



UNIVERSITÄT  
LEIPZIG

Sichtbarkeit

# COMPUTERGRAPHIK

---

# Inhaltsverzeichnis

## 8 Sichtbarkeit

8.1 Problemstellung

8.2 Projektion

8.3 Strahlverfolgung

8.4 Raumteilung

8.5 Culling

## 8.1 PROBLEMSTELLUNG

- Sichtbarkeitsberechnung
  - Ihr Ziel ist die (möglichst) exakte Bestimmung der von einem gegebenen Blickpunkt aus sichtbaren bzw. unsichtbaren Teile der darzustellenden Szene
- Anforderungen
  - Wünschenswert ist **hohe Interaktionsrate**, so dass Eingaben des Benutzers sich direkt auf die Darstellung auswirken
  - In den meisten Fällen ist die **Echtzeitausgabe** der Szene nötig
- Einteilung der Verfahren:
  - **Objektraumverfahren**
    - prinzipiell geräteunabhängig
    - Rechengenauigkeit ist die Maschinengenauigkeit
  - **Bildraumverfahren**
    - geräteabhängig
    - Rechengenauigkeit ist die Auflösung des Ausgabegerätes

## 8.1 PROBLEMSTELLUNG

- Bestimmung der sichtbaren Objekte
  - Welches Objekt liegt dem Betrachter (Bildebene) am nächsten?
  - Notwendig für die korrekte Darstellung
  - Abhängig von der Darstellungsart
- Wireframe-Darstellung:
  - Darstellung der Begrenzungskanten von Flächen
  - Entfernung verdeckter Kanten (Hidden Line Removal, HLR)
- Flächendarstellung
  - Darstellung der Flächen
  - Hidden Surface Removal (HSR)
- Transparenzen müssen ggf. berücksichtigt werden

## 8.2 Projektion

- Dreidimensionale Szene wird auf die Bildebene projiziert
  - Unterschiedliche Objektteile werden auf dieselbe Stelle abgebildet
    - $(x, y)$ -Koordinate in der Bildebene
  - Sichtbar sind diejenigen Objektpunkte, die dem Auge des Betrachters am nächsten gelegen sind
    - Tiefenrelation der Szene ( $z$ -Koordinate)
- Beschleunigung
  - Berechnung nicht sichtbarer Teile der Szene (Culling)
  - Nutzt Heuristiken
  - In der Regel keine exakte Lösung

## 8.2 Projektion

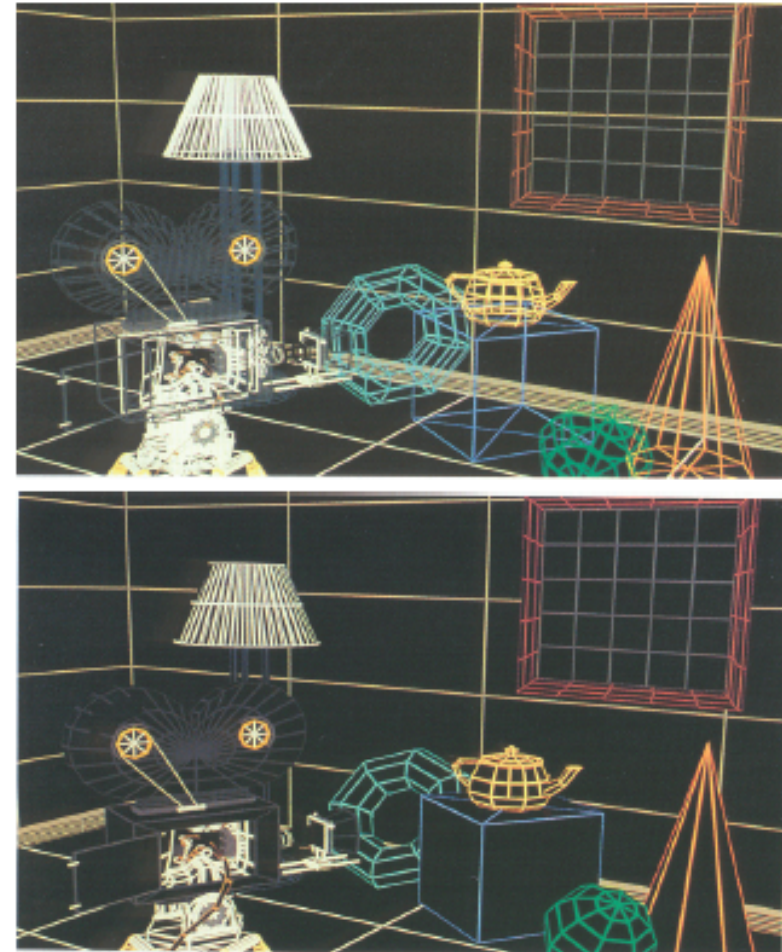
Kohärenz: Ausnutzung lokaler Ähnlichkeiten

- Objektkohärenz:  
schneiden sich zwei Objekte nicht,  
so müssen auch ihre Flächen nicht  
auf Schnitt getestet werden
- Flächenkohärenz:  
Eigenschaften benachbarter Punkte  
auf einer Fläche ändern sich oft nur  
unwesentlich
- Tiefenkohärenz:  
Die Tiefe  $z$  an der Position  $(x, y)$   
einer Fläche kann oft inkrementell  
berechnet werden
- Zeit-/Framekohärenz:  
Oft ändern sich nur wenige Anteile  
eines Frames

## 8.2 Projektion

### Wireframe-Darstellung

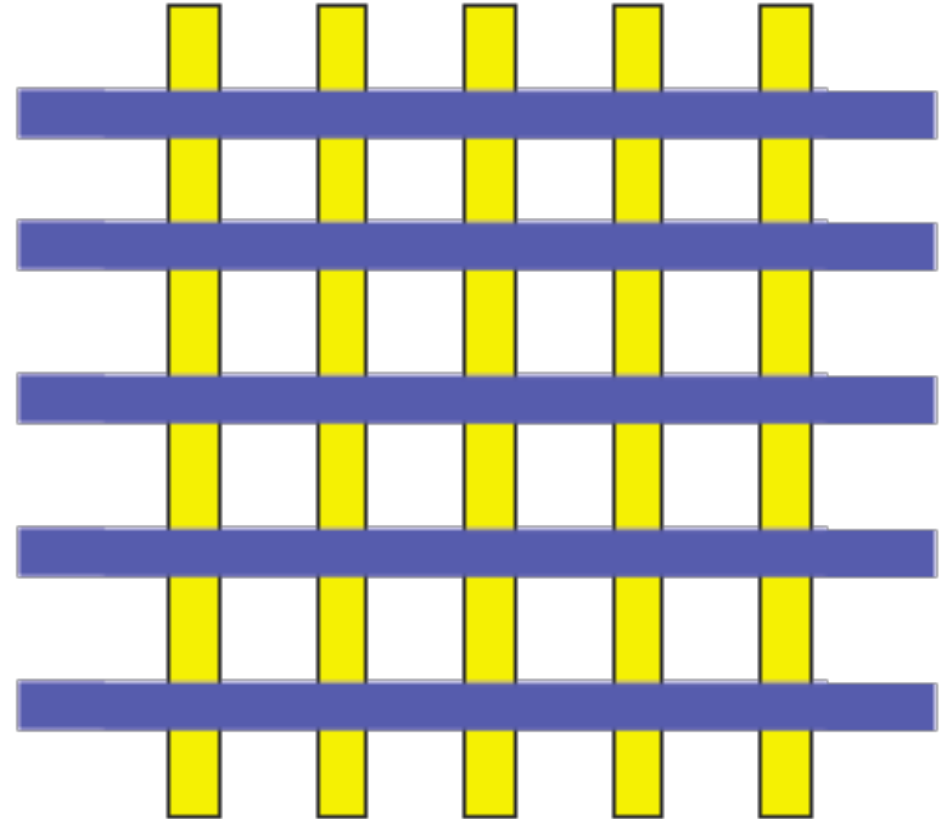
- Kanten der Flächen werden dargestellt
- Eigentlich verdeckte Kanten scheinen durch
- Hidden-Line-Removal:  
Entfernen verdeckter Kanten



## 8.2 Projektion

### Sichtbarkeit von Polygonen

- Projektion auf die Bildebene
- Zerlegung in Pixel
- Pixel werden im Framebuffer abgelegt
- Letztes gerastertes Polygon besetzt Pixelpositionen im Framebuffer



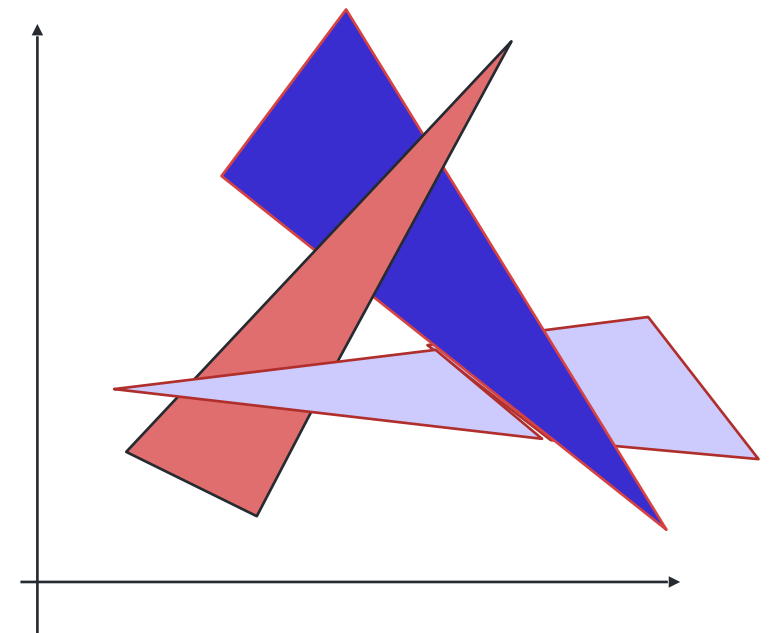
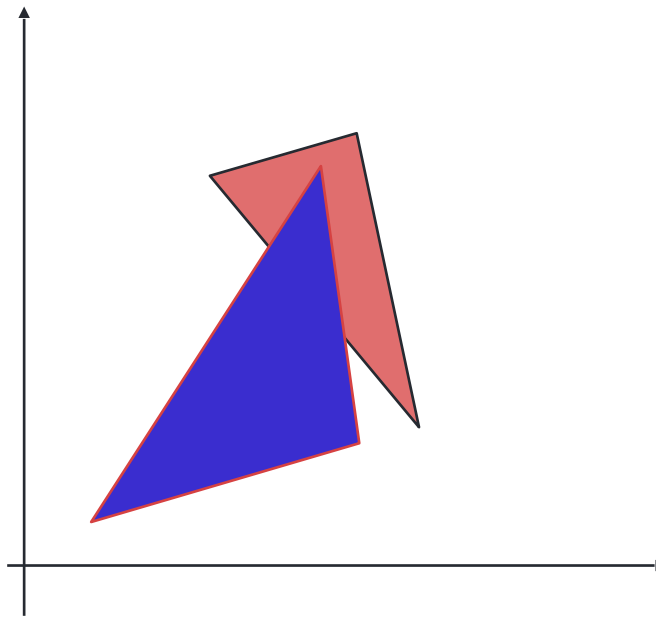
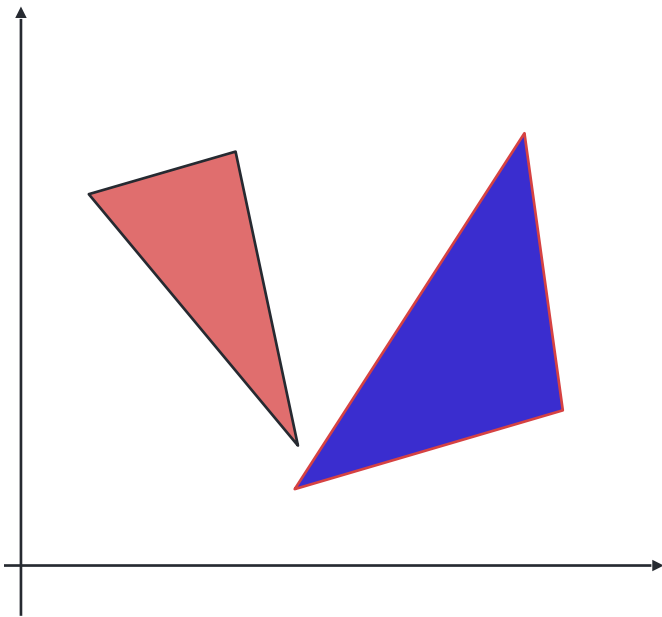


## 8.2 Projektion

### Sichtbarkeit von Polygonen

- Welches Polygon müsste gesehen werden?  
Das dem Beobachter am nächsten Gelegene!

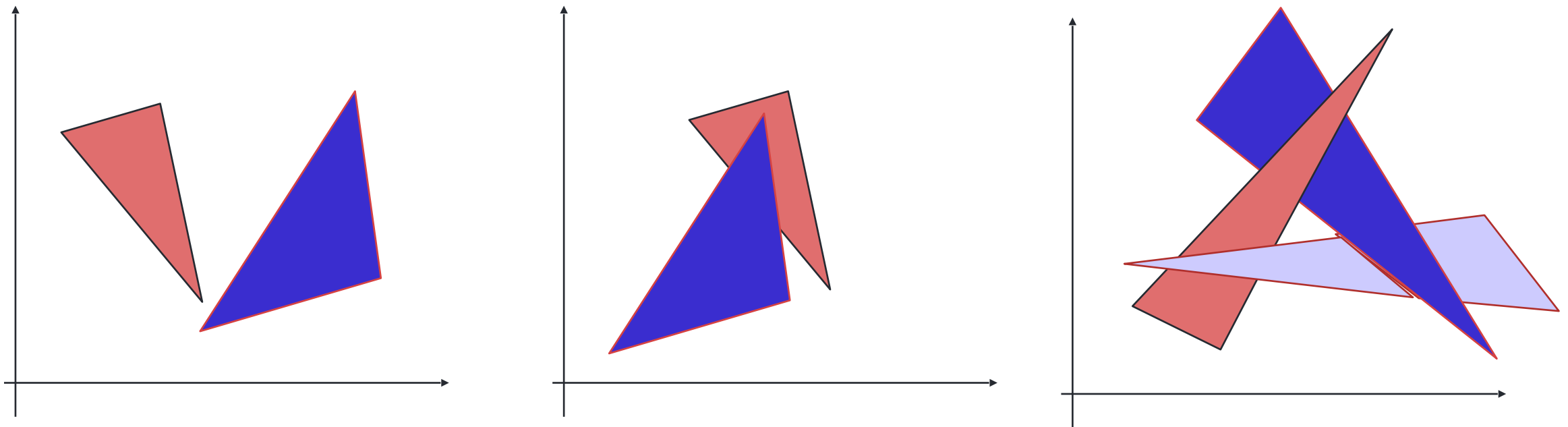
- Klare Aussage, welches das nächste ist, ist nicht immer gegeben.



