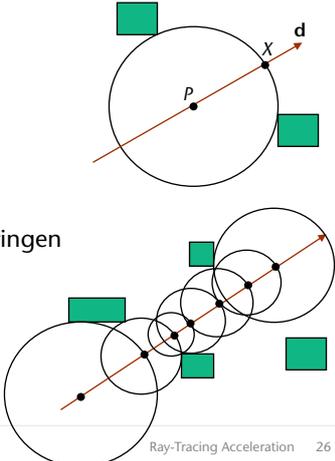


Proximity Clouds [1994]

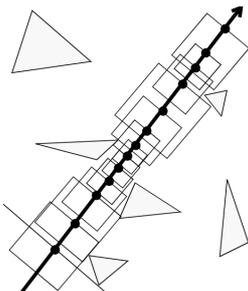
- Gedankenexperiment:
 - Annahme: wir stehen auf dem Strahl an Punkt P, und wissen, daß sich in einer Kugel um P mit Radius r kein Objekt befindet
 - Dann können wir direkt zum Punkt

$$X = P + \frac{r}{\|d\|}d$$
 weiterspringen
 - Annahme: diesen "clearance" Radius wissen wir in jedem Punkt des Raumes
 - Dann kann man von Punkt zu Punkt springen



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 26

- Das funktioniert genau so mit jeder anderen Metrik
- Problem: man kann nicht in **jedem** Punkt diese Info speichern
- Idee: diskretisiere den Raum
 - Speichere für jede Gitterzelle, wieviele (leere) Zellen man überspringen kann, unabhängig von der Strahlrichtung
- Beispiel:



	1	1	1	1					
	2	2	2	2					
	3	3	3	3					
	4	4	4	3					
		3	3						
			1	1	1				

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 27

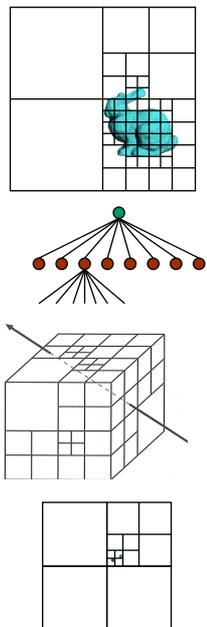
Allgemeine Regeln zur Optimierung

- "Premature Optimization is the Root of All Evil" [Knuth]
 - Erst naiv und langsam implementieren, dann optimieren!
 - Nach jeder (möglichst kleinen) Optimierung einen Benchmark machen!
 - Manchmal/oft stellen sich "Optimierungen" als Verlangsamungen heraus
 - Vor einer Optimierung Profiling machen!
 - Oft wird 80% der Zeit wo ganz anders verbraten
 - Erst nach schlaueren / einfacheren / effizienteren Algos suchen, dann "Bit-Knipsereien" betreiben

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 28

Octree / Quadtree

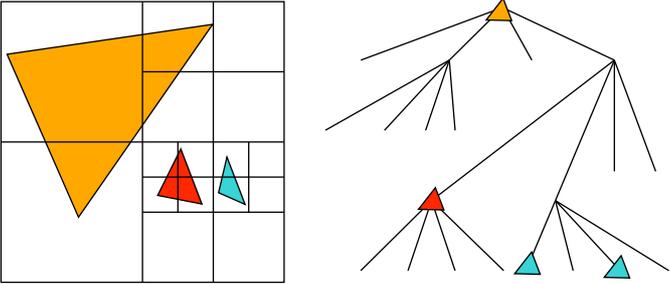
- Idee: extreme Variante der rekursiven Gitter
- Aufbau:
 - Mit BBox der gesamten Szene beginnen
 - Voxel in 8 gleiche Sub-Voxels rekursiv unterteilen
 - Abbruchkriterien: Zahl der restlichen Primitive und maximale Tiefe
- Vorteil: lässt große Traversal-Schritte in den leeren Regionen zu („empty space skipping“)
- Nachteile:
 - Rel. komplizierte Traversalalgorithmen
 - Benötigt manchmal sehr viele Unterteilungen zur Auflösung versch. Objekte



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 29

Primitive in adaptiven Gittern / Octrees

- Leben jetzt auf inneren Levels, oder ...
- Nur in Blättern, aber dann mehrfach vorhanden



Octree/(Quadtree)

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 30

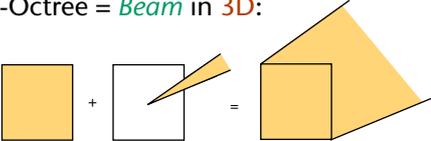
5D-Octree für Strahlen

[Arvo u. Kirk 1987]

- Was ist ein Strahl?
 - Punkt + Richtung = 5-dim. Objekt
- Octree über Menge aller Strahlen:
 - Richtungswürfel D
 - Bidirektionale Abbildung für Richtungen:

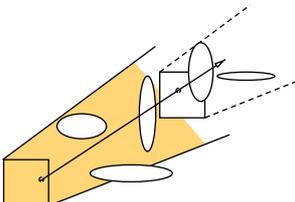
$$S^2 \leftrightarrow D := [-1, +1]^2 \times \{\pm x, \pm y, \pm z\}$$
 - Alle Strahlen im Universum $U = [0, 1]^3$:

$$R = U \times D$$
- Knoten eines 5D-Octree = *Beam* in 3D:



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 31

- Aufbau (6x):
 - Assoziiere Objekt mit Knoten \leftrightarrow Objekt schneidet Beam
 - Start mit Wurzel = $U \times [-1, +1]^2$ und Menge aller Objekte
 - Teile Knoten (in 32 Kinder) wenn
 - zu viele Objekte, *und*
 - zu große Zelle.
 - Ordne Objekte den Kindern zu
- Strahltest:
 - Konvertiere Strahl in 5D-Punkt
 - Finde Blatt des Octree
 - Schneide Strahl mit assoziierten Objekten
- Optimierungen...



