wegzeigen



Berechnung

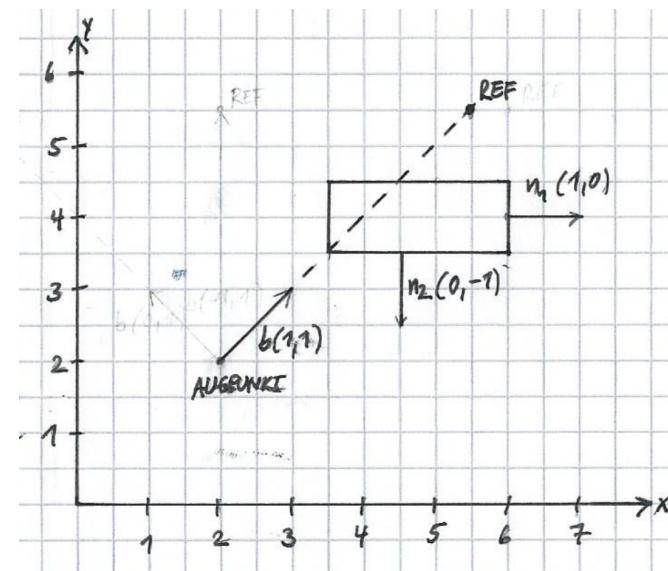
- Blickrichtung vs. Facettennormale
 - Skalarprodukt

$$b \cdot n_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1 > 0$$

ignoriert

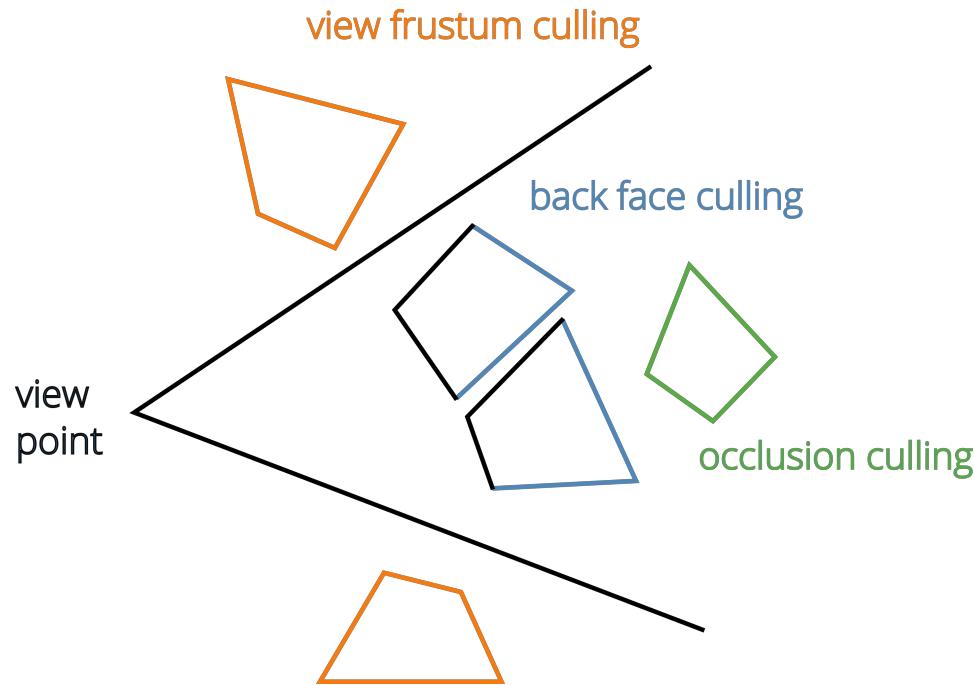
$$b \cdot n_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -1 < 0$$

gezeichnet



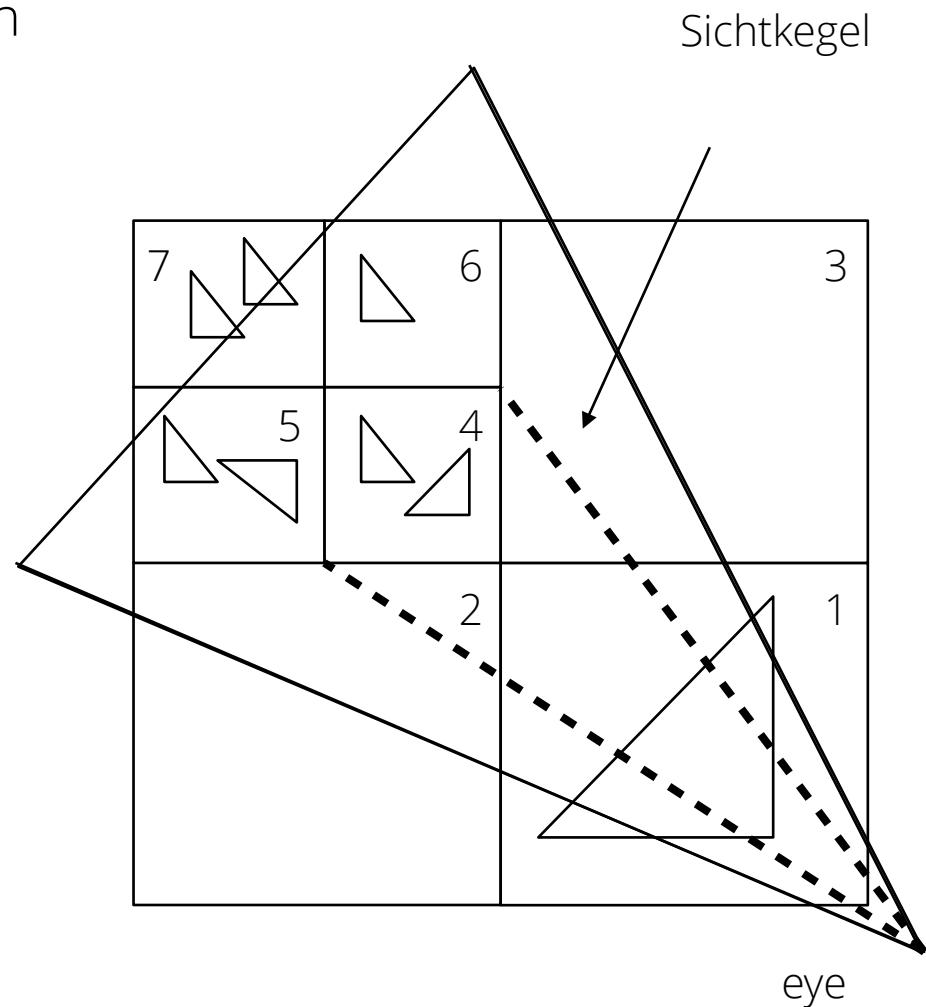
Occlusion-Culling

- Ignorieren von Objekten „hinter“ anderen Objekten
 - verdeckte Objekte: occludees
 - verdeckende Objekte: occluders