## 8.2 Projektion

### Sichtbarkeit von Polygonen

- Sichtbarkeit: Sortierproblem
- Gleiches muss bei der Berechnung von Transparenz berücksichtigt werden



## 8.2 Projektion

### Painter's Algorithmus

- Idee: Male Polygone von hinten nach vorne
- Erfordert Tiefensortierung!
- Wenn Tiefenwerte (z-Wert) der Polygone sich überlappen, müssen Polygone geschnitten werden
  - $n^2$  mögliche Teile!
  - Beginne mit dem (Teil-)Polygon mit größtem z-Wert
- Komplexität:
  - $O(n^2)$
  - $n$  ist Anzahl der Dreiecke

## 8.2 Projektion

### Erste Verfahren

- Erste Lösung des Hidden-Line-Problems:  
Roberts, 1963:  
Objektraumverfahren für konvexe Objekte
- Appels Algorithmus (1967):  
Berechnet sichtbare Kanten/  
Konturen (NPR)
- Area Subdivision (divide-and-conquer):  
Warnock, 1969: Ausnutzung von Flächenkohärenz durch Quadrees
- Sample Spans:  
Watkins, 1970: Ausnutzung von Rasterzeilenkohärenz

## 8.2 Projektion

### Erste Verfahren

- Depth List:  
Newell et al., 1972:  
Prioritätslistenalgorithmus im  
Objektraum
- Weiler-Atherton-Algorithmus (1977):  
Sortiert Polygone näherungsweise  
in der Tiefe

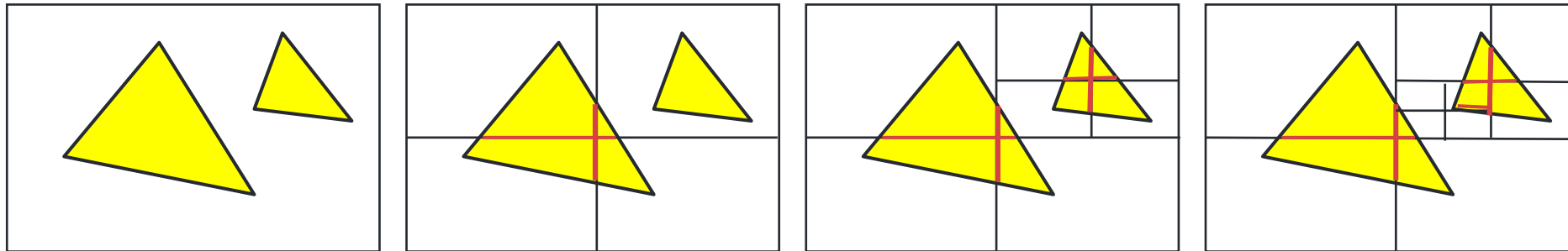
## 8.2 Projektion

Divide and Conquer [Warnock 1969]

- Einfache Fälle
  - Naheste Fläche überdeckt gesamten Rasterbereich
  - Es gibt maximal eine Fläche im Rasterbereich
- Ansonsten rekursive Unterteilung bis nur noch einfache Fälle auftreten
- Aufwand:
  - $O(n \cdot p)$
  - $p$ : Anzahl der Pixel
  - $n$ : Anzahl der Polygone
- Ggf. muss Unterteilung bis auf Pixelebene durchgeführt werden
  - Fast immer bei  $n > p$  (große Szenen)
  - Entspricht dann dem Z-Buffer Algorithmus
    - aber mit Overhead

## 8.2 Projektion

Divide and Conquer [Warnock 1969]



## 8.2 Projektion

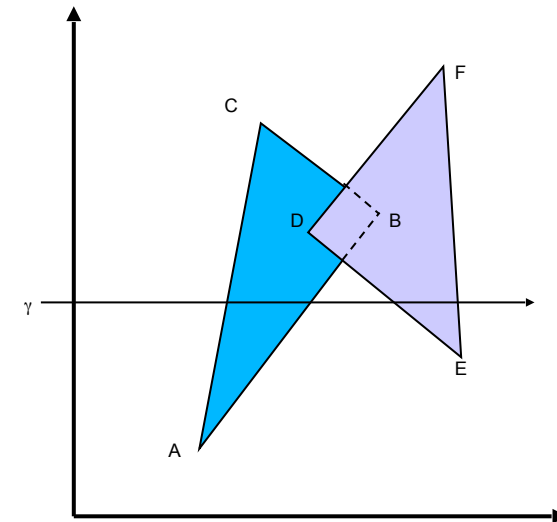
Scanline/Sample Span [Watkins 1970]

- Suche nach Kanten entlang der  $y$ -Scanline  $\gamma$
- Sortiere Kanten nach Tiefenwerten
- Datenstruktur
  - Kantentabellen
  - Polygontabellen
  - ActiveEdge-Tabellen (AET)
- Aktualisierung der AET für jede neue Scanline  $\gamma$

- Scanline  $\gamma$  in AET:

-  $P_1, P_2$  Polygone

AC	AB	DE	FE
Ein	Aus	Ein	Aus





## 8.2 Projektion

### Z-Buffer-Algorithmus

- [Straßer 1974, Catmull 1974]
  - Bestimmt Sichtbarkeit von Pixeln (Bildraum)
  - Geeignet für die Bildausgabe auf Rastergeräten
  - Einfache Hardware-Unterstützung
- Arbeitsweise
    - Sucht für jeden Pixel bei Rasterung nach Polygon mit kleinstem (am weitesten vorne liegendem) z-Wert
    - Zusätzlicher Speicher
      - Z-Buffer
      - Speichere in jedem Pixel den bisher kleinsten aufgetretenen z-Wert

## 8.2 Projektion

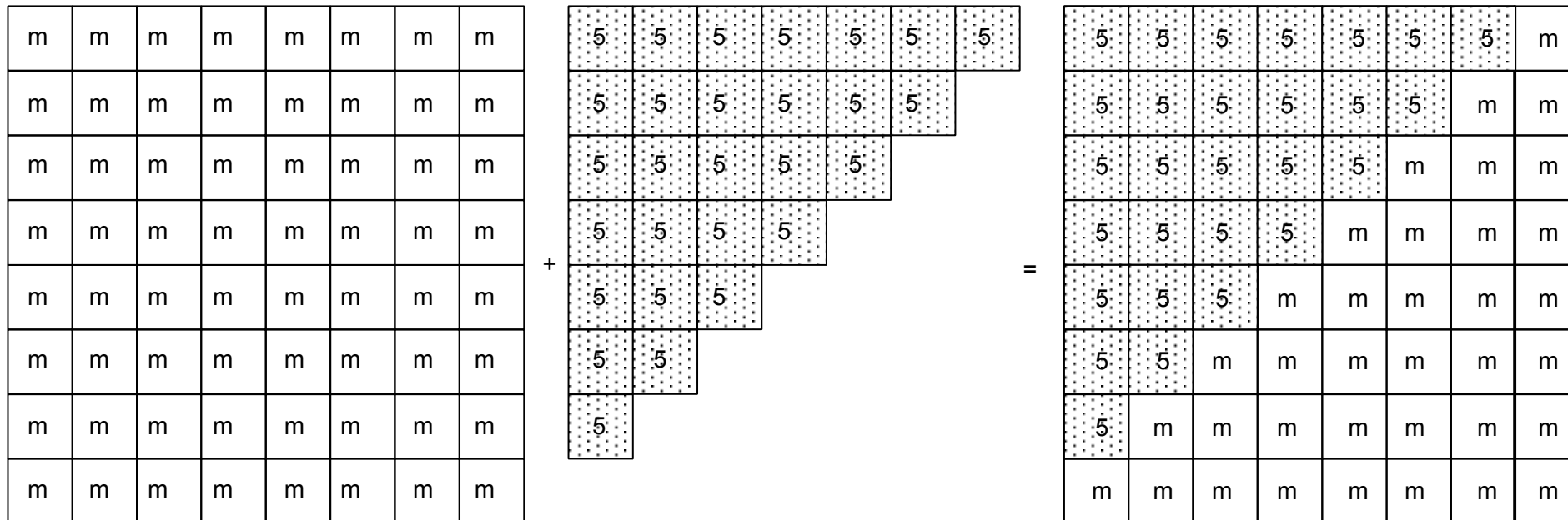
### Z-Buffer-Algorithmus

- Initialisiere Framebuffer (Farbbuffer) mit Hintergrundfarbe
- Initialisiere Z-Buffer mit maximalem  $z$ -Wert
- Scan-Conversion aller Polygone in beliebiger Reihenfolge
  - Berechne  $z$ -Wert für jedes Pixel  $(x, y)$  im Polygon
  - $z < z_{Buffer}$ :
    - zeichne Polygonfarbe in Farbbuffer bei  $(x, y)$  ein
    - setze  $z_{Buffer} = z$
- Am Ende enthält
  - der Farbbuffer das gewünschte Bild
  - der z-Buffer dessen Tiefenverteilung

# 8.2 Projektion

## Z-Buffer-Algorithmus

- z-Werte codiert durch Zahlen:  
kleinere Zahl → näher am Auge
- Initialisiere Z-Buffer mit z-Wert  $m$   
(ganz hinten)
- Addiere ein Polygon mit konstantem  
z-Wert 5



## 8.2 Projektion

### Z-Buffer-Algorithmus

- Addiere ein Polygon, welches das 1. Polygon schneidet

- Artefakte bei Pixeln mit gleichem z-Wert beider Polygone

5	5	5	5	5	5	5	m
5	5	5	5	5	5	m	m
5	5	5	5	5	m	m	m
5	5	5	5	m	m	m	m
5	5	5	m	m	m	m	m
5	5	m	m	m	m	m	m
5	m	m	m	m	m	m	m
m	m	m	m	m	m	m	m

+

8							
7	8						
6	7	8					
5	6	7	8				
4	5	6	7	8			
3	4	5	6	7	8		

=

5	5	5	5	5	5	5	m
5	5	5	5	5	5	m	m
5	5	5	5	5	m	m	m
5	5	5	5	m	m	m	m
5	5	5	8	m	m	m	m
4	5	6	7	8	m	m	m
3	4	5	6	7	8	m	m
m	m	m	m	m	m	m	m

## 8.2 Projektion

### Z-Buffer-Algorithmus

- Zur Berechnung von  $z(x, y)$  für ebene Polygone (z.B. Dreiecke) entlang einer Scan-Line

- Ebene:  $Ax + By + Cz + D = 0$

- $z(x, y) = -\frac{D + Ax + By}{C}$

$$z(x + \Delta x, y) = -\frac{D + A(x + \Delta x) + By}{C}$$

- $= z(x, y) - \Delta x \cdot \frac{A}{C}$

- Nur eine Subtraktion ist notwendig

- $\frac{A}{C}$  ist konstant

- $\Delta x = 1$

## 8.2 Projektion

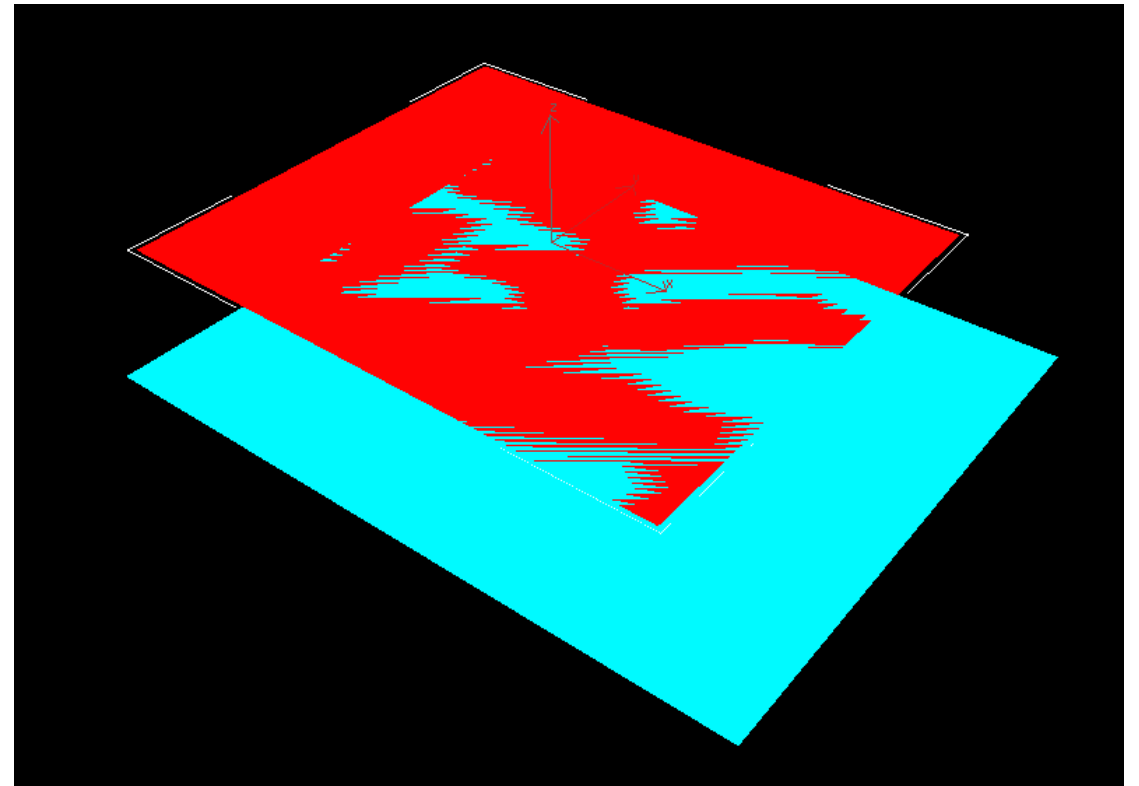
### Z-Buffer-Algorithmus: Vorteile

- sehr einfache Implementierung des Algorithmus
- keine besondere Reihenfolge oder Sortierung notwendig
- keine Komplexitätsbeschränkung der Bildszene
- unabhängig von der Repräsentation der Objekte; es muss nur möglich sein, zu jedem Punkt der Oberfläche einen  $z$ -Wert zu bestimmen
- Auflösung des Z-Buffers bestimmt Diskretisierung der Bildtiefe
  - 20 Bit  $\rightarrow 2^{20}$  Tiefenwerte unterscheidbar

## 8.2 Projektion

### Z-Buffer-Algorithmus: Nachteile

- problematisch sind weit entfernte Objekte mit kleinen Details (perspektivische Transformation)
- Aufwändige Modifikationen notwendig für
  - Transparenz (Alpha-Buffering)
  - Antialiasing



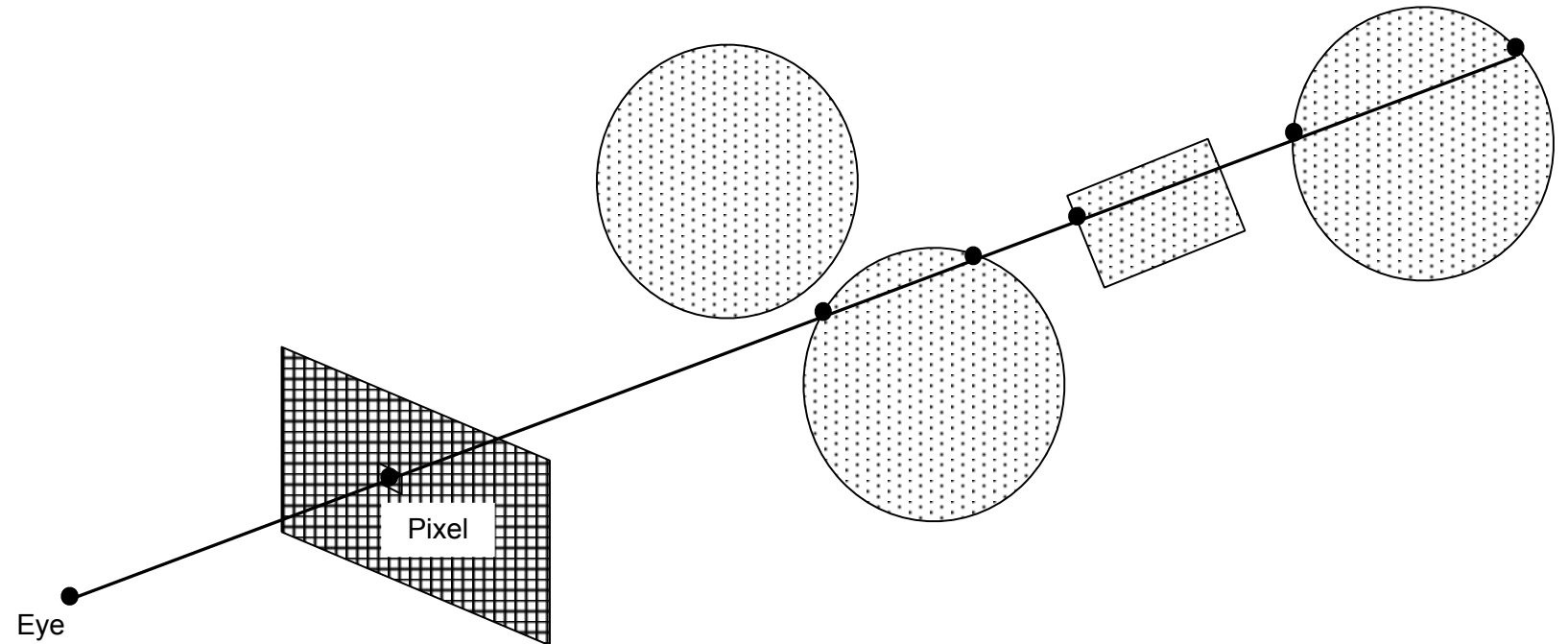
## 8.3 Strahlverfolgung

- RayCasting
  - Sendet (to cast) Strahlen (ray)
    - vom Auge
    - durch alle Pixel der Bildebene
    - in den Objektraum
  - Polygon opak
    - addiere Farbwert hinzu
    - stopp
  - Polygon transparent
    - addiere Farbwert hinzu
    - stopp
- RayTracing
  - RayCasting
  - Farbwert
    - Wie RayCasting
    - Verfolge reflektierte Strahlen
    - Verfolge gebrochene Strahlen
  - globales Beleuchtungsmodell