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 32

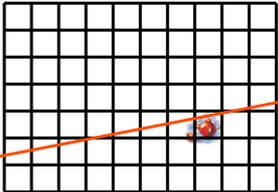
Bemerkungen

- Die Methode führt im Prinzip eine approximierende Vorberechnung der Visibility für die komplette Szene durch
 - Was ist von jedem Punkt in jede Richtung sichtbar?
- Sehr teure Vorberechnung, billiges Traversal
 - Unangemessener Kompromiss zwischen Precomputation und Laufzeit
- Speicherhungrig, sogar mit *lazy evaluation*
- Wird selten in der Praxis verwendet

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 33

kD-Trees

- Problem der Gitter: "teapot in a stadium"
- Problem der Octrees:
 - zu starr bei der Platzierung der Unterteilung (immer Mittelpunkt)
 - Unterteilung in allen Richtungen nicht immer nötig
- Lösung: hierarchische Raumunterteilung, die sich an die "Verteilung" der Geometrie lokal und möglichst flexibel anpasst
- Idee: rekursive Raumunterteilung durch **eine** Ebene:
 - Unterteile gegebenes Teilvolumen mit einer Ebene
 - Wähle Ebene senkrecht zu einer Koordinatenachse, aber sonst beliebig
- „Best known method“ [Siggraph Course 2006]
- ... jedenfalls für statische Szenen



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 34

- **Informelle Definition:**
 - Binärer Baum:
 - Blätter: enthalten einzelne Objekte oder Objektliste
 - Innere Knoten: *Splitting Plane* (senkrecht zu einer Achse) und Kind-Pointer
 - Abbruchkriterium:
 - Maximale Tiefe, Zahl der Objekte, Kostenfunktion, ...
- **Vorteile:**
 - Adaptiv
 - Kompakt (nur 8 Bytes pro Knoten notwendig)
 - Einfacher und schneller Traversal
- **Kleiner Nachteil:**
 - Polygone müssen oft mehrfach im Baum gespeichert werden

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 35

Beispiel

[Slide courtesy Martin Eisemann]

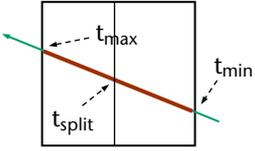
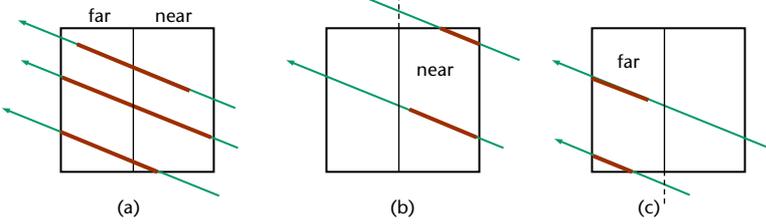
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 36

3D-Beispiel

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 37

Ray-Traversal in einem kd-Tree

- Schneide Strahl mit Root-Box $\rightarrow t_{\min}, t_{\max}$
- Rekursion:
 - Schneide Strahl mit Splitting Plane $\rightarrow t_{\text{split}}$
 - Fallunterscheidung:
 - a) Erst "near", dann "far" Teilbaum traversieren
 - b) Nur "near" traversieren
 - c) Nur "far" traversieren

(a) (b) (c)

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 38

Pseudo-Code für die Traversierung

```

traverse( Ray r, Node n, float t_min, float t_max ):
  if n is leaf:
    intersect r with each primitive in object list,
      discarding those farther away than t_max
    return object with closest intersection point (if any)

  t_split = signed distance along r to splitting plane of n
  near = child of n containing origin of r    // test signs in r.d
  far = the "other" child of n
  if t_split > t_max:
    return traverse( r, near, t_min, t_max )    // (b)
  else if t_split < t_min:
    return traverse( r, far, t_min, t_max )    // (c)
  else:
    t_hit = traverse( r, near, t_min, t_split )
    if t_hit < t_split:
      return t_hit                               // early ray terminat'n
    return traverse( r, far, t_split, t_max )
  
```

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 39

Optimierte Traversierung [1999]

- Beobachtung:
 - 90% aller Strahlen sind Schattenstrahlen
 - Irgendein Hit genügt (nicht notw. der nächste)
- Konsequenz:
 - Reihenfolge des Besuchs der kD-Tree-Kinder ist egal → mache reines DFS
- Idee: Rekursion durch Iteration ersetzen
- Dazu Baum transformieren:

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 40

Algorithmus:

```

traverse( Ray ray, Node root ) :
  stopNode = root.skipNode
  node = root
  while node < stopNode:
    if intersection between ray and node:
      if node has primitives:
        if intersection between primitive and ray:
          return intersection
      node ++
    else:
      node = node.skipNode
  return "no intersection"

```

Diplomarbeit ...

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 41

Aufbau eines kD-Trees

- **Gegeben:**
 - Achsenparallele BBox der Szene ("Zelle")
 - Liste der Geometrieprimitive in dieser Zelle
- **Ablauf:**
 1. Wähle eine achsenparallele Fläche, um die Zelle in zwei aufzuspalten
 2. Verteile die Geometrie auf die beiden Kinder
 - 📄 evtl. einige Polygone (konzeptionell) aufspalten
 3. Rekursion, bis Abbruchkriterium erfüllt ist
- **Bemerkung:** jede Zelle (Blatt oder innerer Knoten) definiert eine Box, ohne daß diese explizit irgendwo gespeichert ist
 - (Theoretisch, wenn man an der Wurzel mit dem **ganzen** Raum startet, können dieses Boxes sogar halb-offen sein)

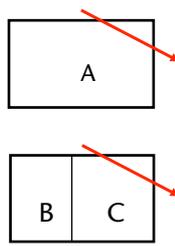
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 43

Ein Abbruchkriterium

- Wie trifft man die Entscheidung, ob sich eine weiterer Split lohnt?
- Betrachte die Kosten beim Strahltest für 2 Fälle:
 - Kein Split \rightarrow Kosten = $t_i N$
 - Split \rightarrow Kosten = $t_t + t_i(p_B N_B + p_C N_C)$

wobei t_i = Zeit für 1 Schnitttest Strahl-Primitiv
 t_t = Zeit für 1 Schnitttest Strahl-Split-Ebene eines kd-Knoten
 p_B = Wahrscheinlichkeit, daß Strahl Zelle B trifft
 N = Anzahl Primitive

- Vereinfachende Annahmen dabei:
 - $t_i = \text{const}$ für alle Primitive
 - $t_i : t_t = 80 : 1$ (festgestellt durch Experimente)
- p_B werden wir später ermitteln



$$p_B \propto \frac{S_B}{S_A}$$

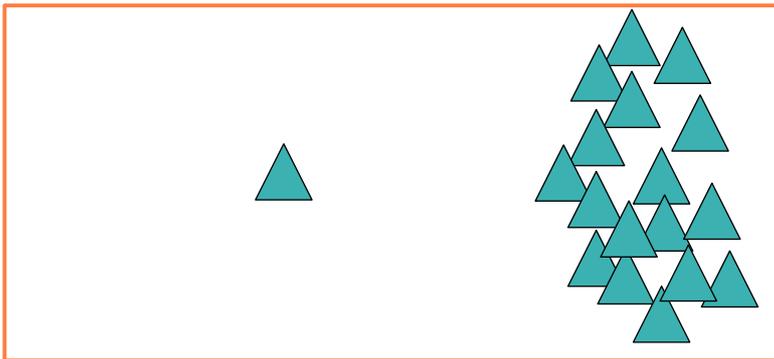
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 44