Occlusion-Culling

- Verwenden von hierarchischen Datenstrukturen
- Idee
 - Zelle 4 ist unsichtbar wegen Objekt in Zelle 1: gestrichelte Linien
 - Objekte in Zelle 4 müssen nicht gezeichnet werden



Portal Culling

- Idee: verschiedene Anwendungen „im Inneren“
- einzelne Öffnungen zwischen Räumen (z.B. Türen, Fenster) = Portale
- Idee: Portale schränken das Sichtbarkeitsvolumen ein

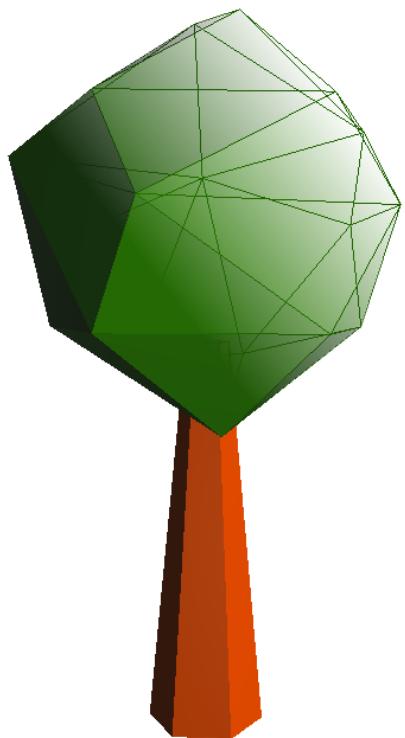


Grundriss eines Gebäudes mit
Türen und Fenstern



gerendertes Bild

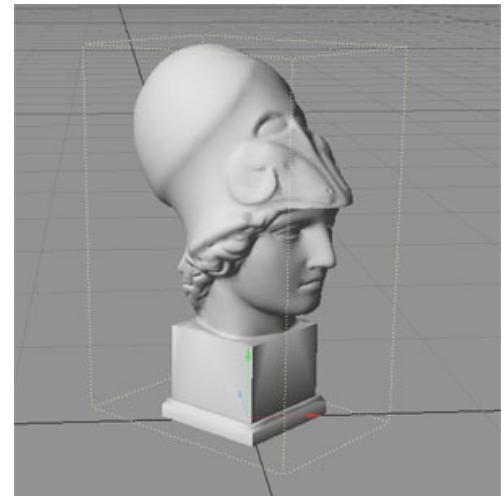
Quelle: [1]



Hüllkörper

Hüllkörper

- „einfacher“ geometrischer Körper
- umschliesst komplexes geometrisches Objekt
- Beispiele
 - Kugel
 - Würfel, Quader
 - Hyperebenen
 - Polyhedra



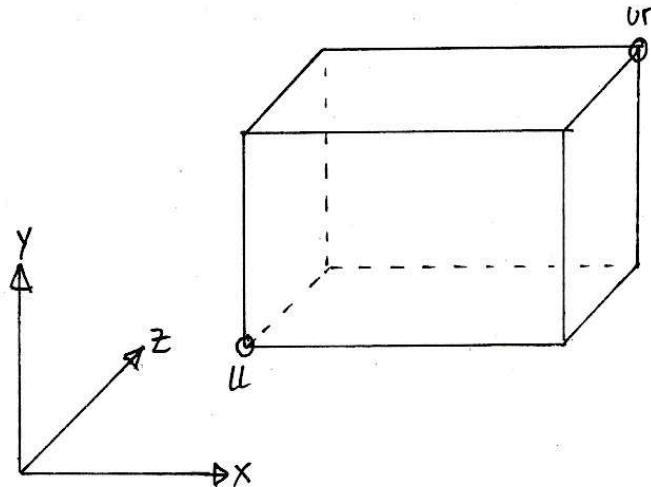
Quader als Hüllkörper
(Quelle: Wikipedia Hüllkörper:
<http://de.wikipedia.org/wiki/H%C3%BCllk%C3%B6rper>,
07.12.2013)

Hüllkörper

- Hüllkörper repräsentiert Objekt normalerweise nicht exakt
- daher: zwei Überprüfungen notwendig
 - 1) Schnittberechnung mit Hüllkörper
 - IF 1) == JA THEN 2) Schnittberechnung mit Objekt selbst
- Warum Überprüfung 1)? Performance!
 - Schnitttest mit Hüllkörper üblicherweise deutlich schneller

Würfel

- häufige Einschränkungen: Kanten parallel zu Hauptachsen
 - axis-aligned-bounding-box (AABB)
 - Repräsentation:
 - Eckpunkt mit kleinen Koordinaten: lower left, ll
 - Eckpunkt mit den größten Koordinaten: upper right, ur

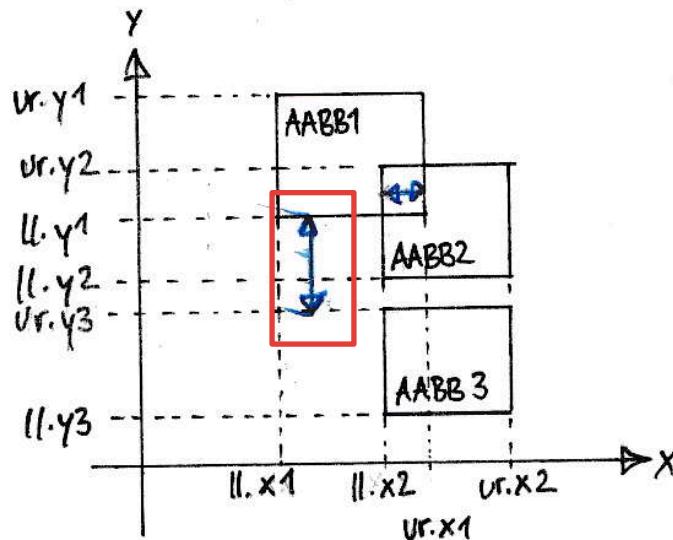


AABB-Schnitt

- Test, ob zwei AABBs überlappen
- Idee: Test für jede Achse
 - Schnitt nur dann, wenn es einen Überlapp entlang jeder Achse gibt

```
FOR jede Achse DO
    IF ur2 > ll1 UND ur1 > ll2
        weiter mit nächster Achse
    ELSE
        kein Schnitt
    RETURN NEIN
END ELSE
END FOR
RETURN JA
```