## 8.3 Strahlverfolgung

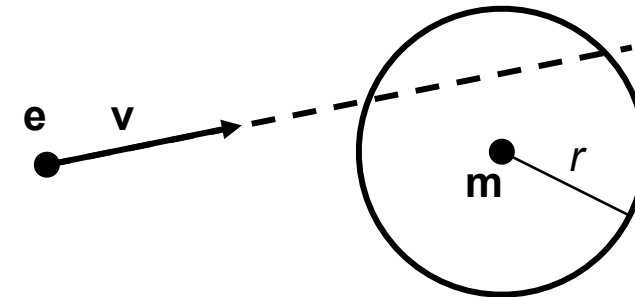
- Berechne Schnittpunkte mit allen Objekten der Szene
- Objekt mit dem nächst gelegenen Schnittpunkt ist in diesem Pixel sichtbar



## 8.3 Strahlverfolgung

### Schnitt des Strahls mit einer impliziten Kugel

- Strahl
  - $e$ : Augpunkt
  - $v$ : Sichtrichtung (Pixel -  $e$ )
  - $t$ : Strahlparameter
  - $r(t) = e + t \cdot v$
- Kugel
  - $\|x - m\|^2 - r^2 = 0$



## 8.3 Strahlverfolgung

Schnitt des Strahls mit einer impliziten Kugel

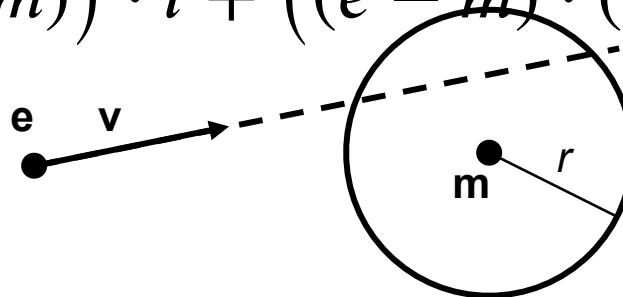
– Einsetzen des Strahls  $r(t)$  für  $x$  liefert:

$$\|e + t \cdot v - m\|^2 - r^2 = 0$$

$$\|t \cdot v + (e - m)\|^2 - r^2 = 0$$

$$t^2 \cdot v \cdot v + 2 \cdot t \cdot v \cdot (e - m) + (e - m) \cdot (e - m) - r^2 = 0$$

$$(v \cdot v) \cdot t^2 + (2 \cdot v \cdot (e - m)) \cdot t + ((e - m) \cdot (e - m) - r^2) = 0$$



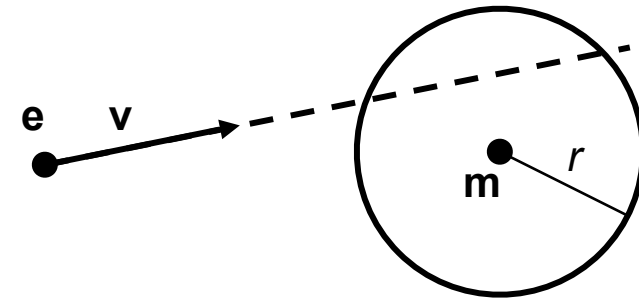
## 8.3 Strahlverfolgung

Schnitt des Strahls mit einer impliziten Kugel

- Lösen der quadratischen Gleichung nach  $t$  liefert die Parameter von maximal 2 Schnittpunkten

$$s_{1,2} = r(t_{1,2}) = e + t_{1,2} \cdot v$$

- Schnittpunkt mit kleinstem  $t > 0$  liegt dem Augpunkt am nächsten



## 8.3 Strahlverfolgung

### Schnitt des Strahls mit einer Ebene

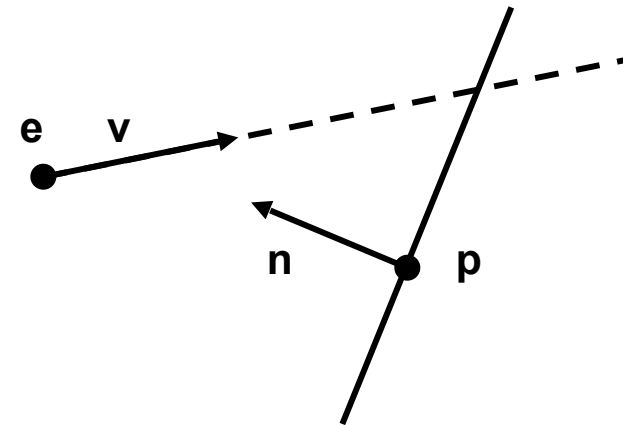
#### – Strahl

- $e$ : Ausgangspunkt
- $v$ : Sichtrichtung (Pixel –  $e$ )
- $t$ : Strahlparameter

- $r(t) = e + t \cdot v$

#### – Ebene

- $p$ : Punkt der Ebene
- $n$ : Normalenvektor nach außen
- $(x - p) \cdot n = 0$



## 8.3 Strahlverfolgung

### Schnitt des Strahls mit einer Ebene

- Einsetzen des Strahls  $r(t)$  für  $x$  liefert:

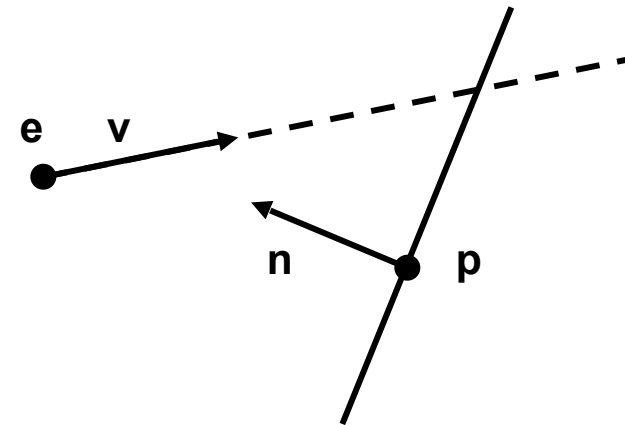
$$(e + t \cdot v - p) \cdot n = 0$$

$$t \cdot v \cdot n + (e - p) \cdot n = 0$$

$$t = \frac{(p - e) \cdot n}{v \cdot n}$$

- Schnittpunkt

$$s = r(t) = e + t \cdot v$$



## 8.3 Strahlverfolgung

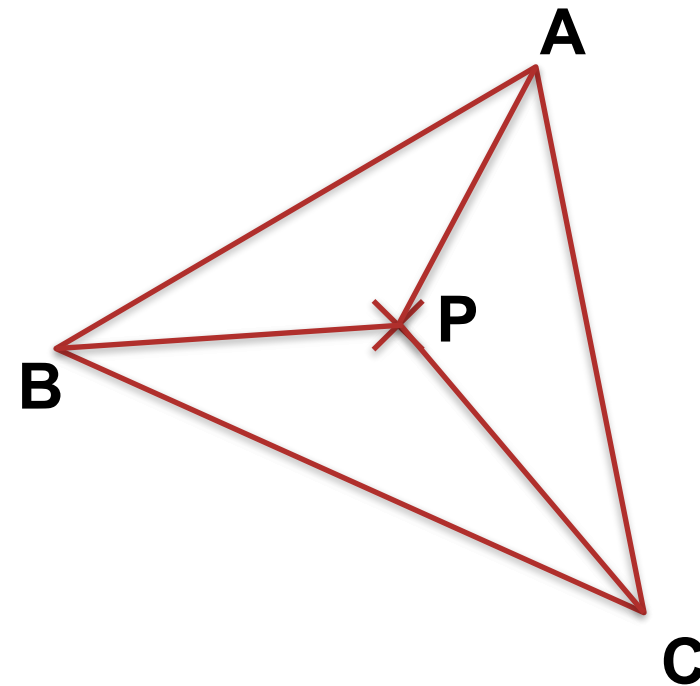
### Schnitt des Strahls mit einer Ebene

- Beim Schneiden von Polygonen ist noch die Gültigkeit des Schnittpunktes zu verifizieren
- Dieser muss innerhalb des Polygons liegen
- Dreiecke
  - Bestimme Summe der Flächeninhalte der Teildreiecke
    - Ist diese Summe größer als der Flächeninhalt des ursprünglichen Dreiecks, so liegt der Punkt außerhalb
    - Achtung: Rundungsfehler!
  - Alternativ Bestimme die baryzentrischen Koordinaten des Dreiecks  $(a, b, c)$

## 8.3 Strahlverfolgung

### Baryzentrische Koordinaten

- Gegeben: Dreieck  $(A, B, C)$
- Gesucht: Koordinaten von  $P$  bezüglich des Dreiecks  $(A, B, C)$
- Ansatz:  
$$P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$$
- Nebenbedingung:  
$$\alpha + \beta + \gamma = 1$$
  
(Normalisierung)

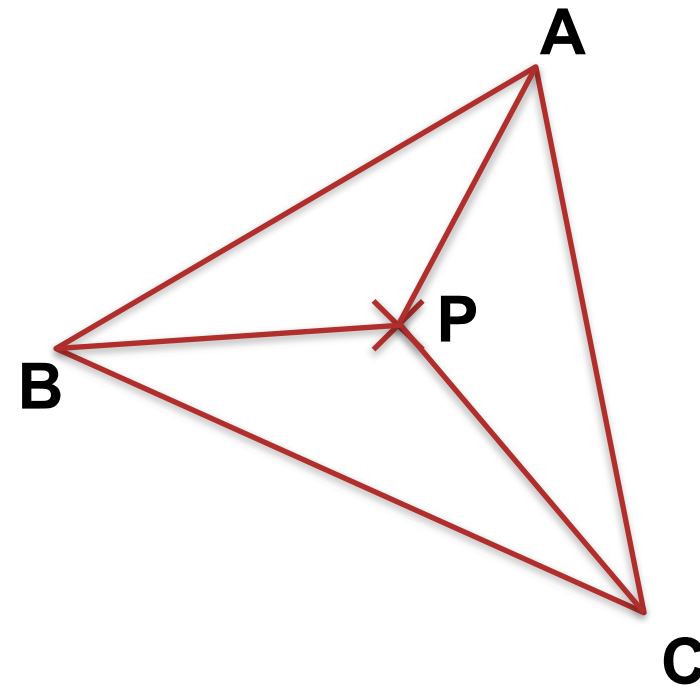




## 8.3 Strahlverfolgung

### Baryzentrische Koordinaten

- $P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$
- Folgerungen:
  - $A = (1, 0, 0)$
  - $B = (0, 1, 0)$
  - $C = (0, 0, 1)$
- $\alpha = 0, \beta + \gamma = 1$ : Kante  $B - C$
- $\beta = 0, \alpha + \gamma = 1$ : Kante  $A - C$
- $\gamma = 0, \alpha + \beta = 1$ : Kante  $A - B$
- $0 \leq \alpha, \beta, \gamma \leq 1$ :  $P$  liegt innerhalb
- Sonst:  $P$  liegt außerhalb



## 8.3 Strahlverfolgung

### Baryzentrische Koordinaten

-  $P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$

$$P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$$

$$= (1 - \beta - \gamma) \cdot A + \beta \cdot B + \gamma \cdot C$$

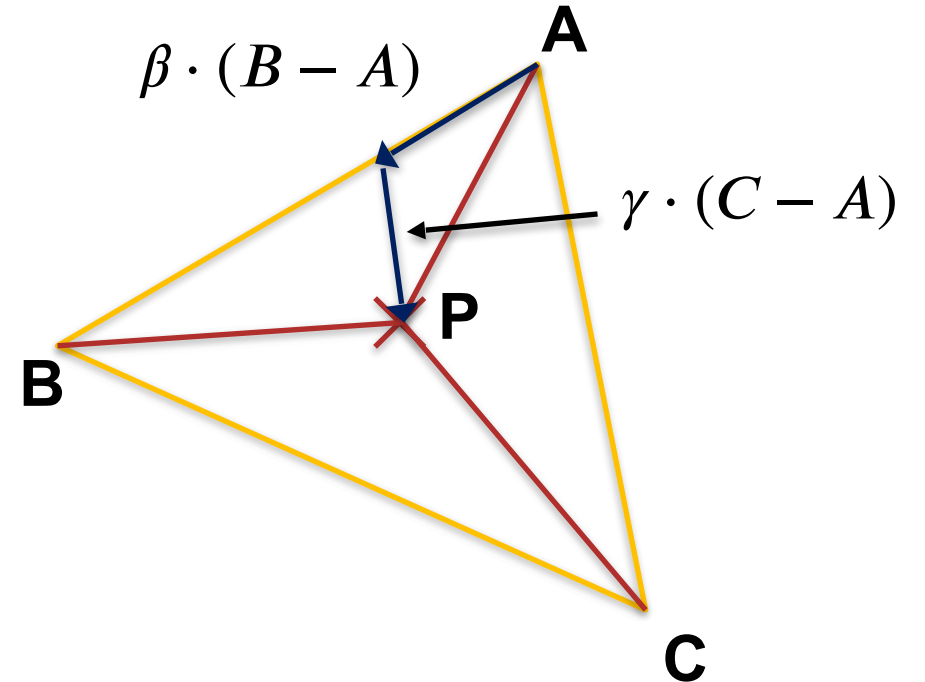
$$= A + \beta \cdot (B - A) + \gamma \cdot (C - A)$$

- Berechnung von  $(\alpha, \beta, \gamma)$

$$\alpha = \frac{\Delta(P, B, C)}{\Delta(A, B, C)}$$

$$\Delta(P, B, C) = \underbrace{((B - P) \times (C - P))}_{\text{Flächeninhalt des Parallelogramms}} \circ \left( \frac{(B - A) \times (C - A)}{\|(B - A) \times (C - A)\|} \right) \underbrace{\Delta}_{\substack{\Delta \text{ entspricht nicht dem} \\ \text{Flächeninhalt des Dreiecks,} \\ \text{sondern dem doppelten} \\ \text{orientierten Flächeninhalt}}}$$

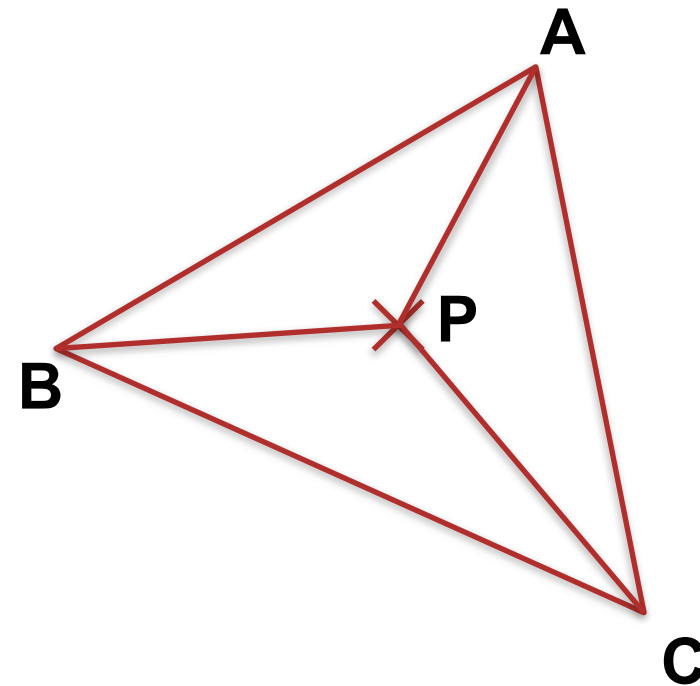
Einheitsvektor kodiert  
Orientierung als Vorzeichen



## 8.3 Strahlverfolgung

### Baryzentrische Koordinaten

- Effizientere Berechnung und Algorithmen:  
Christer Ericson.  
Real Time Collision Detection.  
Morgan Kaufman – Elsevier, 2005.