Zur Wahl der Splitting-Plane

- **Naïve Wahl der Splitting-Plane:**
 - **Split-Achse:**
 - Round Robin (x, y, z, x, ...)
 - Die längste Achse teilen
 - **Split-Position:**
 - Mitte der Zelle
 - Median der Geometrie
- **Besser: verwende Kostenfunktion**
 - Kostenfunktion sollte die **erwarteten** Kosten eines Strahltests auf beide Teilbäume **gleichmäßig** verteilen
 - Probiere alle 3 Achsen
 - Suche entlang jeder Achse das Minimum
 - wähle die Achse und Split-Position mit dem kleinsten Minimum

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 45

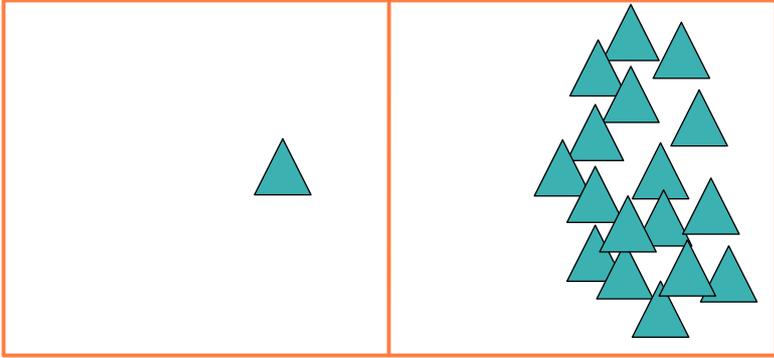
Motivation der Kostenfunktion



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 46




▪ Split in der Mitte:

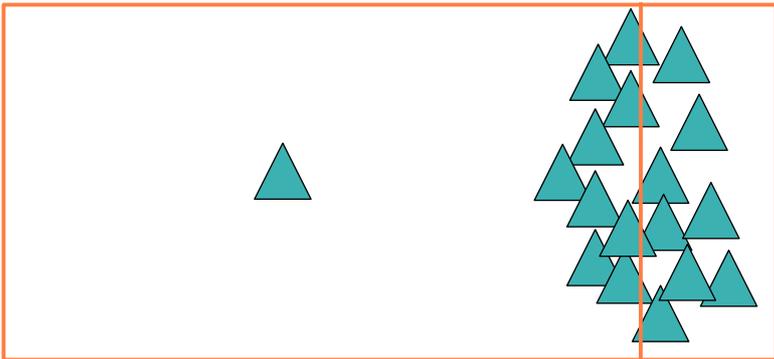


▪ Wahrscheinlichkeit, dass Strahl links oder rechts durchgeht ist gleich
 ▪ Erwartete Kosten für linkes oder rechtes Kind sind sehr verschieden!

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 47



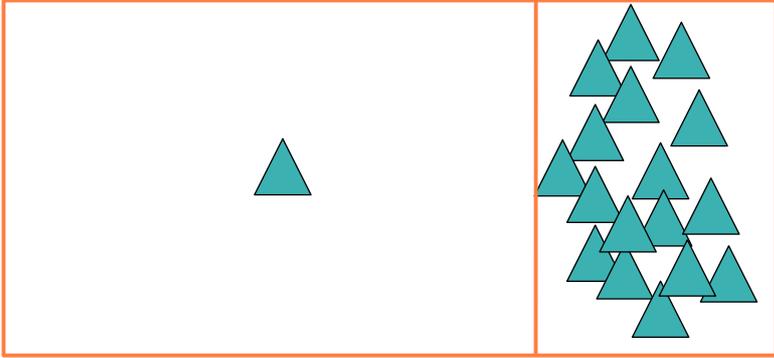

▪ Split am Median:



▪ Zeitaufwand links und rechts gleich, nicht aber die Wahrscheinlichkeit eines Hits

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 48

▪ Kosten-optimierte Heuristik:



▪ Ungefähr gleiche erwartete Kosten

- Wahrscheinlichkeit für Hit links größer, dafür sind dort weniger Polygone

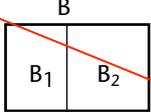
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 49

Die Surface-Area-Heuristic (SAH) [1990]

▪ Frage: Wie misst man die Kosten eines gegebenen kd-Trees?

▪ Erwartete Kosten eines Strahltests:

- bei der Traversierung ist man bei Zelle B angekommen
- Zelle B habe Kinder B_1, B_2
- Erwartete Kosten (\sim Zeit):

$$C(B) = P[\text{Schnitt mit } B_1] \cdot C(B_1) + P[\text{Schnitt mit } B_2] \cdot C(B_2)$$


▪ Annahmen im folgenden:

- alle Strahlen haben denselben, weit entfernten Ursprung
- alle Strahlen treffen das Root-BV des kd-Tree

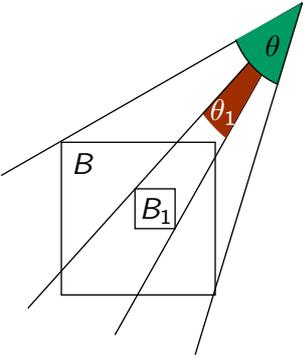
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 51

- Wahrscheinlichkeit:

$$P[\text{Schnitt mit } B_1 \mid \text{Schnitt mit } B] = \frac{\theta_1}{\theta} \approx \frac{\text{Area}(B_1)}{\text{Area}(B)}$$
 wobei $\frac{\theta}{\theta_1}$ der von B bzw. B_1 aufgespannte Raumwinkel ist
- Erklärung: bei einer Kugel ist

$$A = 4\pi r^2$$
 und wenn Ursprung der Strahlen weit entfernt, dann ist

$$r \sim \sin(\theta) \approx \theta$$



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 52

- Auflösung der "rekursiven" Formel:
 - Wie berechnet man $C(B_1)$ bzw. $C(B_2)$?
 - Einfache Heuristik: setze

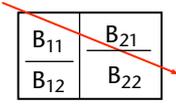
$$C(B_i) \approx \text{Anzahl Dreiecke in } B_i$$
- Die Surface-Area-Heuristic komplett:
minimiere beim Aufteilen der Menge der Polygone die Funktion

$$C(B) = \text{Area}(B_1) \cdot N(B_1) + \text{Area}(B_2) \cdot N(B_2)$$

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 53

■ **Achtung:** für andere Queries (z.B. Punkte, Boxes,...) ist die Fläche **kein** Maß für die Wahrscheinlichkeit!