2D Beispiel: AABB Schnitt



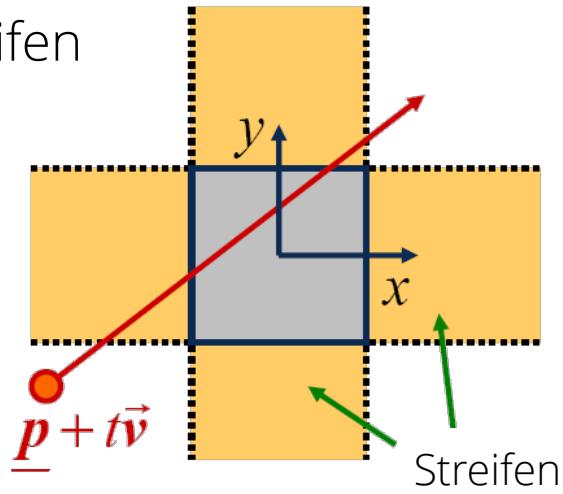
- AABB1 vs. AABB2 (Schnitt)
 - x-Achse: $ur_2 > ll_1 \text{ AND } ur_1 > ll_2$ (Überlapp)
 - y-Achse: $ur_2 < ll_1 \text{ AND } ur_1 > ll_2$ (Überlapp)
- AABB1 vs. AABB3 (kein Schnitt)
 - x-Achse: $ur_3 > ll_1 \text{ AND } ur_1 > ll_3$ (Überlapp)
 - y-Achse: **$ur_3 < ll_1$** , $ur_1 > ll_3$ (kein Überlapp)

Übung: Strahl-Ebenen-Schnitt

- gegeben
 - Ebene E: Punkt $p = (1,3)$, Normale $n = (0,1)$
 - Strahl R: $(1,0) + \lambda (2/3, 1)$
- gesucht:
 - Schnittpunkt x

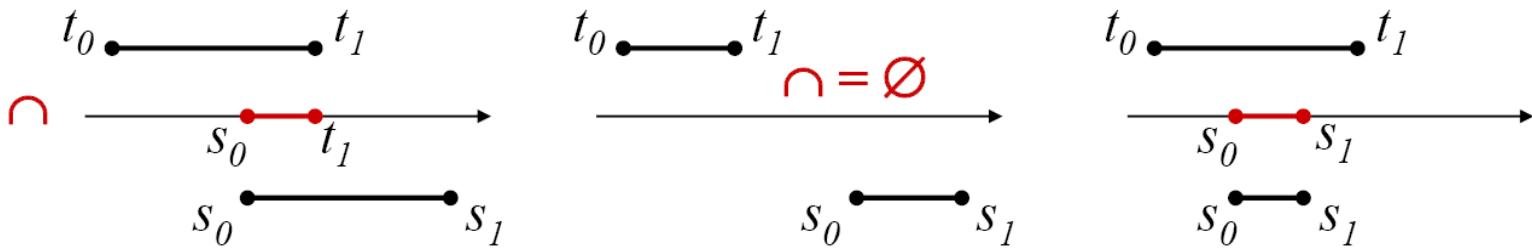
Strahl-Würfel-Schnitt

- Würfel ist konvex
 - maximal zwei Schnittpunkte: Eintritt, Austritt
- parallele Randebene definieren einen Streifen
- also: 3 Streifen (x-, y-, z-direction)
 - 2 Streifen im 2D
 - Schnitt der Streifen = Würfel
- Idee:
 - Schnitt mit Streifen
 - dabei: Schnitt mit Randebenen (-linien im 2D)
 - Schnitt mit Würfel = Schnittintervall der Zwischenergebnisse



Strahl-Würfel-Schnitt

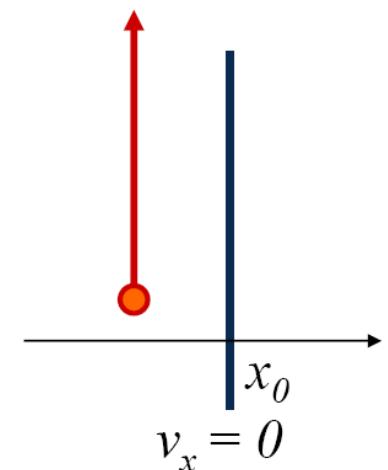
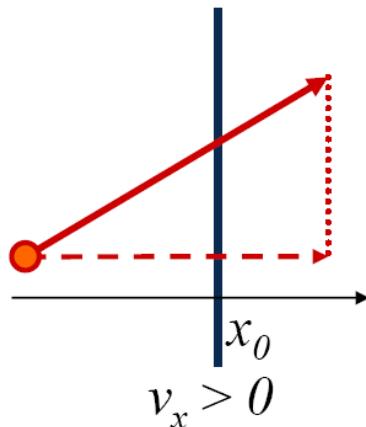
- Intervallarithmetik
- Reduktion auf den 1D-Fall
- Intervall-Beispiel
 - $T = [t_0, t_1]$ mit $t_0 < t_1$ oder $T = \{\}$ (leeres Intervall)
- Schnitttypen:



Strahl-Würfel-Schnitt

- Schnitt mit x-Ebene $x = x_0$
 - Strahl startet bei $x(t = 0) = p_x$
 - x-Komponente wächst mit „Geschwindigkeit“ v_x
 - aus $x_0 = p_x + t_0 v_x$ folgt für t_0 :

$$t_0 = \begin{cases} \frac{x_0 - p_x}{v_x} : v_x \neq 0 \\ \text{undefined} : v_x = 0 \end{cases}$$



Beispiel

- Strahl $s(t)$:

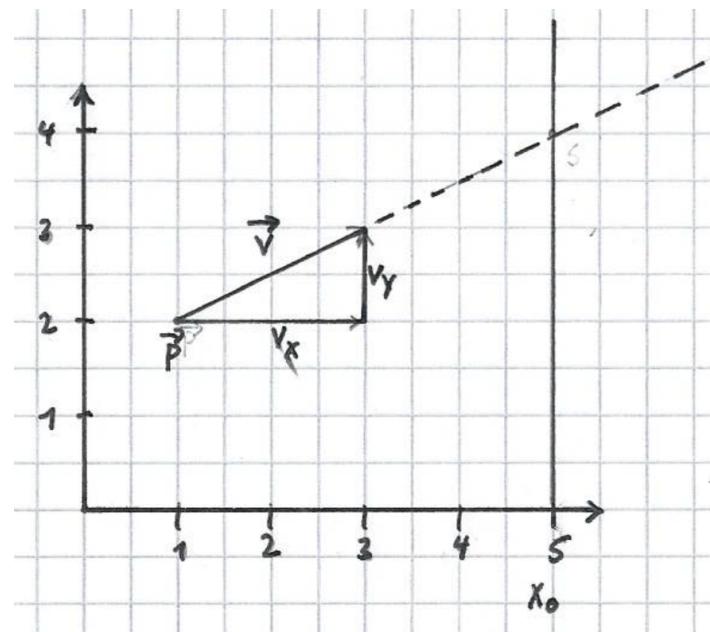
$$s(t) = \vec{p} + t\vec{v} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} + t \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