## 8.3 Strahlverfolgung

- Nachteile
  - 1 Strahl pro Pixel
    - Full HD: ca. 2 Millionen Strahlen
  - Schnitttests
    - jeder Strahl
    - jedes Objekt
  - Für typische Szenen:
    - bis zu 95% der Rechenzeit
- Beschleunigung
  - Vermeidung von unnötigen Schnitttests
    - Hierarchien von Bounding Volumes
      - Bounding Spheres
      - Bounding Boxes
      - Axis-Aligned Bounding Boxes
  - Raumteilung
  - Occlusion Culling

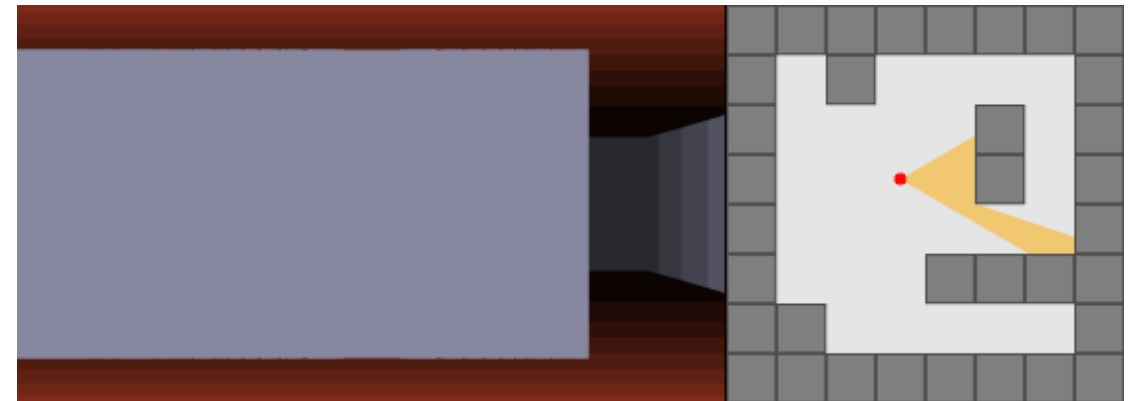
## 8.3 Strahlverfolgung

### Hierarchien von Bounding Volumes

- Baumartige Strukturen von Bounding Volumes
  - Blätter: Objekte der Szene (Geometrie)
  - Innere Knoten: Bounding Volumes um die Objekte der Unterbäume
- Schneidet ein Strahl das Bounding Volume eines inneren Knotens nicht, so entfällt ein Test der untergeordneten Teilbäume
- Generierung guter Hierarchien ist schwierig
  - Geometrisch: Nach Szenenausdehnung unterteilen
  - Dichte: Nach Polygonschwerpunkten sortieren und unterteilen
  - Verwendung des Szenengraphen möglich

## 8.3 Strahlverfolgung

- Raycasting kann auch zur Darstellung von 3D Darstellungen genutzt werden
- 2D Blöcke mit Objekte werden mit dem Boden und Decke dargestellt
- Szene wird dann aus der Position des Betrachters berechnet



[https://commons.wikimedia.org/wiki/File:Simple\\_raycasting\\_with\\_fisheye\\_correction.gif](https://commons.wikimedia.org/wiki/File:Simple_raycasting_with_fisheye_correction.gif)

## 8.3 Strahlverfolgung

- Pro Bildspalte wird ein Strahl berechnet
- Die Textur des getroffenen Objektes wird dann spaltenweise in die Szene eingefügt
  - Basierend auf deren Position
  - Anteil Boden und Decke wird addiert
- -> Performancevorteile gegenüber „echtem“ 3D



[https://upload.wikimedia.org/wikipedia/it/5/55/Wolfenstein\\_3D\\_Screenshot.png](https://upload.wikimedia.org/wikipedia/it/5/55/Wolfenstein_3D_Screenshot.png)

## 8.4 Raumteilungsverfahren

- Space-Subdivision-Techniques
  - Der Objektraum wird in disjunkte Teilräume zerlegt
  - Für jedes Objekt wird bestimmt, in welchen Teilräumen es sich befindet
- Standardverfahren
  - Unterteilung des Raumes durch ein festes, regelmäßiges Gitter in Zellen identischer Geometrie
  - 2D: Pixel (Picture Element)
    - Quadrate
  - 3D: Voxel (Volume Element)
    - Würfel



## 8.4 Raumteilungsverfahren

### – Vorteile

- Es ist sehr einfach zu bestimmen, welche Objekte (teilweise) in einem Voxel liegen
- Ist das Voxel leer wird keine Schnittberechnung durchgeführt
- Sonst: teste nur Objekte die in diesem Voxel liegen

### – Nachteile

- Auflösung von  $n$  Voxeln in jeder Dimension:  $n^3$  Voxel
- ⇒ günstigere Repräsentation durch Octrees

## 8.4 Raumteilungsverfahren

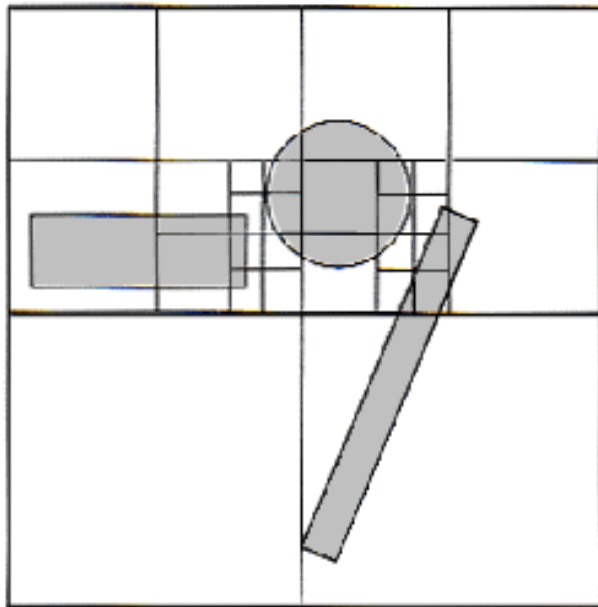
### Quadtrees

- Erstes Auftreten: Ende der 1960er Jahren
- Octrees sind eine Erweiterung von Quadtrees
  - Ende 1970er, Anfang 1980er
- Unterteilung einer Ebene
- Erzeuge Quadrat, welches alle Objekte enthält (Wurzel)
- Rekursive Zerlegung
  - Enthält das Quadrat zu viele Objekte
    - Zerlege das Quadrat in vier Teile
    - Sortiere Objekte in Teilquadrate ein
  - Sonst → Stopp
- Jeder innere Knoten des Baumes besitzt genau vier Kinder

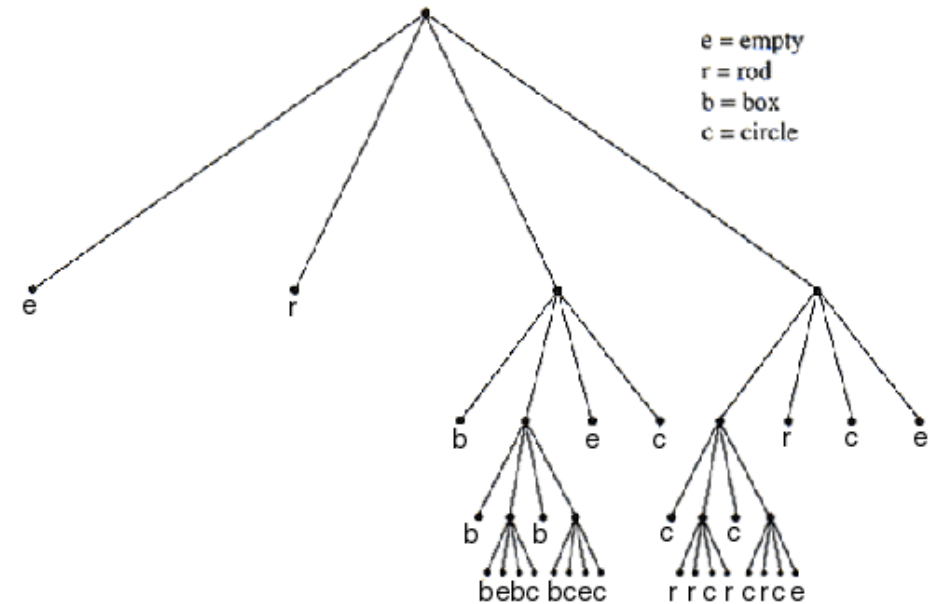
## 8.4 Raumteilungsverfahren

### Quadrees

- Unterteilung des Raumes, bis die Zellen maximal eine Objektreferenz enthalten



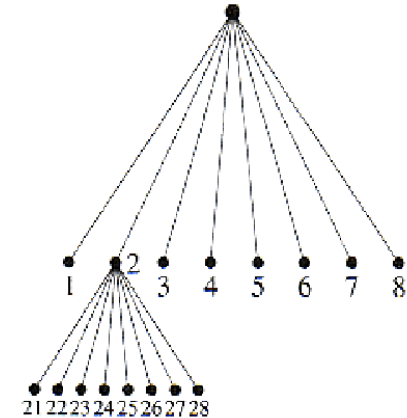
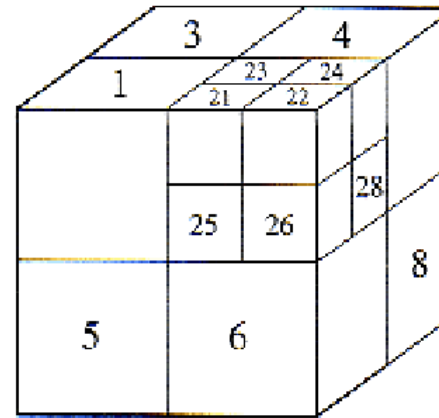
- Zugehörige Quadtree-Datenstruktur, vier Generationen



## 8.4 Raumteilungsverfahren

### Octrees

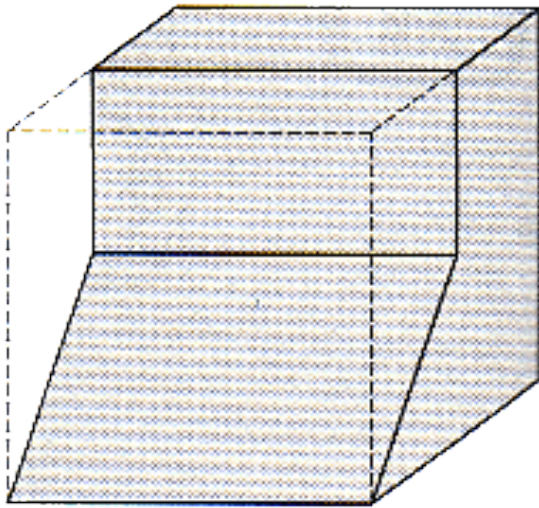
- Unterteilung des 3D-Raums analog dem Quadtree
- Erzeuge Würfel, der alle Objekte enthält (Wurzel)
- Rekursive Zerlegung
  - Enthält der Würfel zu viele Objekte
    - Zerlege den Würfel in acht Teile
    - Sortiere Objekte in Teilwürfel ein
  - Sonst → Stopp
- Jeder innere Knoten des Baumes besitzt genau acht Kinder



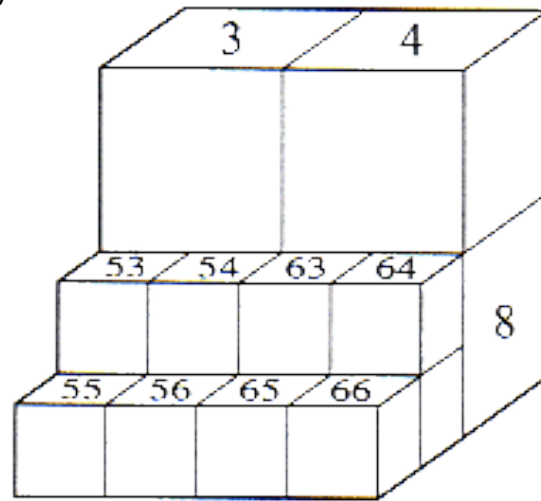
# 8.4 Raumteilungsverfahren

## Octrees

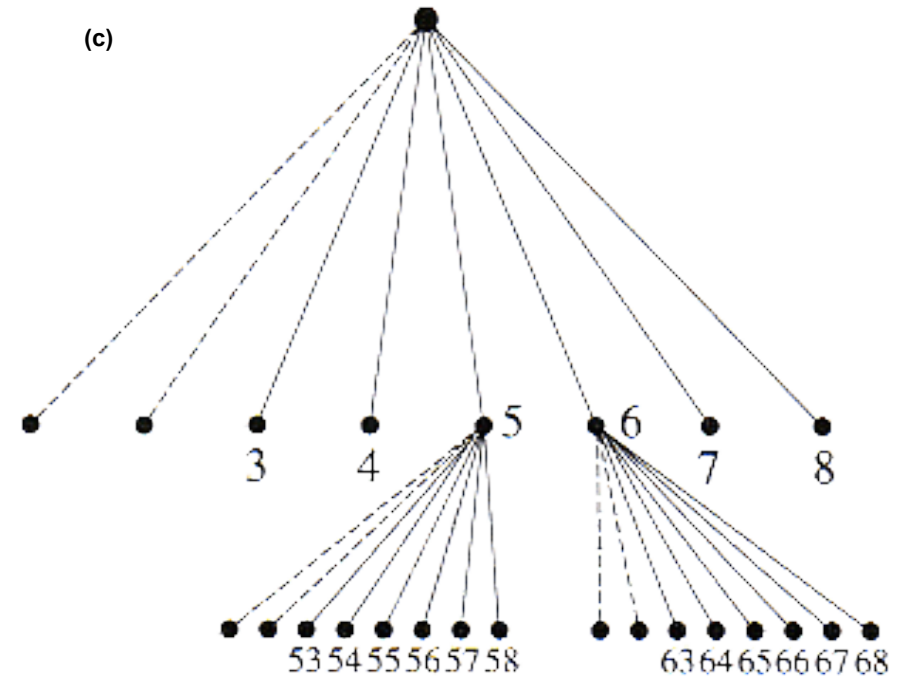
(a)