■ Naheliegende, verbesserte(?) Heuristik: mache „Look-Ahead“

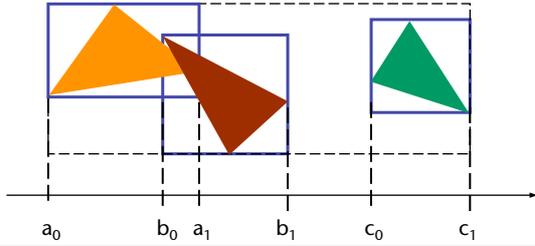
$$\begin{aligned}
 C(B) &= P[\text{Schnitt mit } B_1] \cdot C(B_1) \\
 &\quad + P[\text{Schnitt mit } B_2] \cdot C(B_2) \\
 &= P[B_1] \cdot (P[B_{11}]C(B_{11}) + P[B_{12}]C(B_{12})) \\
 &\quad + P[B_2] \cdot (P[B_{21}]C(B_{21}) + P[B_{22}]C(B_{22})) \\
 &\quad \dots
 \end{aligned}$$


Diplomarbeit ...

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 54

Bemerkungen

- Es genügt, die Kostenfunktion (SAH) nur an einer endlichen Folge von Stellen auszuwerten
 - Nämlich an den Rändern der Bboxes der Dreiecke
 - Dazwischen ist der Wert der SAH auf jeden Fall schlechter
- Alle Ränder aller Elementar-BBoxes sortieren, SAH nacheinander an diesen Stellen auswerten (*plane sweep*)
- Sortieren erlaubt Intervallhalbierung und schnellere Auswertung



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 55

Speicherung eines kd-Tree

- Daten pro Knoten:
 - Flag, ob innerer Knoten oder Blatt (bool)
 - Falls innerer Knoten:
 - Split-Achse (uint),
 - Split-Position (float),
 - 2 Zeiger auf Kinder (2 pointer)
 - Falls Blatt:
 - Anzahl Primitive (uint)
 - Liste der Primitive (pointer)
- Naïve Implementierung: 16 Bytes + 3 Bits — sehr **Cache-ineffizient**
- Optimierte Implementierung:
 - 8 Bytes (!)
 - Bringt 20% Speedup (manche berichten sogar Faktor 10!)

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 58

- Idee der optimierten Speicherung: Daten überlagern
- Fasse alle Flags in 2 Bits zusammen
- Überlagere Flags, Split-Position, Anzahl Primitive

innerer Knoten →	s 1	exponent 8	mantissa 23	{ 00 = "Blatt" 01 = "X-Achse" 10 = "Y-Achse" 11 = "Z-Achse"	
Beide →					flags 2
Blatt →	Anzahl 30				

```

union
{
  unsigned int m_flags;    // both
  float m_split;         // inner node
  unsigned int m_nPrims;  // leaf
};
  
```

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 59

- Für innere Knoten: nur 1 Zeiger auf Kinder
 - Verwalte eigenes Array von kd-Knoten (nicht `malloc()` oder `new`)
 - Speichere beide Kinder in aufeinanderfolgende Array-Zellen; oder
 - speichere eines der Kinder direkt hinter dem Vater.
- Überlagere Zeiger auf Kinder mit Zeiger auf Primitive
- Zusammen:


```
class KdNode
{
private:
  union {
    unsigned int m_flags;      // both
    float m_split;           // inner node
    unsigned int m_nPrims;    // leaf
  };
  union {
    unsigned int m_rightChild; // inner node
    Primitive * m_onePrim;     // leaf
    Primitive ** m_primitives; // leaf
  };
};
```

Falls `m_nPrims == 1`

→

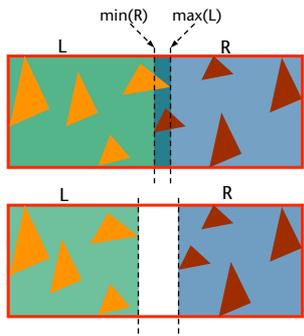
Falls `m_nPrims > 1`

→

- Achtung: Zugriff auf Instanzvariablen natürlich nur noch über Kd-Node-Methoden!
 - Z.B.: beim Schreiben von `m_split` muß man darauf achten, daß danach (nochmals) `m_flags` geschrieben wird (ggf. mit dem ursprünglichen Wert)!
 - Beim Schreiben/Lesen von `m_nPrims` muß ein Shift durchgeführt werden!

Spatial KD-Trees (SKD-Tree) [1987/2006]

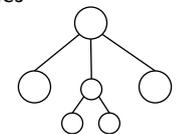
- Variante der kd-Trees
- Andere Namen: "Bounding Interval Hierarchy" (BIH), BoxTree
- Unterschied:
 - 2 parallele Splitting Planes pro Knoten
 - Alternative: die 2 Splitting Planes dürfen verschieden orientiert sein
- Vorteil: "straddling" Polygone brauchen nicht mehr in beide Teilbäume aufgenommen werden
 - Bei kd-Trees hat man ca. $2 \cdot 3 \cdot N$ Zeiger auf Dreiecke, $N = \text{Anzahl Dreiecke in der Szene}$
- Nachteil: Überlappung \rightarrow man kann Traversierung nicht stoppen, wenn man Hit im "near" Teilbaum gefunden hat



G. Zachmann Computer-Graphik 2 - SS 08
Ray-Tracing Acceleration 62

Oversized Objects

- Problem:
 - manchmal sind die Größen der Dreiecke sehr verschieden (z.B. Architektur-Modelle)
 - Diese erschweren das Finden von guten Splitting-Planes
- Lösung: ternärer Baum
- Aufbau:
 - Vor jedem Splitting: filtere "oversized objects" heraus
 - Falls viele "oversized objects": baue eigenen kd-Tree
 - Sonst: einfache Liste

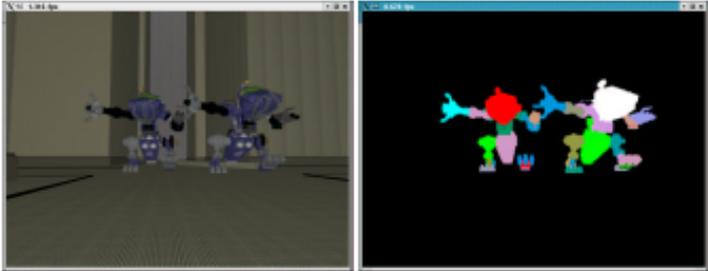




G. Zachmann Computer-Graphik 2 - SS 08
Ray-Tracing Acceleration 63

Zwei-stufige Datenstrukturen

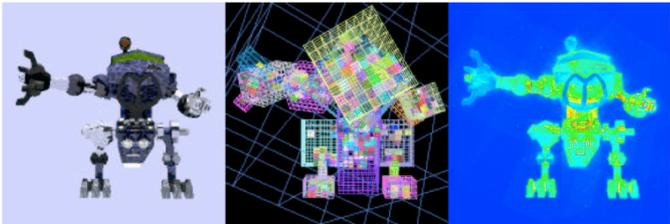
- Beobachtung:
 - Oft ist nur ein Teil der Szene dynamisch
 - Die dynamischen Teile sind oft sog. "*articulated bodies*", d.h., sie bestehen aus starren, miteinander beweglich verbundenen Teilen (z.B. Roboter)