- Schnittebene: $x_0 = 5$

$$t_0 = \begin{cases} \frac{x_0 - p_x}{v_x} : v_x \neq 0 \\ \text{undef} : v_x = 0 \end{cases}$$

$$t_0 = \frac{5-1}{2} = 2$$

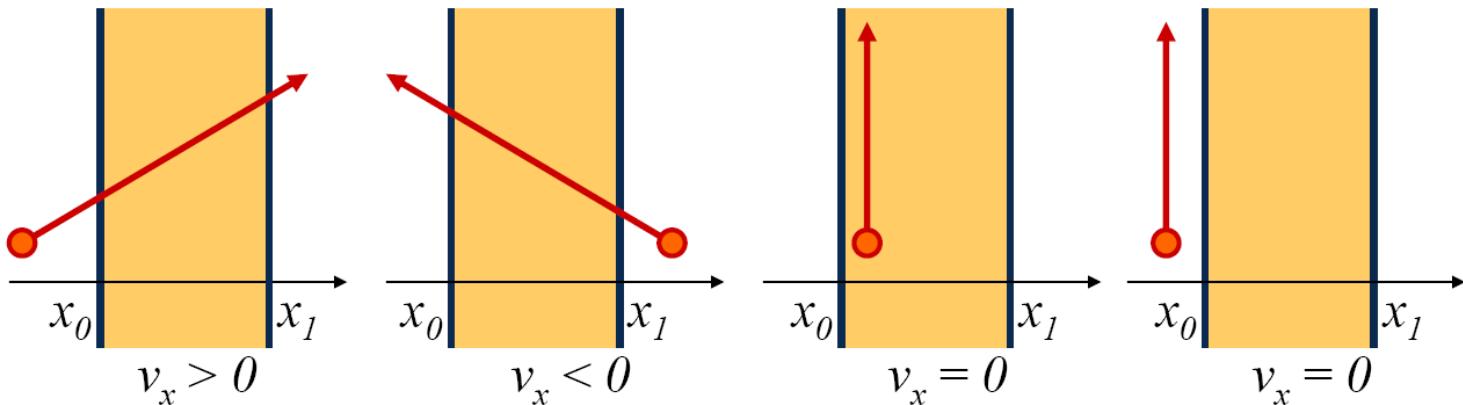
$$s(2) = \begin{pmatrix} 1 \\ 2 \end{pmatrix} + 2 \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \end{pmatrix}$$



Strahl-Würfel-Schnitt

- Schnitt mit Streifen $x \in [x_0, x_1]$
 - abhängig vom Vorzeichen von v_x
 - Spezialfall: Ebene parallel zu Strahl

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

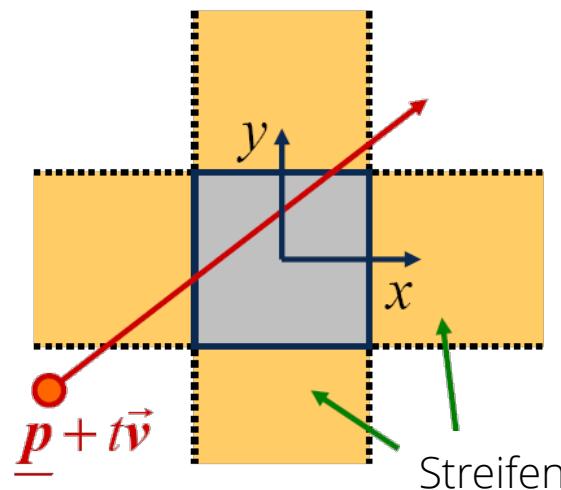


Strahl-Würfel-Schnitt

- bestimme Schnittintervalle $T_{x/y/z}$
- Schnitt der drei Schnittintervalle T_w

$$T_w = T_x \cap T_y \cap T_z$$

- Abschluss:
 - gibt es ein $t \in T_w$ mit $t > 0$



Übung: Strahl-Würfel-Schnitt

- Bestimmen Sie den Schnitt zwischen dem Strahl s mit

$$s = \vec{p} + t\vec{v} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} + t \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

- und der Box mit den Eckpunkten (2,2) und (3,4)

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

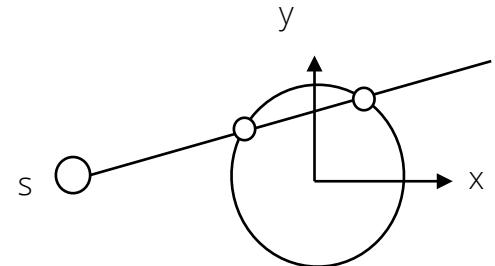
Kugel

- Schnittergebnis: quadratische Gleichung mit 0,1 oder 2 Lösungen
 - Kugel K

$$K = \{x \in R^3 \mid \| \vec{x} - \vec{m} \|^2 - r^2 = 0\}$$

- Strahl s

$$s : \vec{p} + \lambda \vec{v}$$



- Strahl in Kugelgleichung

$$\| \vec{p} + \lambda \vec{v} - \vec{m} \|^2 - r^2 = 0$$

Kugel: Innen-Test

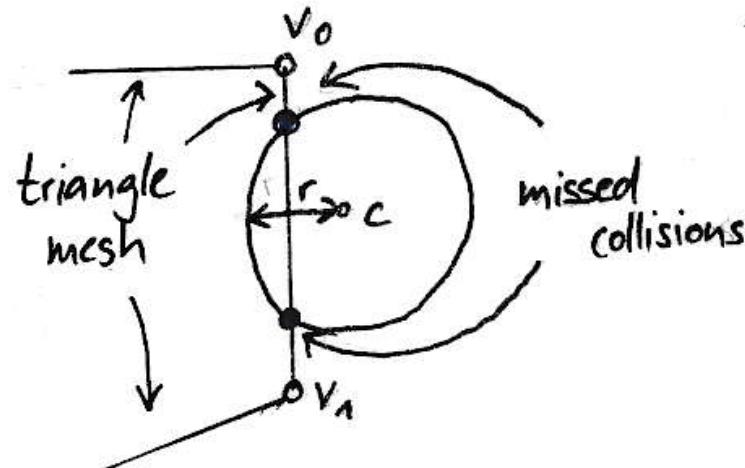
- Aufgabe
 - prüfe, ob ein Punkt p liegt innerhalb einer Kugel S mit Mittelpunkt c und Radius r
- Idee:
 - berechne quadratischen Abstand zwischen p und c
 - vergleiche mit quadriertem Radius r

$$||p - c||^2 < r^2$$

Kugel-Dreiecksnetz - Approximation

- Idee:

```
for v in vertices
    if collides(sphere, v) return true
return false
```
- funktioniert gut für hochauflöste Dreiecksnetze
 - kleine Dreiecke im Vergleich zum Kugelradius
- Fehler (verpasste Kollisionen) bei großen Dreiecken



Kugel-Dreiecksnetz - Exakt

- Idee:

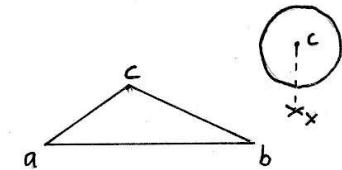
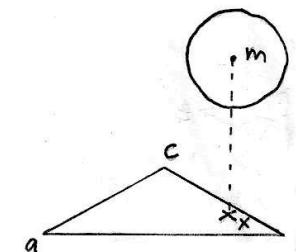
```
for triangle in triangles
    d = distance(triangle, v)
    if d < radius
        return true
return false
```

- Wie berechnet man `distance(triangle, v)`?