(b)



(c)



## 8.4 Raumteilungsverfahren

### n-ary trees

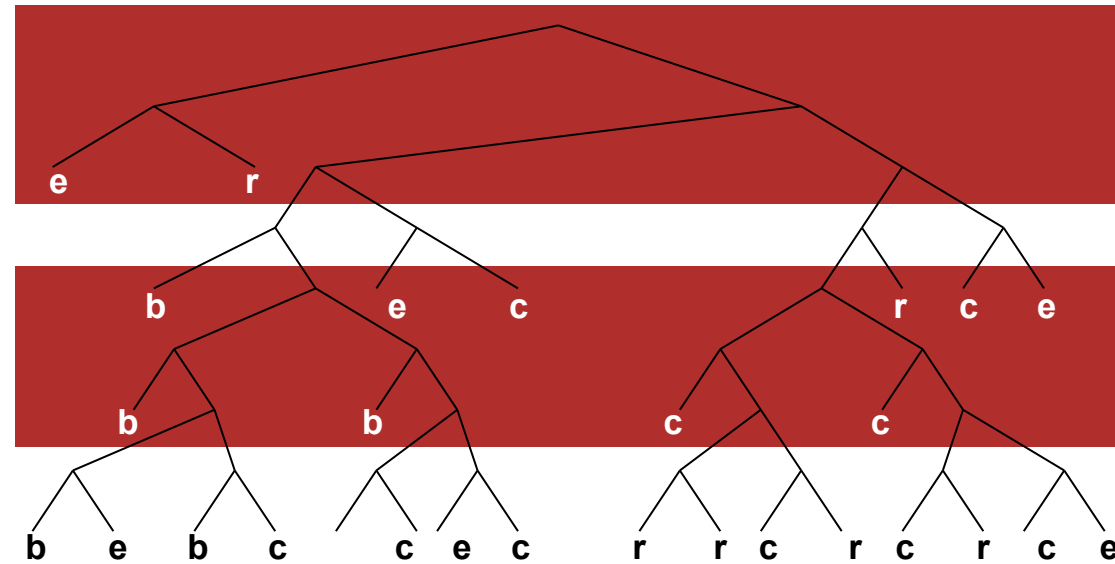
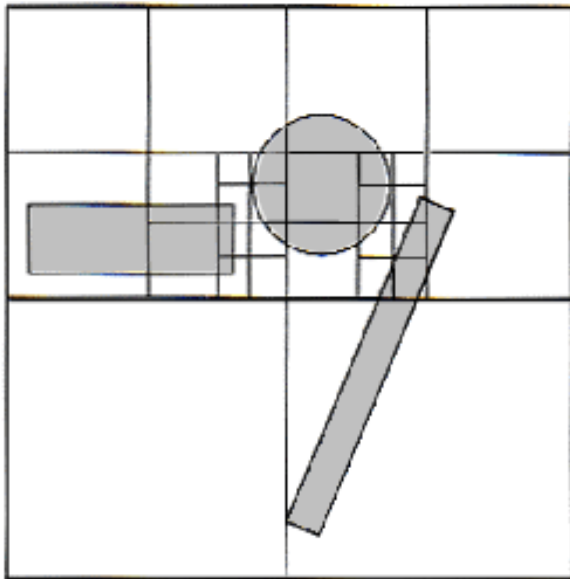
- Verallgemeinerung auf  $n$  Dimensionen
- Muss keine feste Unterteilungsstrategie repräsentieren
- Für den Fall  $n = 1$ 
  - kD-Tree
  - Elternknoten hat bis zu zwei Kinder
- Für den Fall  $n = 2$ 
  - Quadtree
- Für den Fall  $n = 3$ 
  - Octree

# 8.4 Raumteilungsverfahren

## kD-Tree

- Unterteilung des Raumes, bis die Zellen maximal eine Objektreferenz enthalten

- zugehörige kD-Tree-Datenstruktur
  - acht Generationen



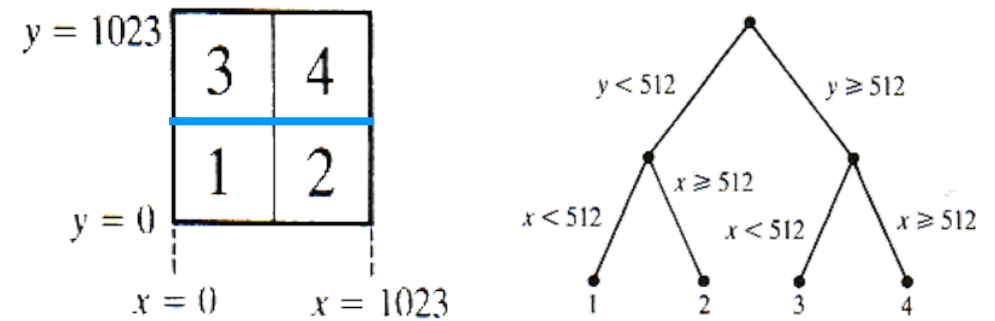
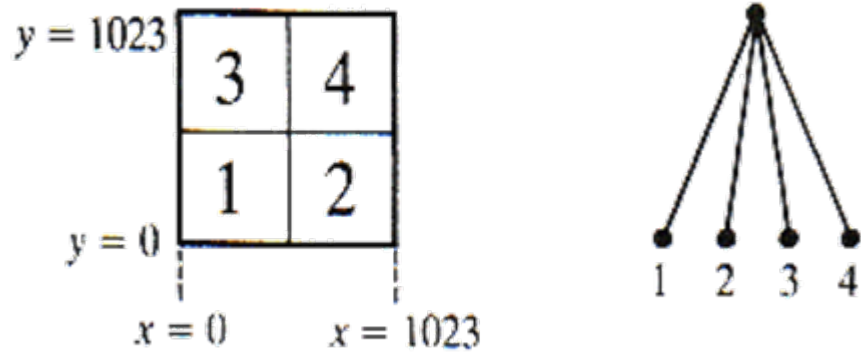
e = empty  
r = rod  
b = box  
c = circle

# 8.4 Raumteilungsverfahren

## Unterteilung einer 2D-Szene

- Quadtree

- kD-Tree

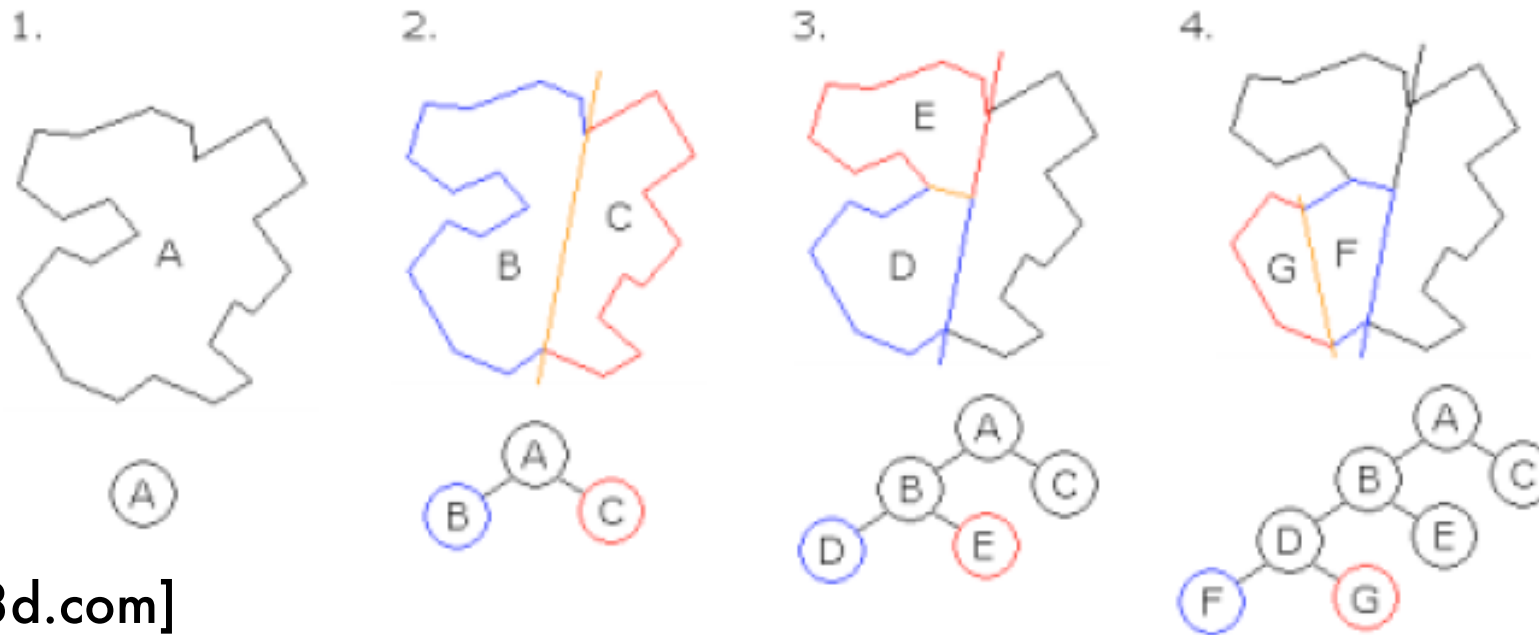




## 8.4 Raumteilungsverfahren

### Binary Space-Partitioning Trees (BSP-Bäume)

- Octrees, Quadrees und kD-Trees
  - Achsen-parallele Unterteilung
  - gleichzeitig oder abwechselnd
- BSP-Trees
  - Unterteilung durch beliebige (Hyper-)Ebene in zwei Unterräume



[[www.beyond3d.com](http://www.beyond3d.com)]

## 8.4 Raumteilungsverfahren

### Binary Space-Partitioning Trees (BSP-Bäume)

- Können zur Unterteilung von Szenen verwendet werden
- Sind an kein Raster gebunden
- Über die relative Lage der Regionen zum Betrachter kann die Tiefenstaffelung der Objekte bestimmt werden
  - Welche Objekte können sichtbar sein?
- Eine ideale Wahl der Unterteilungsebene für BSP-Bäume liefert die PCA (Principal Component Analysis)
- Angenommen, eine komplexe Szene ist gegeben durch die Punktwolke  $P_i \in \mathbb{R}^3, (i = 1, \dots, n)$  (z.B. Objektmittelpunkte oder Eckpunkte der Polygone)
- PCA liefert ein orthogonales Koordinatensystem  $e_1, e_2, e_3$  dessen Ausrichtung der Punktwolke entspricht

## 8.4 Raumteilungsverfahren

### Binary Space-Partitioning Trees (BSP-Bäume)

- Wähle als Ursprung den Mittelwert

$$c = \frac{1}{n} \sum_{i=1}^n P_i$$

- Konstruiere Matrix der Punktwolke

$$A = \begin{pmatrix} P_{1,x} - c_x & P_{1,y} - c_y & P_{1,z} - c_z \\ P_{2,x} - c_x & P_{2,y} - c_y & P_{2,z} - c_z \\ \vdots & \vdots & \vdots \\ P_{n,x} - c_x & P_{n,y} - c_y & P_{n,z} - c_z \end{pmatrix}$$

- Berechne

$$B = \frac{1}{n-1} A^T A$$

$$b_{ij} = \frac{1}{n-1} \sum_{k=1}^n a_{ki} a_{kj}$$

- $B$  hat

- Reelle Eigenwerte  $\lambda_1, \lambda_2, \lambda_3$

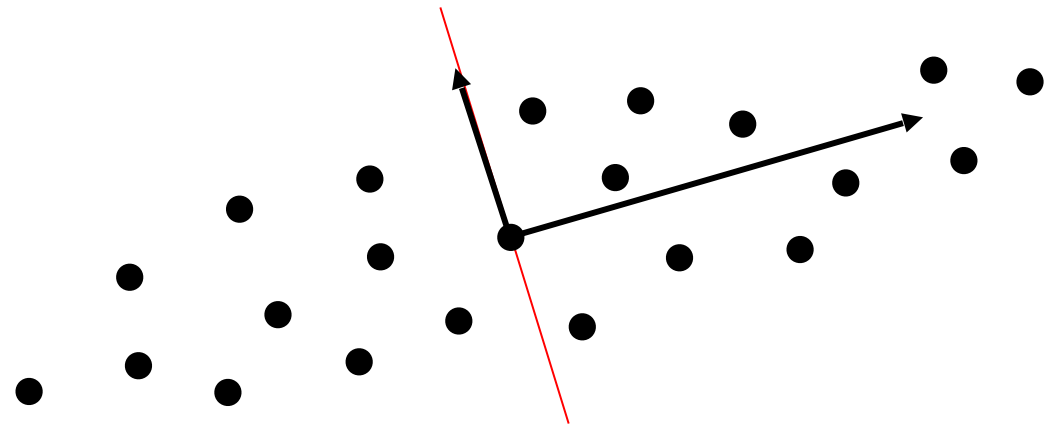
- Eigenvektoren  $e_1, e_2, e_3$

$$\lambda_i \cdot e_i = B \cdot e_i$$

## 8.4 Raumteilungsverfahren

### Binary Space-Partitioning Trees (BSP-Bäume)

- Die Eigenvektoren bilden zusammen mit dem Ursprung  $c$  das gesuchte System
- Die Ausdehnung der Punktwolke in Richtung  $e_i$  ist proportional zu  $\sqrt{\lambda_i}$



## 8.4 Raumteilungsverfahren

### Principal Component Analysis (PCA)

- Andere Namen
  - Karhunen-Loève Transformation (KLT, Bildverarbeitung)
  - Empirical orthogonal functions (Meteorologie, Geophysik)
  - Hotelling Transformation
  - Proper orthogonal decomposition (POD)
  - Deutsch: Hauptkomponentenanalyse
- Ziele
  - Berechne eine reduzierte Menge von Dimensionen
    - Orthogonal
    - Linear
  - Ordne die Dimensionen absteigend bezüglich der Varianz
- Varianz
  - Maß für den Informationsgehalt einer Variable
- Mathematisch
  - Suche eine neue Basis des Vektorraumes

## 8.4 Raumteilungsverfahren

### Principal Component Analysis (PCA)

- Konstruktion

- Gegeben:

$$X = (X_1, \dots, X_m), X_j = (x_{1,j}, \dots, x_{n,j})^T$$

- Mittelwert:

$$\mu = (\mu_1, \dots, \mu_m), \mu_j = \frac{1}{n} \cdot \sum_{i=1}^n x_{i,j}$$

- Zentrierte Werte:

$$Y = (y_{ij}), y_{ij} = x_{ij} - \mu_j$$

- Kovarianz-Matrix:

$$C = \frac{1}{n-1} \cdot Y^T Y$$

- Symmetrisch → spektrale Zerlegung möglich

- Spektrale Zerlegung:

$$C = U \Lambda U^T, U^T U = I_m$$

- Eigenwert Matrix:

$$\Lambda = \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_m \end{pmatrix}$$