G. Zachmann Computer-Graphik 2 - SS 08
Ray-Tracing Acceleration 64

Idee:

- Verwende für jedes in sich starre Teil ein eigenes Gitter (oder eine andere DS)
- Verwende ein globales Gitter, in dem die einzelnen Teile als elementare Objekte einsortiert werden
- Bei Bewegung der Figur muß nur dieses globale Gitter aktualisiert werden



Articulated Body
Gitter für jedes Teil
Ray-Tracing-Zeit pro Pixel

G. Zachmann Computer-Graphik 2 - SS 08
Ray-Tracing Acceleration 65

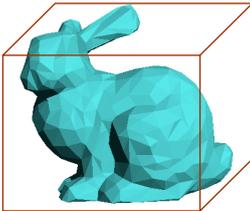
Raumunterteilung vs. Objektunterteilung

- Bisher: Raum wird unterteilt, Objekte (= Dreiecke) den Teilräumen zugeordnet
- Jetzt: Menge der Objekte wird unterteilt, jeder Teilmenge wird ein Bounding Volumen (= Teilraum) zugordnet
- In Wahrheit sind die Grenzen zwischen beiden Verfahren fließend

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 66

Bounding Volumes (BVs)

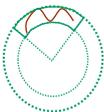
- Komplexe geometrische Objekte bzw. ganze Objektgruppen werden durch "Hüllen" angenähert
- Anforderungen:
 - Die approximierten Objekte müssen vollständig innerhalb des Bounding Volumes liegen
 - Das BV sollte so kompakt wie möglich sein
 - Der Test auf Schnitt mit einem Strahl sollte möglichst schnell berechenbar sein



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 67

Beispiele für Bounding Volumes

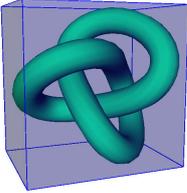
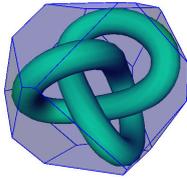
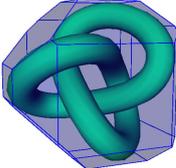
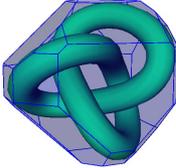
Diplomarbeit ...

 Zylinder [Weghorst et al., 1985]	 Box, AABB (R*-trees) [Beckmann, Kriegel, et al., 1990]	 Konvexe Hülle [Lin et. al., 2001]
 Kugel [Hubbard, 1996]	 Prisma [Barequet, et al., 1996]	 OBB (oriented bounding box) [Gottschalk, et al., 1996]
 Kugelschale (spherical shell) [...]	 k-DOPs / Slabs [Zachmann, 1998]	 Schnitt mehrerer anderer BVs

G. Zachmann Computer-Graphik 2 - SS 08
Ray-Tracing Acceleration 68

k-DOPs

▪ Beispiele:

6-DOP (AABB)		14-DOP	
18-DOP		26-DOP	

G. Zachmann Computer-Graphik 2 - SS 08
Ray-Tracing Acceleration 69

Kosten eines BVs

- **Kosten** einer Schnittpunktberechnung eines Strahls mit einer Teilszene:

$$T = n \cdot B + m \cdot l$$
 - $T =$ gesamte Schnittpunktberechnungskosten
 - $n =$ Anzahl der Strahlen, die gegen das BV getestet werden
 - $B =$ Kosten des Schnittpunkttests mit dem BV
 - $m =$ Anzahl der Strahlen, die das BV schneiden
 - $l =$ Kosten der Tests mit den Objekten der enthaltenen Teilszene
- T soll minimiert werden
- 2 unterschiedliche Anforderungen bei der Wahl eines BVs:
 - einfache BVs (z.B. Kugel, Box) = kleine Schnittkosten B , relativ hohe Strahlentrefferzahlen m
 - komplexe BVs (z.B. exakte konvexe Hülle) = kleines m , hohe Schnittkosten B

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 70

Qualitativer Vergleich:

Better approximation,
higher build and update costs

sphere AABB DOP OBB convex hull

Smaller computational costs
for overlap test

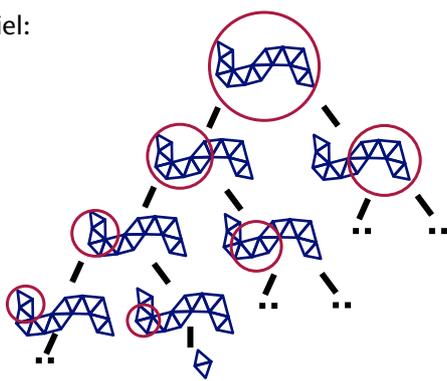
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 71

Bounding Volume Hierarchy (BVH)

- **Definition:**
Eine BVH über einer Menge von Dreiecken \mathcal{P} (oder allg. Primitiven) ist ein Baum, in dem jedem Knoten
 - eine Teilmenge der Primitive aus \mathcal{P} und
 - ein BV \mathcal{B}
 zugeordnet sind, so daß \mathcal{B} \mathcal{P} vollständig einschließt, und so daß \mathcal{B} die BVs aller Kinder einschließt.
- **Bemerkungen:**
 - Man verwendet \mathcal{B} oft auch als Synonym für den Knoten im Baum
 - Primitive werden (üblicherweise) nur an den Kindern gespeichert
 - Ausnahmen können durchaus Sinn machen
 - Üblicherweise ist auch $\mathcal{P} = \mathcal{P}_1 \dot{\cup} \dots \dot{\cup} \mathcal{P}_n$
wobei \mathcal{P}_i die den Kindern zugeordneten Primitive sind

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 72

Schematisches Beispiel:



- **Parameter:**
 - Art des BVs
 - "Arity" (Grad der Knoten)
 - Abbruchkriterium (insbesondere: wieviel Dreiecke pro Blatt)
 - **Aufteilungskriterium** der Primitive (während der Konstruktion)

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 73

Beispiele

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 74

Beispiel für Traversierung einer BVH für Strahlschnitttest

```

    graph TD
      13[13] --> 9[9]
      13 --> 10[10]
      9 --> 1((1))
      9 --> 2((2))
      9 --> 3((3))
      10 --> 11[11]
      10 --> 12[12]
      11 --> 6((6))
      11 --> 7((7))
      11 --> 8((8))
      12 --> 4((4))
      12 --> 5((5))
  
```

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 75

```

graph TD
    13[13] --> 9[9]
    13 --> 10[10]
    9 --> 1[1]
    9 --> 2[2]
    9 --> 3[3]
    10 --> 11[11]
    10 --> 12[12]
    11 --> 6[6]
    11 --> 7[7]
    11 --> 8[8]
    12 --> 4[4]
    12 --> 5[5]
  
```

- Schnitt mit 13 → Ja
 - Schnitt mit 9 → Ja
 - Schnitt mit 1 → Nein
 - Schnitt mit 2 → Nein
 - Schnitt mit 3 → Ja
 - Schnitt mit 10 → Ja, aber weiter entfernt

→ Nur 3 anstatt 8 Tests mit Szenenobjekten, dazu 3 Tests mit BV's

- **Frage:** Wieso haben wir mit BV 9 angefangen?