Entfernung Dreieck-Punkt

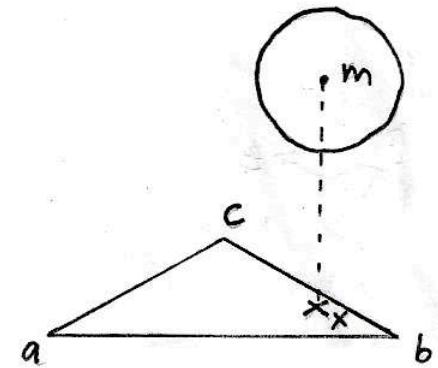
- Dreieck $t = (a,b,c)$, Punkt m
- Punkt ist entweder näher an
 - Punkt auf einer Kante
 - Punkt im Inneren
- x = Projektion von m Auf Ebene abc
- $\text{distance}(\text{edge}, m)$ // siehe Skript: Skelettautomation

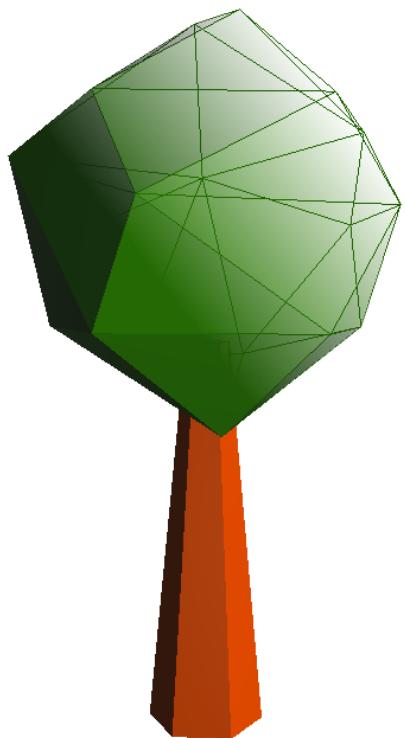
```
d_ab = segment_distance(ab,m)
d_bc = segment_distance(bc,m)
d_ca = segment_distance(ca,m)
x = projection of m onto plane abc
if x in abc
    return ||x - m||
else
    return min(d_ab, d_bc, d_ca)
```



Entfernung Dreieck-Punkt

- Projektion: m auf Ebene abc
 - Ebenennormale: $n = \text{normalize}((b-a) \times (c-a))$
 - Ebene P : $p \cdot n - a \cdot n = 0$
 - Strahl R : $m + \lambda n$
 - Einfügen: R in P : $(m + \lambda n) \cdot n - a \cdot n = 0$
 - Auflösen nach λ
 - projizierter Punkt $x = m + \lambda n$
- Innen-Test
 - bestimme Baryzentrische Koordinaten α, β, γ
 - falls α, β, γ all in $[0,1]$, dann innen
 - ansonsten außen

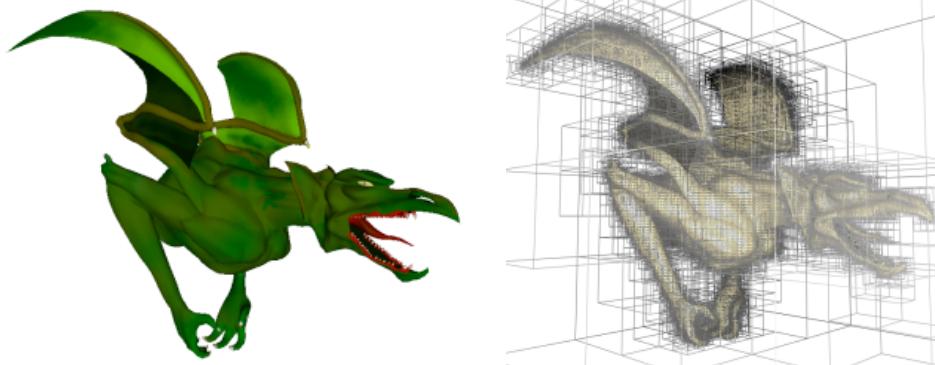




Octree

Octree

- Lateinisch „octo“ = 8
- jeder Knoten (= Würfel) hat ...
 - keine Kinder
 - oder 8 Kindknoten
- Wurzelknoten-Würfel umschliesst gesamte Szene

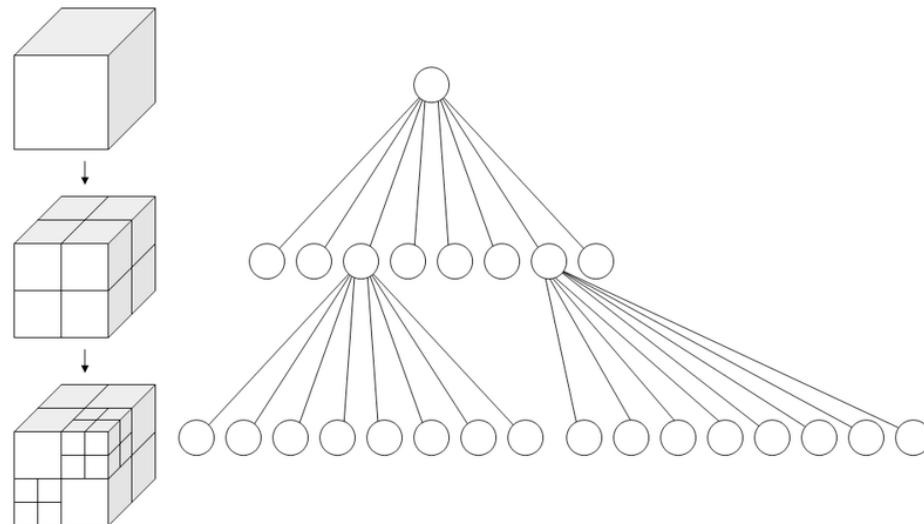


3D-Modell und Octree-Würfel (gezeigt sind nur

Prof. Dr. Philipp Jenke, Department Informatik Knoten, die Elemente beinhalten)