- Eigenvektor Matrix:

$$U = (u_1, \dots, u_m)$$

- Ordne die Eigenwerte absteigend:

$$i > j \rightarrow \lambda_i > \lambda_j$$

## 8.4 Raumteilungsverfahren

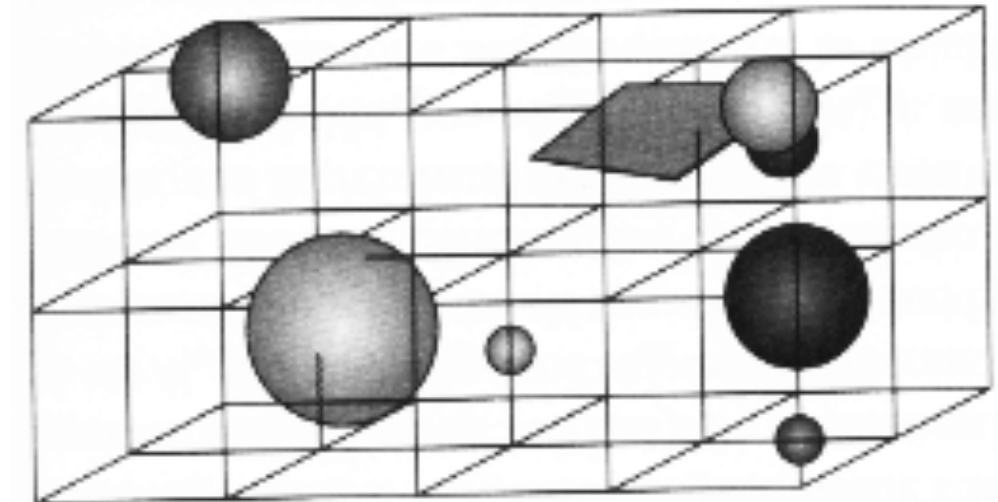
### Principal Component Analysis (PCA)

- Eigenschaften:
  - $\lambda_1, u_1$ : größte Varianz
  - $\lambda_2, u_2$ : zweitgrößte Varianz
  - ...
  - Die ersten  $p$  Eigenwerte beschreiben einen Großteil der Varianz
  - Principal components mit einer Varianz nahe 0 können verwendet werden, um Ausreißer zu identifizieren
- Wird auch häufig in der Scientific Visualization verwendet
- Einschränkungen:
  - Das Ergebnis ist abhängig von der Skalierung der einzelnen Variablen (Attribute)
  - Ausreißer haben einen großen Einfluss auf das Ergebnis
  - Lineare Methode
  - Interpretation der Basis (Ergebnisse)

## 8.4 Raumteilungsverfahren

### Ray-Casting

- Für das Raycasting wird der Raum zunächst unterteilt
- Dann werden die Objekte in die Teilräume einsortiert

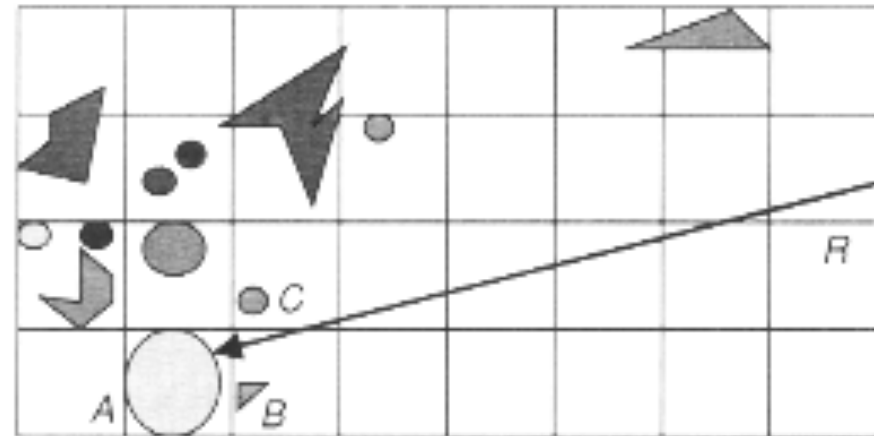




## 8.4 Raumteilungsverfahren

### Ray-Casting

- Der Strahl läuft zum ersten Teilraum
- Ist der Teilraum leer
  - keine Schnittberechnung notwendig
- Enthält der Teilraum Objekte
  - Schnitttest mit allen enthaltenen Objekten
  - Schnitt
    - Fertig
  - Kein Schnitt
    - Betrachte nächsten Teilraum in Richtung der Strahls



## 8.1 PROBLEMSTELLUNG

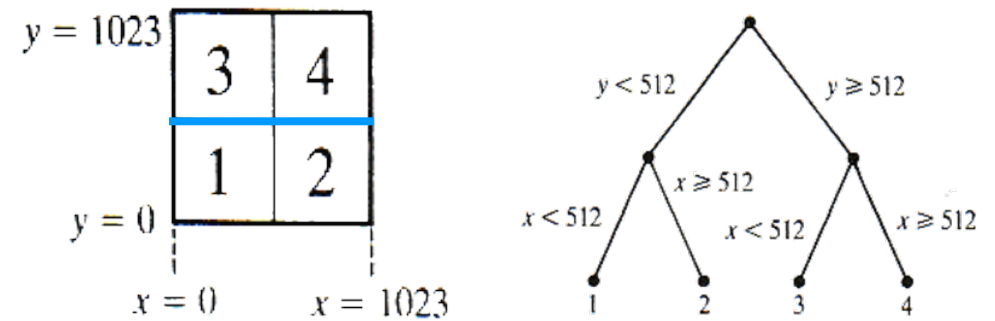
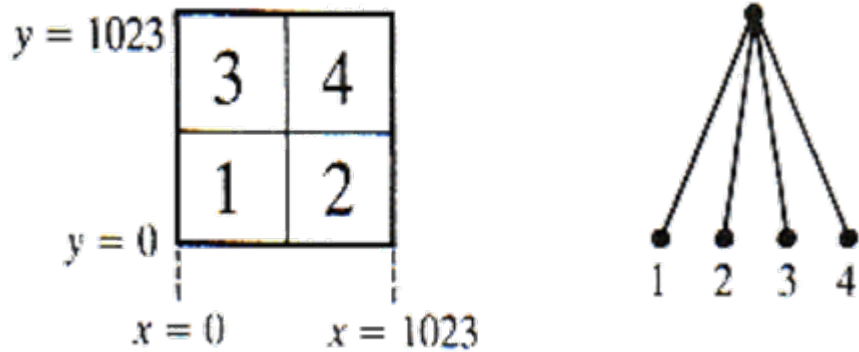
- Sichtbarkeitsberechnung
  - Ihr Ziel ist die (möglichst) exakte Bestimmung der von einem gegebenen Blickpunkt aus sichtbaren bzw. unsichtbaren Teile der darzustellenden Szene
- Anforderungen
  - Wünschenswert ist **hohe Interaktionsrate**, so dass Eingaben des Benutzers sich direkt auf die Darstellung auswirken
  - In den meisten Fällen ist die **Echtzeitausgabe** der Szene nötig
- Einteilung der Verfahren:
  - **Objektraumverfahren**
    - prinzipiell geräteunabhängig
    - Rechengenauigkeit ist die Maschinengenauigkeit
  - **Bildraumverfahren**
    - geräteabhängig
    - Rechengenauigkeit ist die Auflösung des Ausgabegerätes

# 8.4 Raumteilungsverfahren

## Unterteilung einer 2D-Szene

- Quadtree

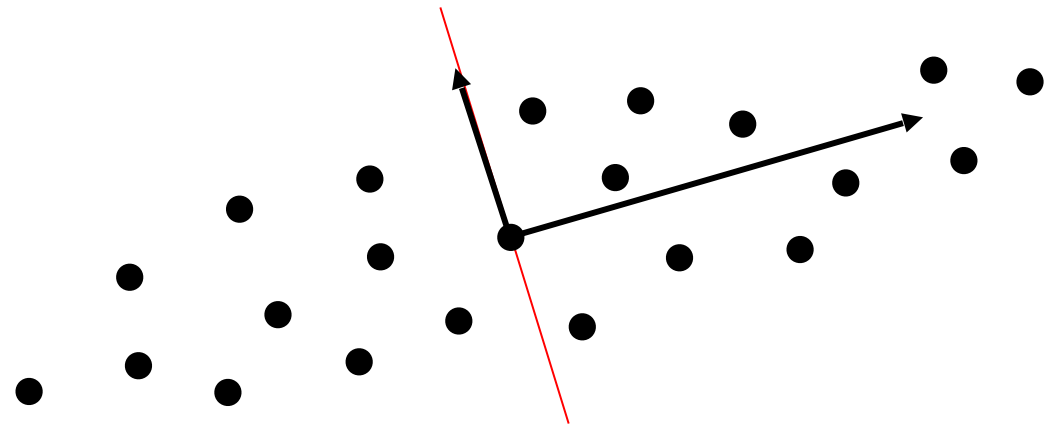
- kD-Tree



## 8.4 Raumteilungsverfahren

### Binary Space-Partitioning Trees (BSP-Bäume)

- Die Eigenvektoren bilden zusammen mit dem Ursprung  $c$  das gesuchte System
- Die Ausdehnung der Punktwolke in Richtung  $e_i$  ist proportional zu  $\sqrt{\lambda_i}$



## 8.5 Culling

- Begriff aus dem Englischen:  
Auslese, Ausdünnen
- Entfernung nicht sichtbarer Objekte
- Große Szenen: Anzahl sichtbarer  
Objekte  $\ll$  Anzahl aller Objekte
- Zielsetzung
  - Zeitersparnis bei der  
Schnittpunktberechnung
  - Entfernung offensichtlich nicht  
sichtbarer Objekte

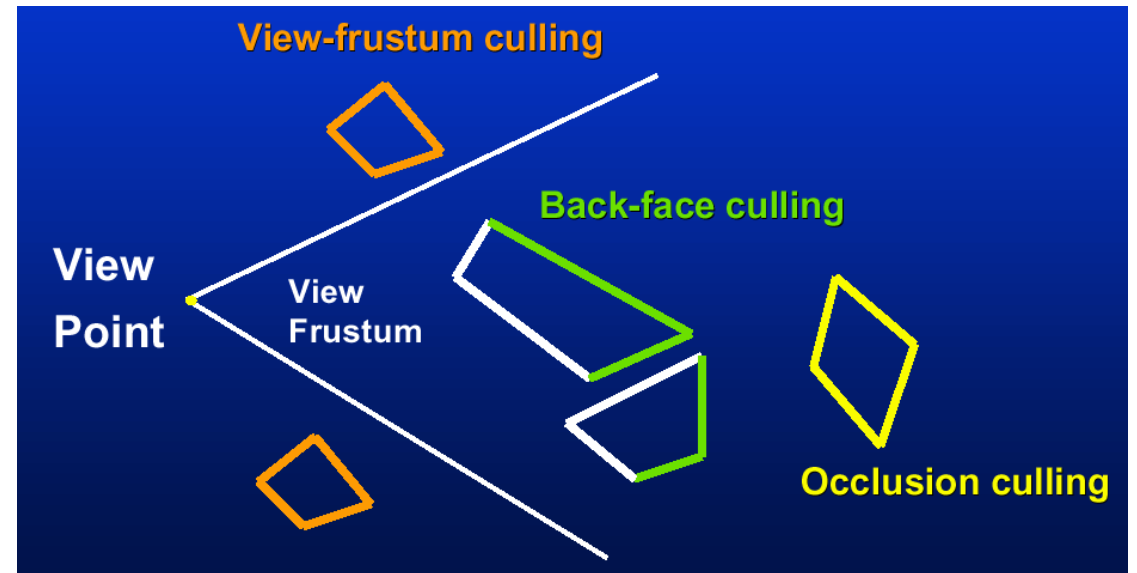
## 8.5 Culling

- heuristische, grobe Antwort
- Test muss erheblich „billiger“ sein als Test der Sichtbarkeit
- Objektraum- und Bildraumverfahren
- Bounding Volumes

## 8.5 Culling

### Arten

- Backface-Culling  
Normale zeigt nach hinten  
(wir sehen die Rückseite)
- View-Frustum-Culling  
Das Objekt liegt außerhalb des  
Blickfeldes
- Occlusion-Culling  
Das Objekt liegt hinter einem  
anderem Objekt



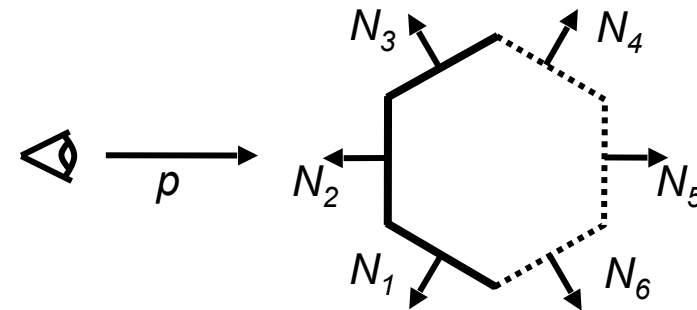
## 8.5 Culling

### Backface-Culling

- Rückseiten von undurchsichtigen Objekten nicht sichtbar
- Orientierung über Normalen kodiert: konsistente Berechnung wichtig
- Bei Inkonsistenzen: Löcher in Objekten
- Wird von OpenGL unterstützt
- Sehr einfache Operation

### - Klassifikation der Rückseiten

- Normalenvektoren  $N_i$  aller Flächen
- Normalenvektor  $N_i$  zeigt in Blickrichtung: Rückseite
- Berechne Skalarprodukt aus Blickrichtungsvektor  $p$  und Normale  $N_i$
- Verwirf, wenn  $p \cdot N_i > 0$





## 8.5 Culling

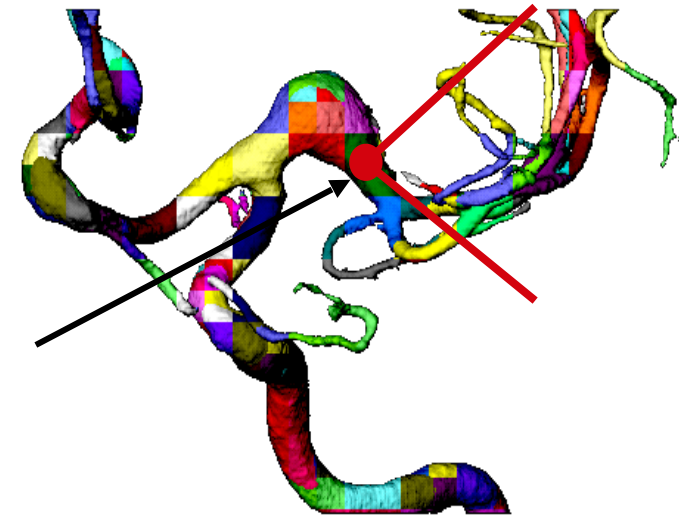
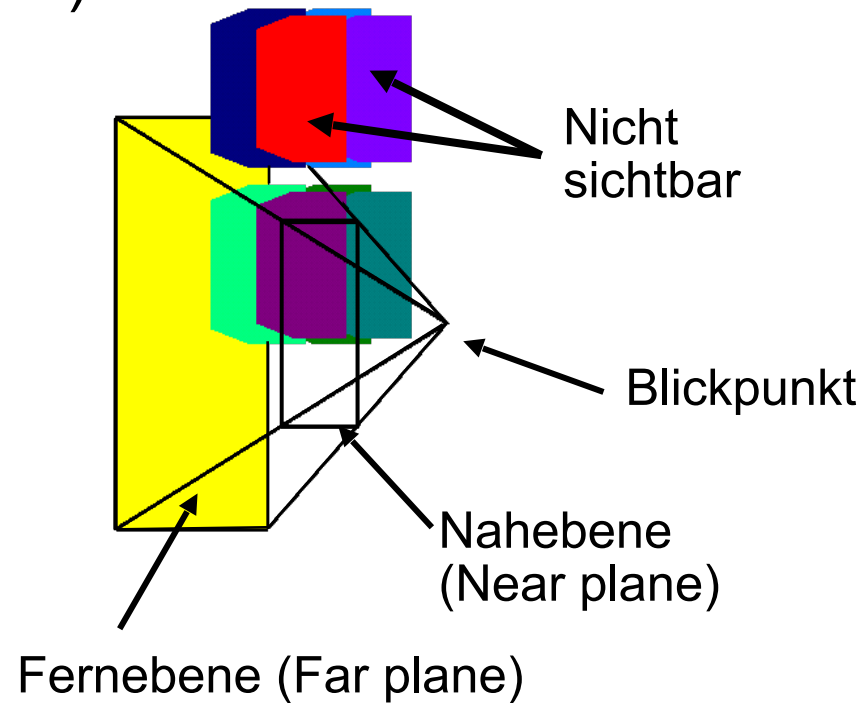
### Backface-Culling: Eigenschaften

- Anzahl der Objektpolygone wird durch Entfernen der Rückseiten durchschnittlich um etwa die Hälfte reduziert
- Aufwand zur Berechnung des Skalarprodukts sehr gering
- Besteht die Szene nur aus einem einzelnen konvexen Polyeder, so löst Backface-Culling bereits das Sichtbarkeitsproblem!
- Bei konkaven Polyedern oder Szenen, an denen mehrere Objekte beteiligt sind, kann es zu Selbst- und/oder Fremdverdeckung kommen
  - Hier werden aufwändigere Verfahren benötigt

## 8.5 Culling

### View-Frustum-Culling

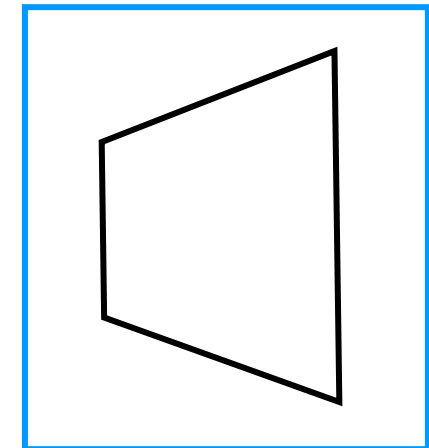
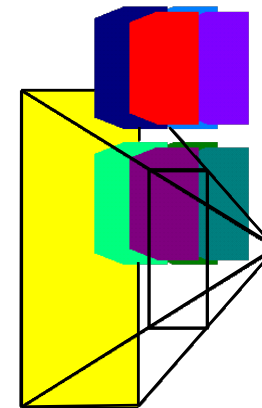
- Blickfeldtest
- Liegt Objekt (Bounding Volume) im Sichtvolumen (View Frustum)?