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 76

Hierarchie-Traversierung nach Kay & Kajiya

- Problem: die Reihenfolge, in der die Knoten beim **reinen DFS** abgearbeitet werden, hängt nur von der Topologie des Baumes ab
- Besser: berücksichtige zusätzlich die **räumliche Lage** der BVs
- Kriterium: Entfernung des Schnittpunktes mit dem BV vom Startpunkt des Strahles (*estimated distance*)

→ Verwendung einer **Priority Queue**

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 77

Algorithmus

- Berechne die Distanz zwischen dem Strahlursprung und dem Schnittpunkt eines Strahls mit dem aktuell besuchten BV
- Ist die Distanz größer als die Distanz zu einem bereits gefundenen Schnittpunkt mit einem Obj, so kann dieses BV und dessen Teilbaum ignoriert werden
- Sonst: Rekursion
- Sortiere alle noch zu testenden BVs gemäß ihrer Distanz zum Strahlursprung in einem Heap
 - Einfügen eines Elementes und Extrahieren des minimalen Elements haben Aufwand von $O(\log n)$
- Als nächster Kandidat wird immer dasjenige BV gewählt, das dem Strahlursprung am nächsten ist

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 78

Beispiel

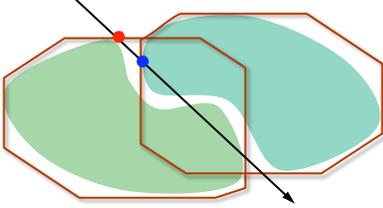
Diagram illustrating the BVH tree structure and intersection tests:

- Schnitt mit 13 → Ja
- 13 herausnehmen
- Schnitt mit 9 → Nein
- Schnitt mit 10 → Ja
- 10 herausnehmen
- Schnitt mit 11 → Ja
- Schnitt mit 12 → Ja
- 12 herausnehmen
- Schnitt mit 4 → Ja
- Schnitt mit 5 → Ja
- 5 herausnehmen, Test mit Primitiv
- 6 herausnehmen, Test mit Primitiv
- 11 herausnehmen ...

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 79

Anmerkungen

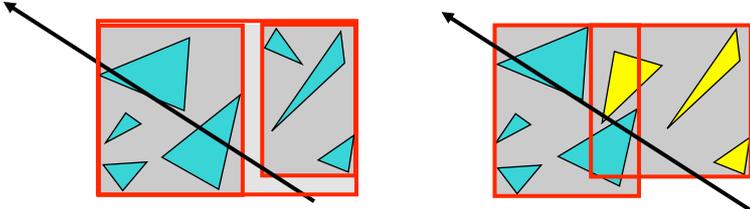
- Achtung:** Der erste gefundene Schnittpunkt mit einem BV liefert nicht unbedingt dasjenige BV, in dem der nächste Schnittpunkt stattfindet!



- Für die Priority Queue ist keine vollständige Ordnung notwendig, da in jedem Schleifendurchlauf nur das Element mit der kleinsten estimated distance benötigt wird
- Effiziente Umsetzung mit einem **Heap**

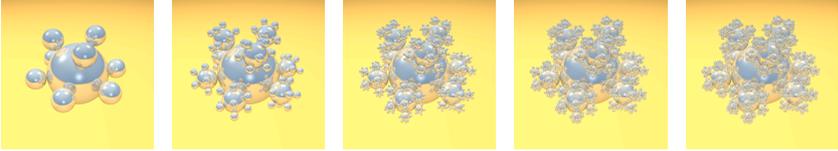
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 80

- Achtung:** man darf auch nicht aufhören, wenn man in einem inneren BV einen Schnitt mit einem Primitiv gefunden hat!
 - Es kann einen **näheren** Schnitt in einem BV mit **größerer estimated distance** geben!
- Beispiel:



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 81

Was bringt es wirklich?



Anzahl Kugeln	10	91	820	7381	66430
Brute-force	2.5	11.4	115.0	2677.0	24891.0
Goldsmith/Salmon	2.3	2.8	4.1	5.5	7.4

Rechenzeiten in Sekunden, Athlon XP 1900+
(Markus Geimer)

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 82

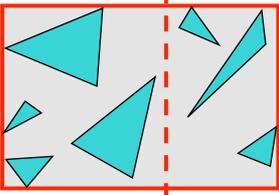
Eigenschaften einer guten Hierarchie

1. Teilbäume der Hierarchie sollten Objekte enthalten, die in der Szene nahe beieinander liegen (*spatial coherence*) Wirklich?
2. Volumeninhalte der einzelnen BVs sollten minimal sein
3. Die Summe der Volumeninhalte der BVs sollte minimal sein
4. Die Bemühungen beim Aufbau der Hierarchie sollten sich auf die oberen Knoten konzentrieren, da durch Abschneiden eines Teilbaums nahe bei der Wurzel mehr Objekte wegfallen als bei einem tieferen Teilbaum
5. Die Zeit zur Berechnung des Bildes durch Raytracing-Verfahren sollte trotz der zusätzlichen Preprocessing-Zeit zum Aufbau der Hierarchie sehr viel geringer werden

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 83

Erzeugung von BV-Hierarchien

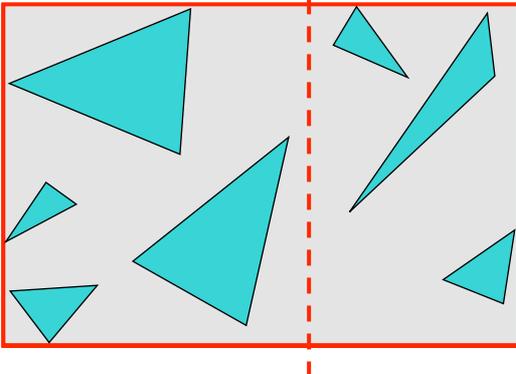
- Durch die Modellierung gegeben (Szenengraph)
- Bottom-up:
 - Rekursives Zusammenfassen von n Objekten mit einem BV
 - Problem: Wie wählt man die Objekte sinnvoll aus?
- Top-down:
 - Unterteile rekursiv die Menge der Primitive
 - Problem: nach welchem Kriterium unterteilt man?
 - Median-Cut:
 - Sortieren der Objekte entlang einer Koordinatenachse, Aufteilen in zwei Hälften und Verfahren rekursiv anwenden
 - Problem: Sortierkriterium (Objekte haben Ausdehnung)
- Iterative Insert:
 - Heuristik nach Goldsmith/Salmon