Konstruktion

- erstelle Wurzelknoten
- solange die Hierarchie noch zu grob ist
 - unterteile Blattknoten in 8 Kindknoten



Octree -Erzeugung (Bildquelle: Wikipedia Octree: <http://de.wikipedia.org/wiki/Octree>, 07.12.2013)

Unterteilung

- Kriterien (u.a.)
 - maximale Tiefe des Baumes
 - Maximale Anzahl von Elementen in Blattknoten
- Achtung: ein Objekt (z.B. Dreieck) kann in mehreren Knoten enthalten sein

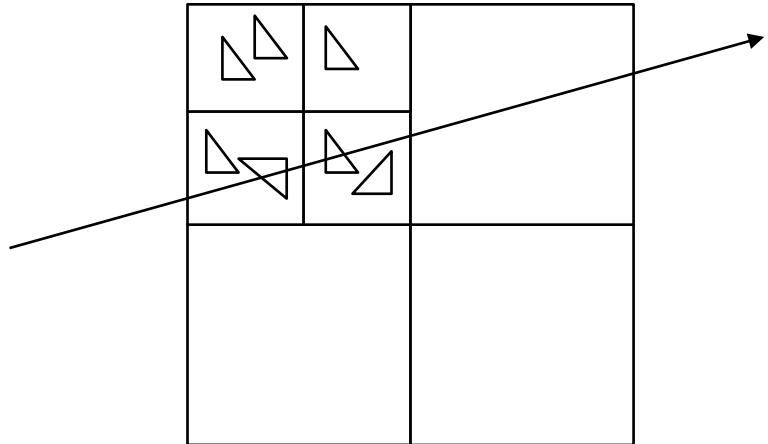
Anwendung: Strahl-Dreieck-Schnitt

- Beispiel
 - Octree für Dreiecke
 - Aufgabe: Finde alle Dreiecke, die durch einen Strahl geschnitten werden
- naive Lösung
 - Schnitt des Strahls mit allen Dreiecken
 - $O(n)$ -Komplexität ($n = \text{Anzahl Dreiecke}$) für einen Strahl

Anwendung: Strahl-Dreieck-Schnitt

- Pseudocode:

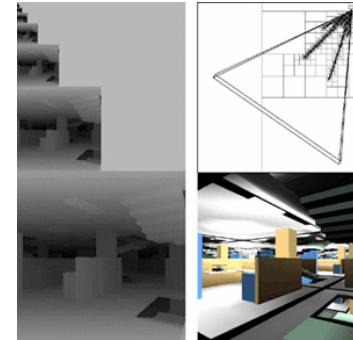
```
function traverse(ray, box)
    if (!intersect(ray, box))
        return
    for each triangle in box
        intersect(ray, triangle)
    for each childbox
        traverse(ray, childbox)
```



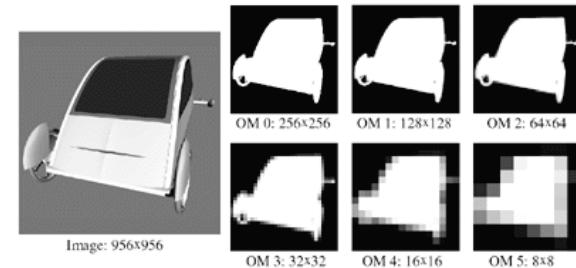
- Hinweis: Octree im 2D ist ein Quadtree
 - 4 Kind-Quadrate für jeden inneren Knoten

Occlusion-Culling: Ansätze

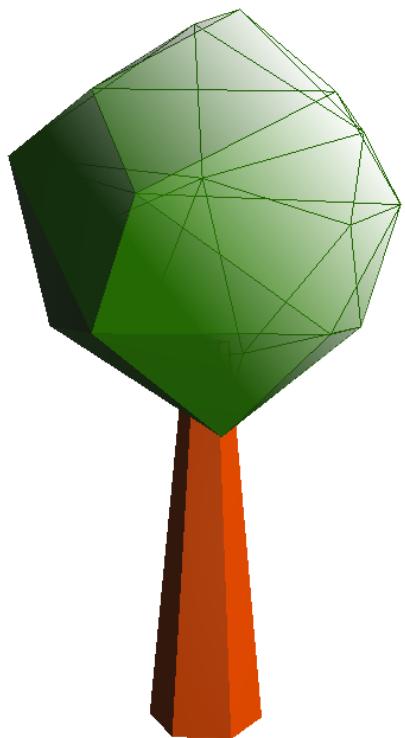
- Hierarchischer Tiefenpuffer [2]



- Hierarchische Occlusion-Map [3]



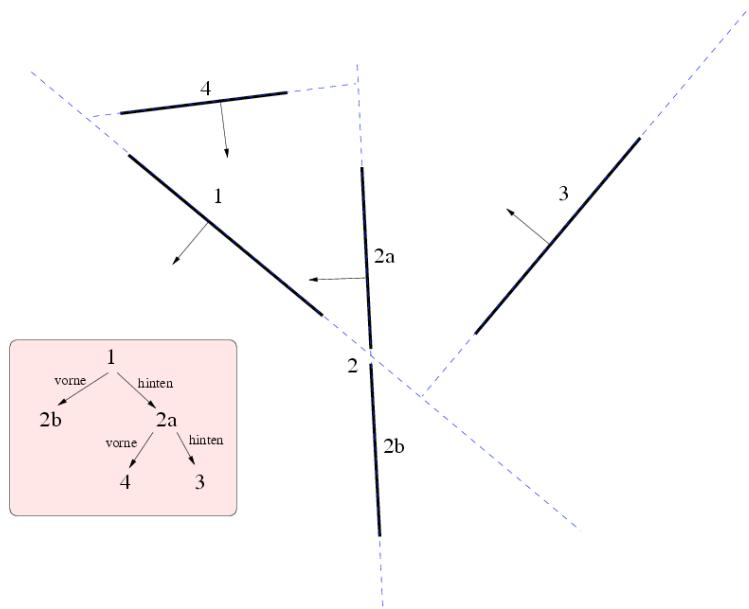
- viele weitere ...