## 8.5 Culling

### View-Frustum-Culling

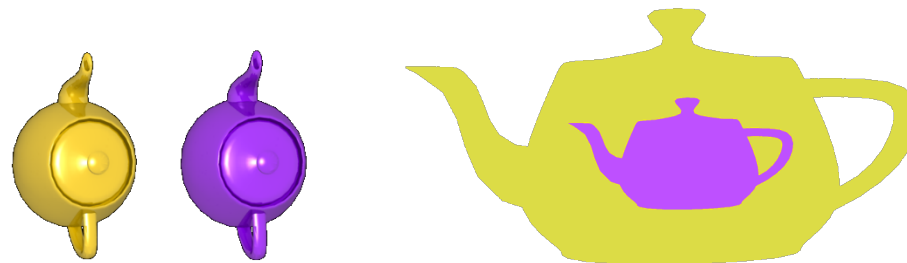
- Berechnung
  - Schneide Bounding Volume mit dem View-Frustum
  - Einfach nach perspektivischer Transformation (Einheitswürfel)
  - Sonderfall: Bounding Volume umfasst View-Frustum
  - Hierarchischer oder linearer Test



## 8.5 Culling

### Occlusion-Culling

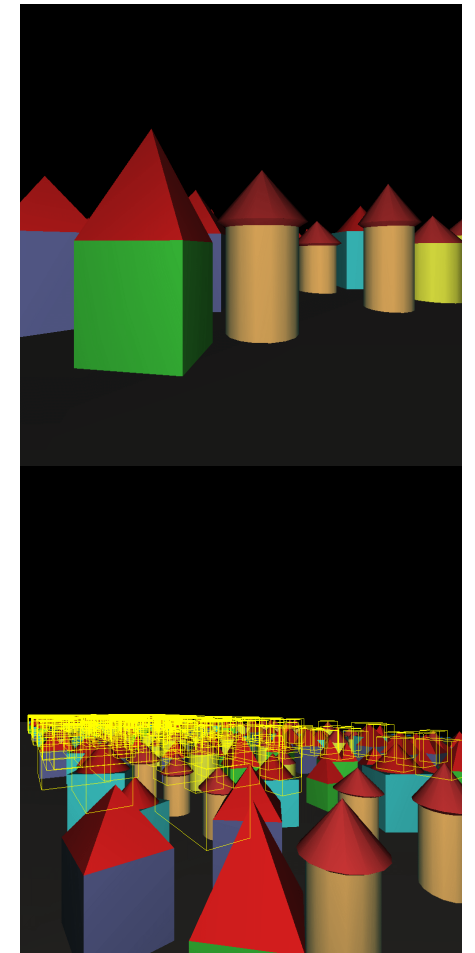
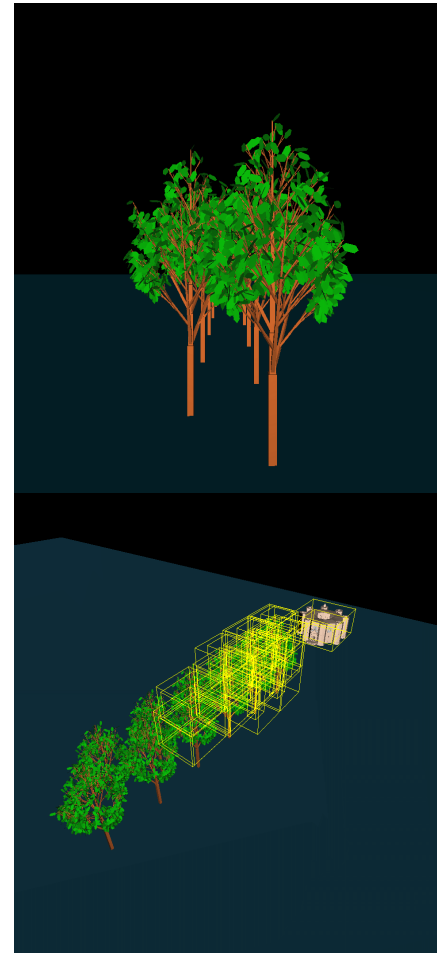
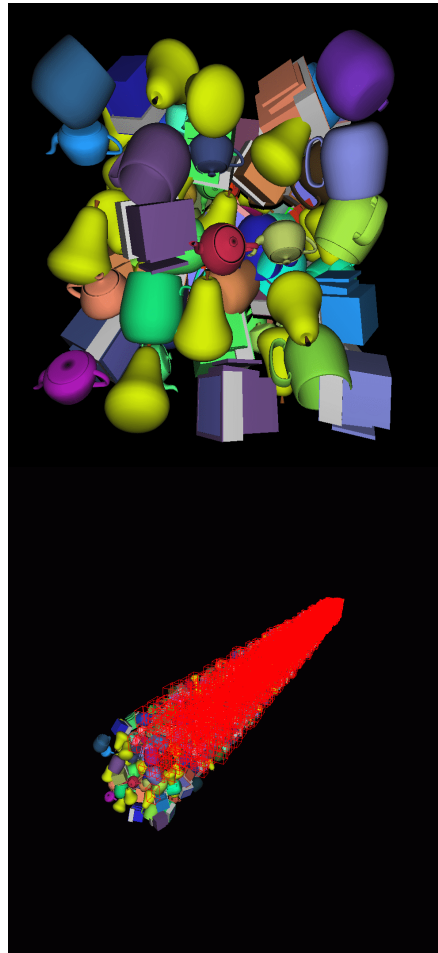
- Ist Geometrie von Geometrie verdeckt?



## 8.5 Culling

### Occlusion-Culling

- Gelbe/rote Bounding Volumes sind verdeckt



## 8.5 Culling

### Occlusion-Culling

- Verdeckungstest
  - Wie wird Verdeckung erkannt?
  - Wo ändert sich etwas, wenn man das Objekt darstellen würde
- Objektraum und Bildraumverfahren
  - Hierarchical Z-Buffer (Greene 1993)
  - Hierarchical Occlusion Maps (Zhang 1997)
  - Virtual Occlusion Buffer (Bartz 1999)

## 8.5 Culling

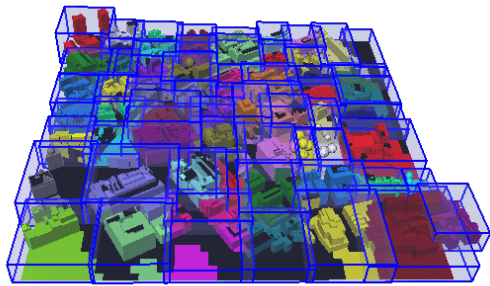
### Occlusion-Culling

- Objekte von vorne nach hinten verarbeiten (Tiefensortierung)
- Verwende Bounding Volumes
  - Wenn die Hülle nicht sichtbar ist, dann auch nicht die in ihr eingeschlossene Geometrie
- Konservativ
  - Nicht exakt, aber immer auf der sicheren Seite
  - Objekte, die auf jeden Fall nicht sichtbar sind
  - Sonst darstellen
- Quantitativ/Approximativ
  - Nicht konservativ
  - Wenn nur ein geringer Anteil sichtbar ist, als verdeckt behandeln

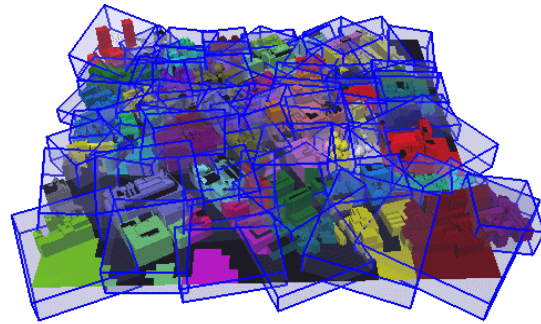
## 8.5 Culling

### Occlusion-Culling

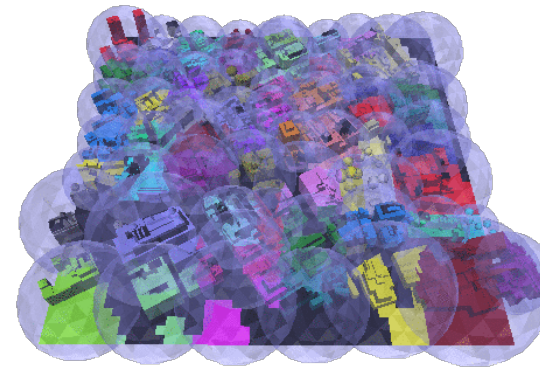
- Optimiere Hüllvolumen: Reduziere Falsch-Positive [Bartz 2001]



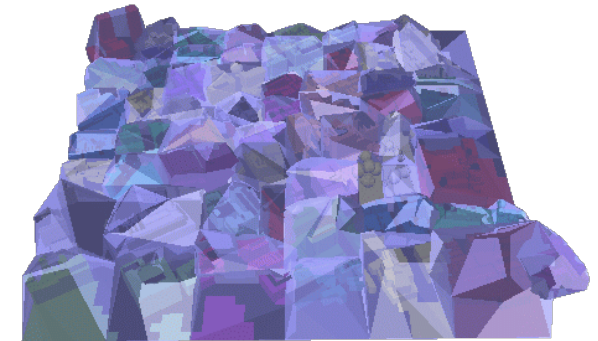
AABB



OBB



Bounding Spheres



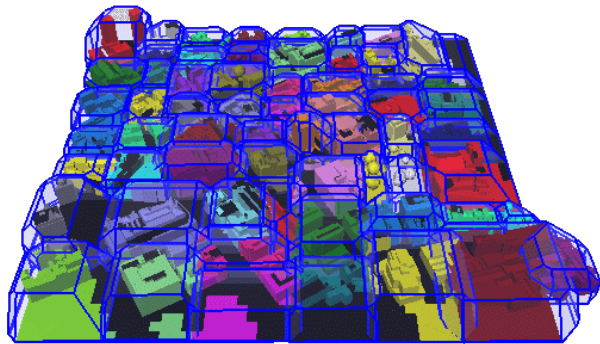
Konvexe Hüllen



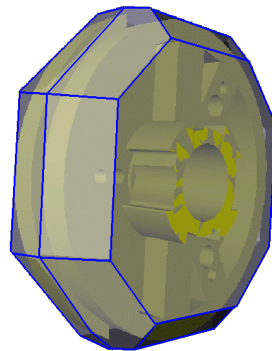
## 8.5 Culling

### Occlusion-Culling

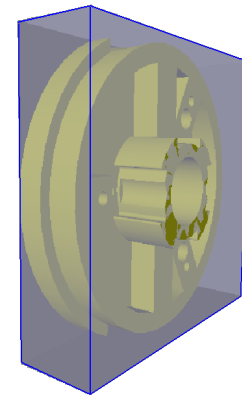
- Optimiere Hüllvolumen: Reduziere Falsch-Positive [Bartz 2001]