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 84

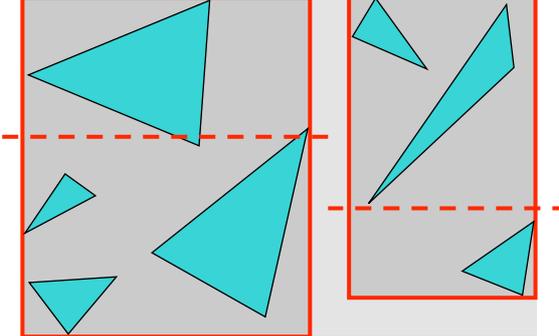
Beispiel für die Erzeugung einer BVH

- Schließe alle Objekte (= Dreiecke meistens) durch (elementare) BVs ein (z.B. AABB)
 - Arbeite ab jetzt nur noch mit diesen elementaren BVs
- Teile die Menge der Objekte in zwei Gruppen auf
- Rekursion



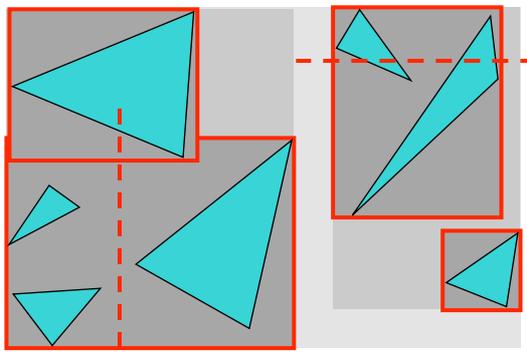
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 88

- Schließe alle Objekte (= Dreiecke meistens) durch (elementare) BVs ein (z.B. AABB)
 - Arbeite ab jetzt nur noch mit diesen elementaren BVs
- Teile die Menge der Objekte in zwei Gruppen auf
- Rekursion



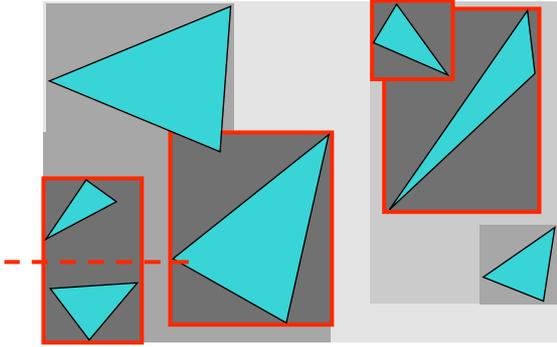
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 89

- Schließe alle Objekte (= Dreiecke meistens) durch (elementare) BVs ein (z.B. AABB)
 - Arbeite ab jetzt nur noch mit diesen elementaren BVs
- Teile die Menge der Objekte in zwei Gruppen auf
- Rekursion



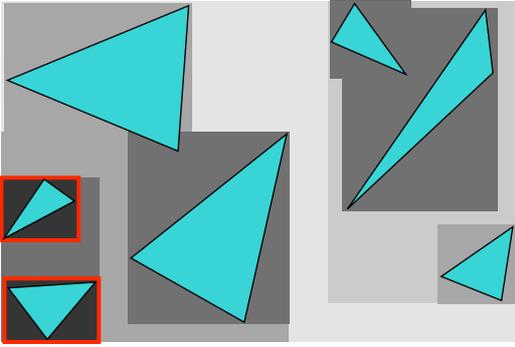
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 90

- Schließe alle Objekte (= Dreiecke meistens) durch (elementare) BVs ein (z.B. AABB)
 - Arbeite ab jetzt nur noch mit diesen elementaren BVs
- Teile die Menge der Objekte in zwei Gruppen auf
- Rekursion



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 91

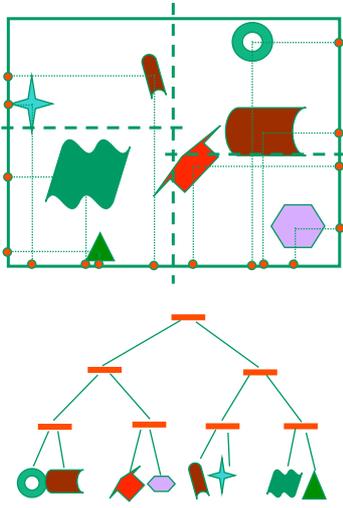
- Schließe alle Objekte (= Dreiecke meistens) durch (elementare) BVs ein (z.B. AABB)
 - Arbeite ab jetzt nur noch mit diesen elementaren BVs
- Teile die Menge der Objekte in zwei Gruppen auf
- Rekursion



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 92

Einfachste Heuristik für Top-Down: Median Cut

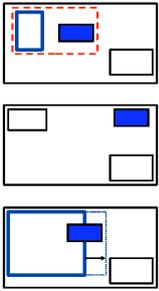
1. Bestimme BV für alle Objekte
2. Sortiere die Objekte gemäß ihrem Mittelpunkt entlang der x-Achse
3. Teile die Szene in der Mitte; die eine Hälfte der Objekte wird dem linken Teilbaum zugeordnet, die andere Hälfte dem rechten Teilbaum
4. Wiederhole 1-3 rekursiv auf die Teilszenen
 1. Variante: wähle auf jeder Ebene zyklisch eine andere Achse
 2. Variante: wähle die Achse der längsten Ausdehnung
 - § Terminierung, wenn Teilszene nur noch n Objekte enthält



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 93

Iterativer Insert [Goldsmith und Salmon, 1987]