Binary Space Partition

Binary Space Partition (BSP)

- Raumunterteilung mit Hyperebenen
 - 2D: Linie mit orthogonalem Normalenvektor
 - 3D: Ebene mit Normale
 - vorne (in Normalenrichtung) vs. hinten
- Idee:
 - rekursive Unterteilung des Raumes
 - z.B. bis jeder Unterraum maximal ein Objekt beinhaltet (oder wenige)

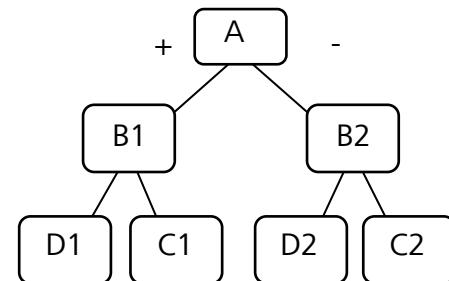
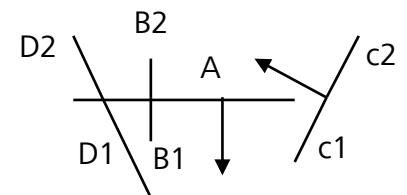
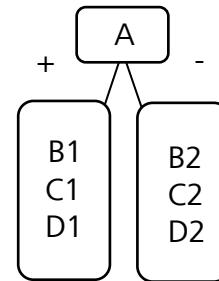
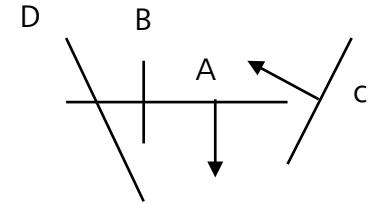
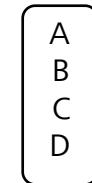


BSP tree

Bildquelle: Wikipedia Binary Space Partitioning, http://de.wikipedia.org/wiki/Binary_Space_Partitioning, abgerufen am 7.12.2013

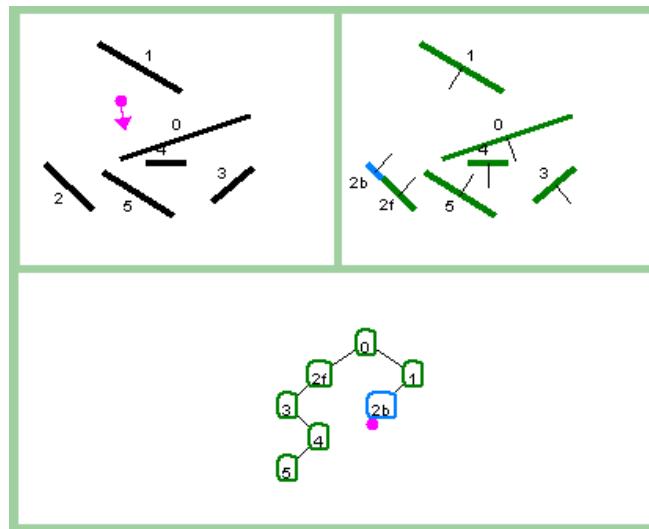
Konstruktion

- Start mit beliebiger Hyperebene h
 - +: alle anderen Hyperebenen „vor“ h : linker Teilbaum
 - -: alle Hyperebenen „hinter“ h : rechter Teilbaum
 - falls notwendig: zusätzliche Hyperebenen einfügen
- Ziel: balancierter Baum



Binary Space Partition (BSP)

- Demo: <http://www.symbolcraft.com/products/bsptrees/>



Interaktive Demo für BSP-Bäume

Quelle: Interactive demo of BSP trees: <http://www.symbolcraft.com/products/bsptrees/>, 07.12.2013

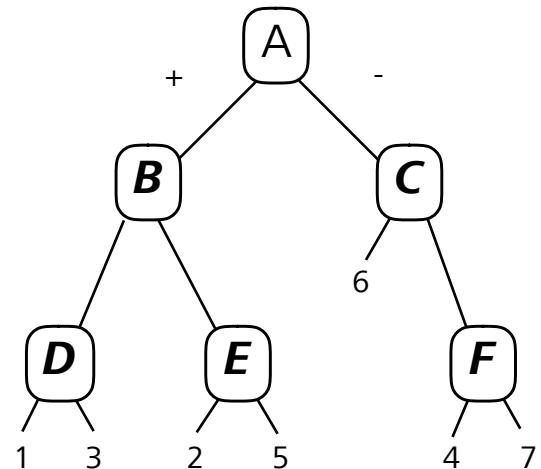
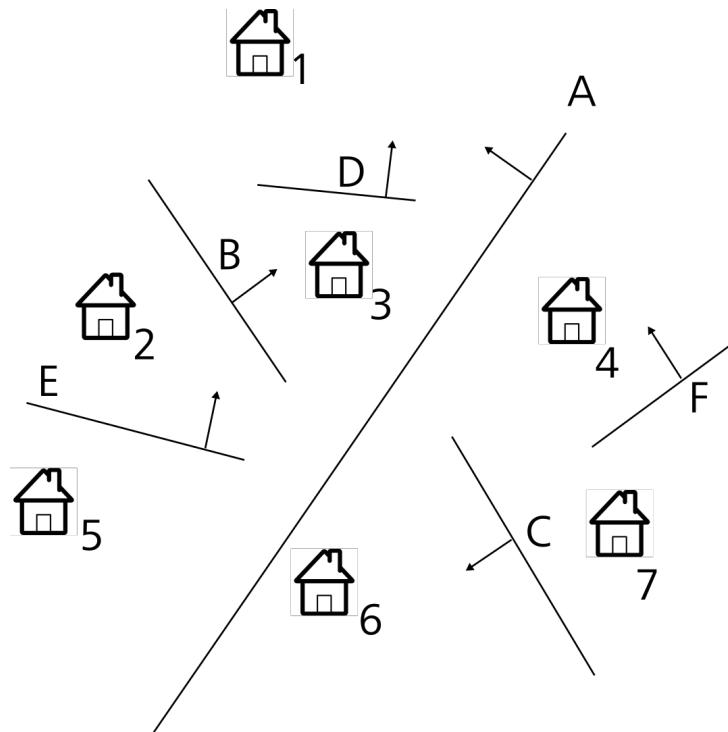
Konstruktion

- Wie sieht der BSP-Baum für die folgenden Objekte 1...7 aus?
- Was sind sinnvolle Hyperebenen?