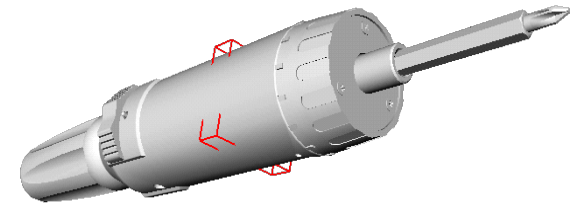
k-DOPs



26-dop



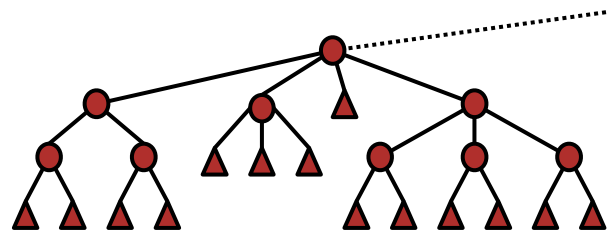
AABB



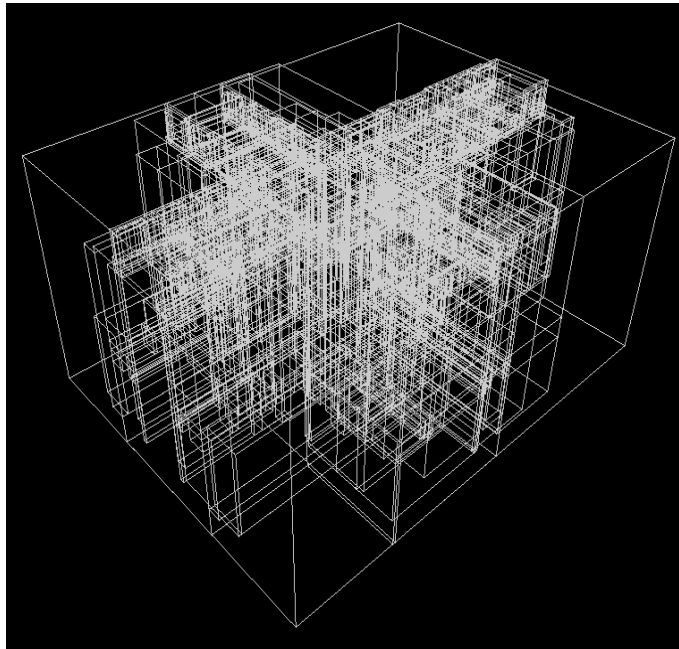
## 8.5 Culling

### Occlusion-Culling

- Hierarchische Organisation
  - AABB-Hierarchie
  - Kathedrale



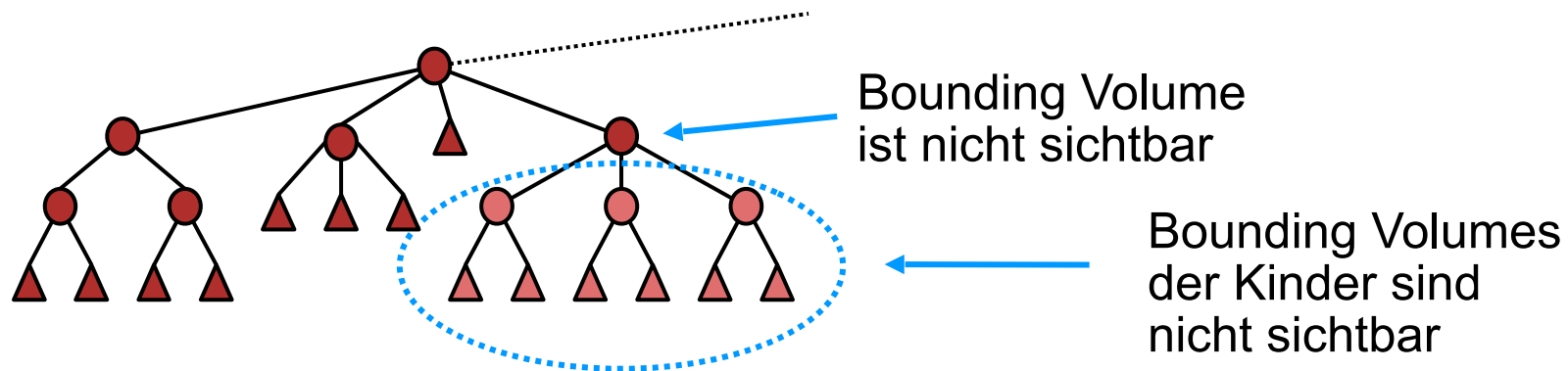
- Innere Knoten
- ▲ Geometrieblätter



## 8.5 Culling

### Occlusion-Culling

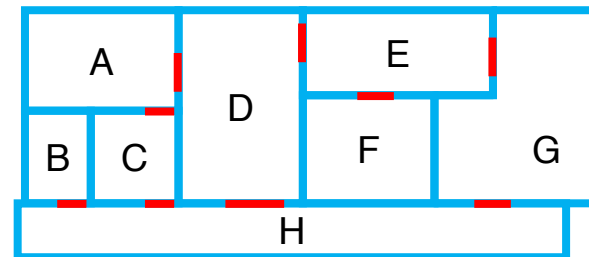
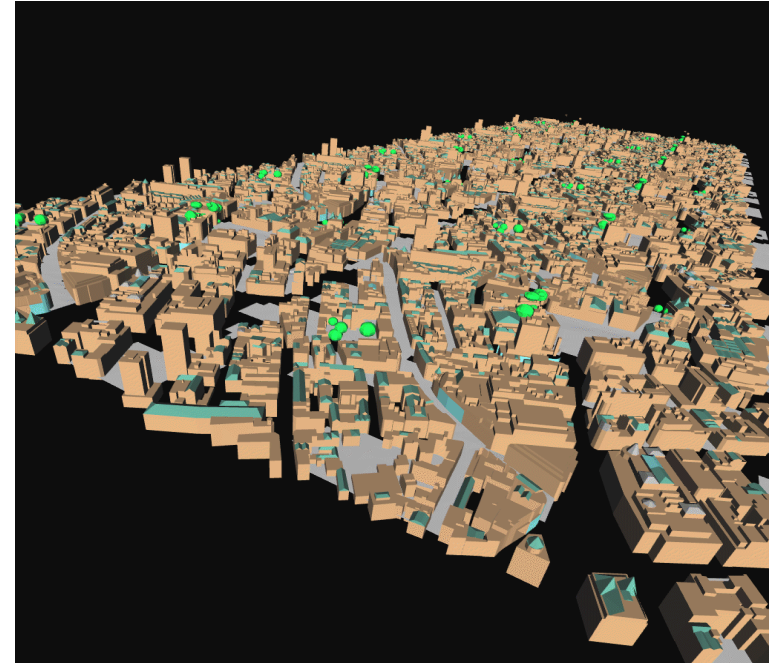
- Hierarchisches Culling testet Baum beginnend an der Wurzel
  - Nicht sichtbar: entferne Knoten
  - Sichtbar: stelle Objekte im Teilbaum dar
    - teste Kindknoten



## 8.5 Culling

### Occlusion-Culling

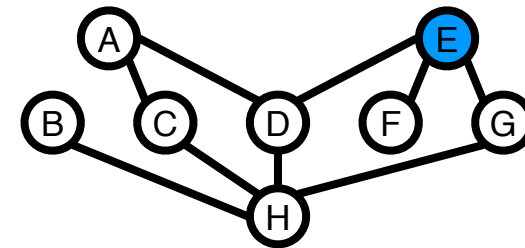
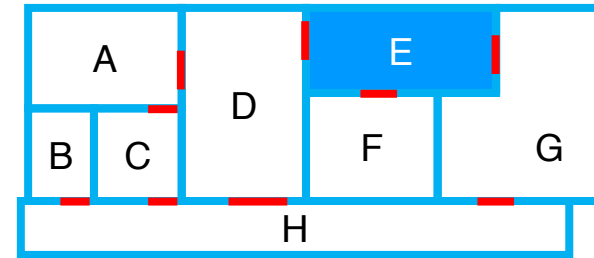
- Verdeckung in Stadtszenen
- Häuser haben feste Zimmer und Öffnungen:  
Cells and Portals
- Sichtbarkeit durch Portale bestimmt: Potentially Visible Sets (PVS)



## 8.5 Culling

### Occlusion-Culling

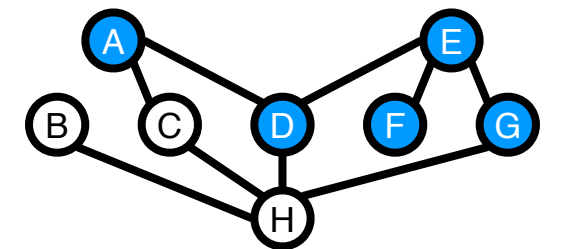
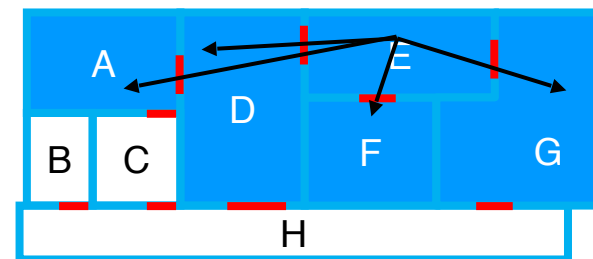
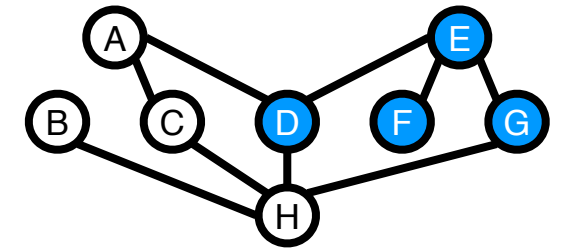
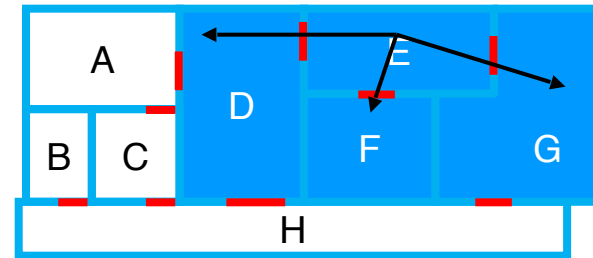
- Zellen und Portale
  - [Teller 1992, Luebke 1995]
  - Berechne Nachbarschaftsgraph zwischen Zellen (durch Portale)
  - Zelle ist nur dann sichtbar, wenn sie durch (eine Folge von) Portale(n) sichtbar ist: Sichtlinie



## 8.5 Culling

### Occlusion-Culling

- Zellen und Portale
  - Speichere für jede Zelle alle durch Portale verbundenen Zellen in einem Graphen
  - Assoziiere alle Objekte in einer Zelle mit ihrem Knoten im Graphen
  - Ist vom Betrachter aus ein Portal sichtbar, muss die zugehörige Zelle mit berechnet werden
  - Iterative Analyse



## 8.5 Culling

### Occlusion-Culling

- Für allgemeines OC sind Softwareverfahren zu teuer
  - 1998: Einführung HP OpenGL Extension Occlusion-Flag (fx4)
  - 2001: NVIDIA OpenGL Extension Occlusion-Query (NV 20)
  - 2003: Occlusion-Queries in OpenGL 1.5
- Hardware-Tests
  - Occlusion-Flag: Binäre Antwort, ob etwas sichtbar wird
  - Occlusion-Queries: Quantitative Antwort, wieviel sichtbar wird

## 8.5 Culling

### Occlusion-Culling

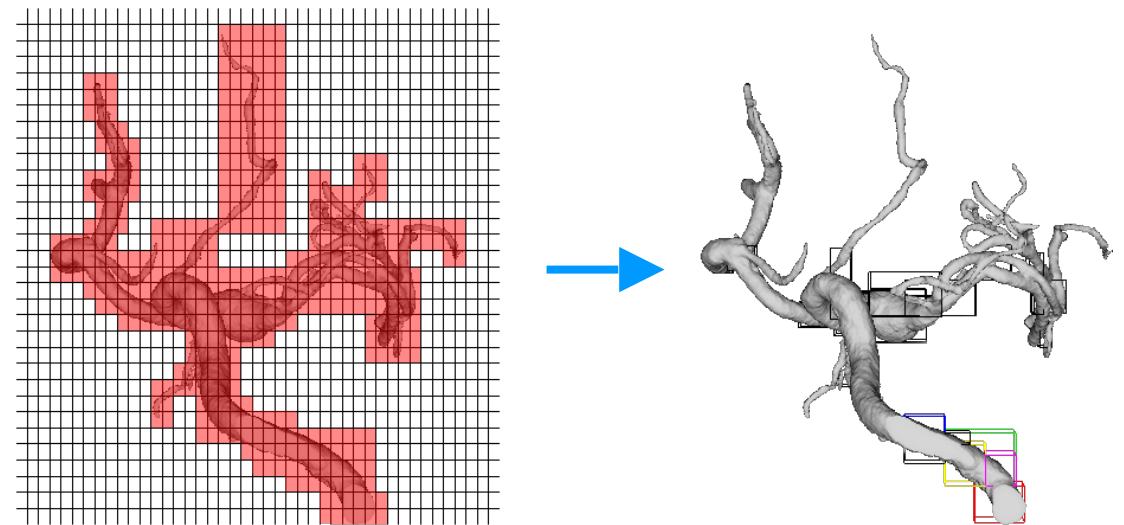
- Stop-and-Wait-Paradigma
  - Pro Frame, pro Objekt:
    - Teste Bounding Volume
    - Warte Occlusion-Antwort ab
    - Wenn sichtbar, zeichne assoziierte Geometrie
    - Sonst: entferne Objekt (culling)
  - Warten (Pipeline-Flush) führt zu Pipeline-Stall
- Multiple Queries
  - Gebündelte Queries
  - Wartet nicht auf Zwischenergebnisse
    - Kein Pipeline-Stall
  - Nutzt aber auch keine Zwischenergebnisse aus
  - Gesparte Stalls viel billiger als zu viel gerenderte Geometrie
    - 1/10, 1/20
  - Also: Vorsortierung in gegenseitig unabhängige Objekte (sich nicht gegenseitig verdeckend)



## 8.5 Culling

### Occlusion-Culling

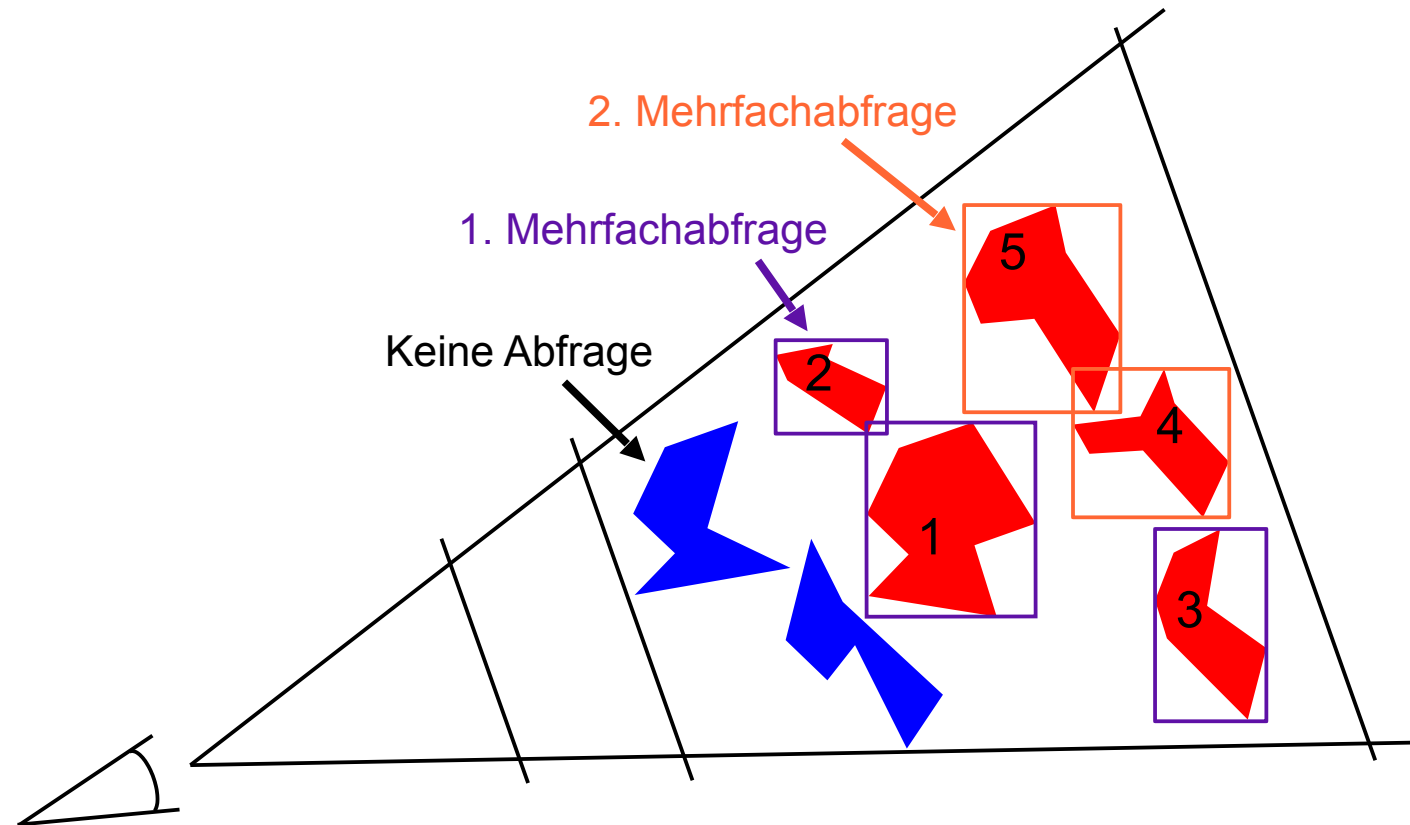
- Vorsortierung in gegenseitig unabhängige Objekte
  - OccupancyMap Hierarchie [Staneker 2003]
    - Nutze Kohärenz
    - Reduziere Falsch-Positive
    - Günstiger Vorab-Overlap-Test
    - Spart auch redundante Occlusion-Queries



# 8.5 Culling

## Occlusion-Culling

- Überlappungsfreie Vorsortierung



## Quellen

- Computergraphik, Universität Leipzig  
(Prof. D. Bartz)
- Graphische Datenverarbeitung I, Universität Tübingen  
(Prof. W. Straßer)
- Graphische Datenverarbeitung I, TU Darmstadt  
(Prof. M. Alexa)