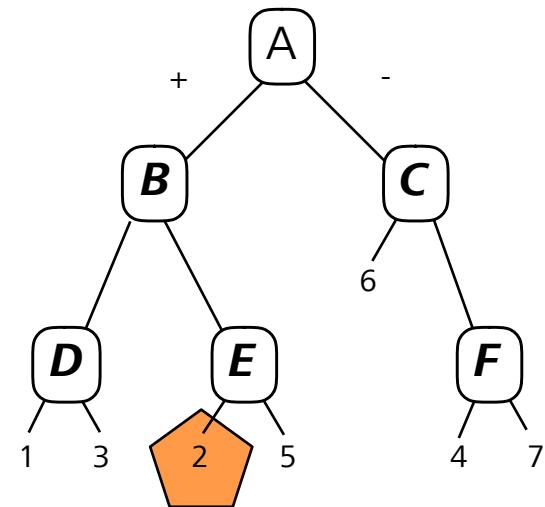
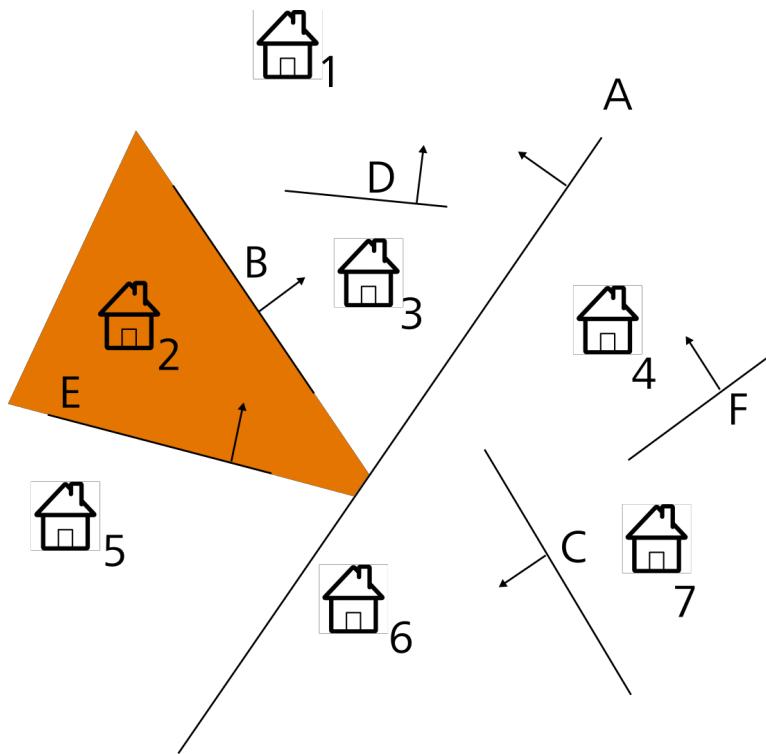
BSP-Baum

- eine mögliche (sinnvolle) Lösung



Unterräume

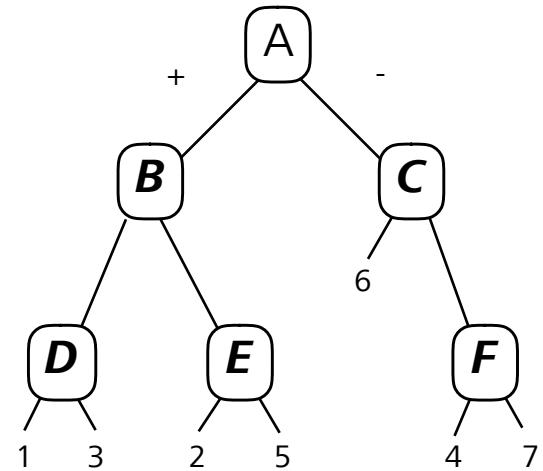
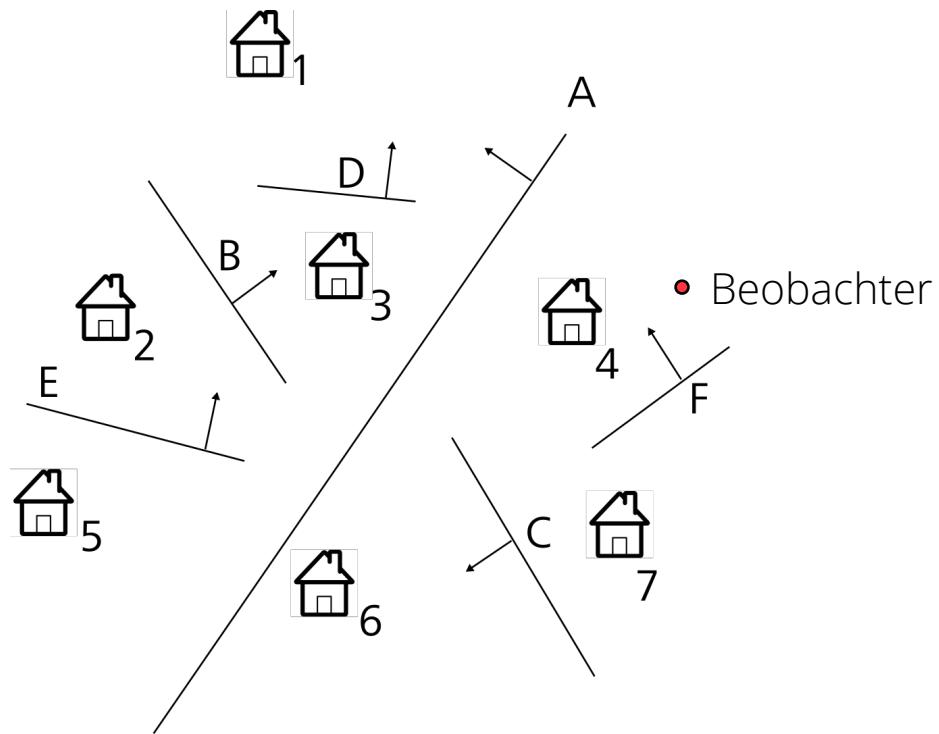
- Unterteilung schafft Unterräume
 - repräsentiert durch Blattknoten



Back-To-Front-Sortierung

- kann aus dem Baum gefolgert werden
- Algorithmus
 - Start am Wurzelknoten
 - Traversierung
 - 1) falls Position „hinter“ Hyperebene: besuche zunächst alle Kind-Hyperebenen „vorne“ (rekursiv)
 - ansonsten: besuche zunächst alle Hyperebenen „hinten“ (rekursiv)
 - 2) besuche Knoten
 - 3) besuche Rest

Übung: Back-To-Front-Sortierung



Zusammenfassung

- Sichtbarkeit
- Hüllkörper
- Hüllkörperhierarchien
 - Octree
 - Binary Space Partition

Literatur

- [1] David Luebke and Chris Georges : Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets, Proceedings of the 1995 Symposium on Interactive 3D Graphics, ACM Press, NY (April, 1995)
- [2] Greene, Ned, Michael Kass, and Gavin Miller, "Hierarchical Z-Buffer Visibility", Computer Graphics (SIGGRAPH 93 Proceedings), pp. 231-238, August 1993.
- [3] Zhang, H., D. Manocha, T. Hudson, and K.E. Hoff III, "Visibility Culling using Hierarchical Occlusion Maps", Computer Graphics (SIGGRAPH 97 Proceedings), pp. 77-88, August 1997. <http://www.cs.unc.edu/~zhangh/hom.html>