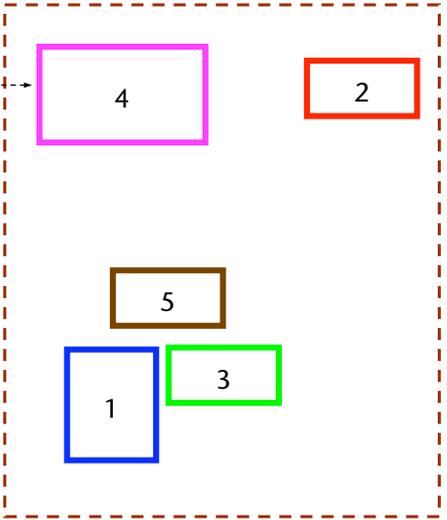
- Iterativer / rekursiver Algorithmus
- Starte mit einem einzelnen Wurzelknoten
- Füge nacheinander jeweils 1 Dreieck in die bis dahin bestehende BVH ein:
 - Lasse das Dreieck rekursiv nach unten "sickern"
 - Vergrößere dabei ggf. das BV der Knoten
 - Ist das Dreieck an einem Blatt angekommen →
 - Ersetze das Blatt durch einen inneren Knoten
 - füge das neue und das alte Dreieck als dessen Kinder an
 - Steht man an einem inneren Knoten → treffe eine der folgenden Entscheidungen:
 - füge das Dreieck am aktuellen (inneren) Knoten an
 - lasse das Dreieck in den linken / rechten Teilbaum sickern



G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 94

Beispiel für Goldsmith und Salmon

- Szene vor der Erzeugung der Hierarchie
- Jedes Objekt wird durch sein Bounding Volume umgeben
- Das gestrichelte Viereck ist die gesamte Szene



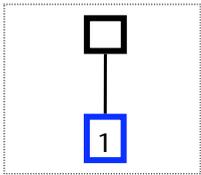
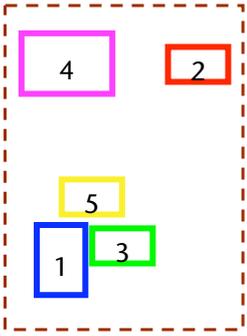
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 95

1. Iteration

Gegenwärtiger Baum



Möglichkeiten

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 96

2. Iteration

Gegenwärtiger Baum

Möglichkeiten

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 97

3. Iteration

Gegenwärtiger Baum

Möglichkeiten

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 98

4. Iteration

Gegenwärtiger Baum

Möglichkeiten

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 99

5. Iteration

Gegenwärtiger Baum

Möglichkeiten

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 100

5. Iteration

Gegenwärtiger Baum

Möglichkeiten

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 101

Bemerkungen

- Die Reihenfolge, in der die Objekte eingefügt werden, hat einen sehr großen Einfluss darauf, wie gut der Baum wird
- Goldsmith/Salmon experimentierten mit:
 - Reihenfolge wie im geladenen Modell
 - zufällig (shuffled)
 - Sortiert entlang einer Koordinatenachse

Zahl der Schnitt-Berechnungen pro Strahl bei verschiedenen Testszenen

User Supplied	5.94	19.9	12.9	10.1	32.0	63.2
Sorted	6.53	20.0	15.9	13.3	32.0	55.2
Average Shuffled	6.21	19.9	14.3	9.4	40.5	44.8
Best Shuffled	5.94	19.9	12.4	8.7	36.7	42.4
Worst Shuffled	6.32	19.9	17.4	18.3	48.2	47.2

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 102

Die entscheidende Frage

- Bei Salmon/Goldsmith (inkrementell):
Zu **welchem Teilbaum** soll ein Dreieck hinzugefügt werden?
- Bei top-down Aufbau:
Welches ist, zu einer geg. Menge von Dreiecken, die **optimale Aufteilung** in zwei Teilmengen? (wie bei kd-Tree)
- Erinnerung: die **Surface-Area-Heuristic (SAH)** —
teile B so auf, daß

$$C(B) = \text{Area}(B_1) \cdot N(B_1) + \text{Area}(B_2) \cdot N(B_2)$$
 minimal wird
- Verwende also diese zur Entscheidung

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 103

- Anwendung auf Salmon/Goldsmith:
 - Propagiere das Objekt in denjenigen Unterbaum, der dadurch die geringste Kostenerhöhung für das Ray-Tracing verursacht
 - Falls beide die gleichen Kosten verursachen (z.B. 0), verwende eine andere Heuristik, z.B. Anzahl Dreiecke im Teilbaum
 - Falls alle Unterbäume zu hohe Kosten verursachen (z.B. Flächenzunahme auf 90% der Fläche von Vater), hänge Objekt als direktes Kind an den aktuellen Knoten (BVH ist also nicht notwendig binär)

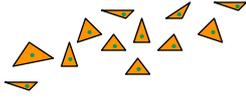
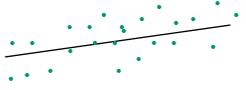
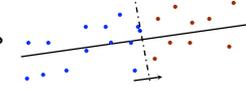
G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 104

- Anwendung auf rekursive top-down BVH-Konstruktion:
 - Berechne BV zu gegebener Menge von Objekten (= elem. BVs)
 - Partitioniere Menge der Objekte in 2 Teilmengen (oder mehr)
 - Konstruiere BVH für jede der Teilmengen
- Gesucht: optimale Aufteilung

$$C(B) = \min_{B' \in \mathcal{P}(B)} C(B', B \setminus B')$$

wobei B = Menge der Polygone im Vater-BV
- Ist natürlich nicht praktikabel

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 105

- Heuristischer Aufbau einer BVH:
 - Repräsentiere Objekte (Dreiecke) durch deren Mittelpunkte 
 - Bestimme die Achse der größten Ausdehnung 
 - Sortiere die Punkte entlang dieser Achse 
 - Suche entlang dieser Achse das Minimum gemäß Kosten-Heuristik mittels Plane-Sweep:

$$k = \arg \min_{j=1 \dots n} \left\{ \frac{\text{Area}(b_1 \dots b_j)}{\text{Area}(B)} \cdot j + \frac{\text{Area}(b_{j+1} \dots b_n)}{\text{Area}(B)} \cdot (n - j) \right\}$$

wobei die $b_j \in B$ die elementaren BVs sind und j bzw. $(n-j)$ die Anzahl der Objekte in B_1 bzw. B_2 .

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 106

■ Laufzeit:

$$T(n) = T(\alpha n) + T((1 - \alpha)n) + O(n \log n)$$

$$\in O(n \log^2 n)$$

■ Bemerkungen:

- Abbruchkriterium bei top-down Verfahren: analog zum kd-Tree
- Top-down-Verfahren liefert i.A. bessere BVHs als iteratives Verfahren

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 107

Vergleich verschiedener Datenstrukturen [Havran, 2001(?)]

Struktur	T_B	T_R	T_B + T_R
KD	~1000	~5000	5241
O93A	~1000	~6000	6820
O84A	~1000	~6000	6930
RG	~1000	~7000	8710
HUG	~1000	~13000	14760
AG	~1000	~20000	22114
UG	~1000	~30000	31900
O93	~1000	~50000	53350
BSP	~1000	~60000	66092
O89	~1000	~80000	89450
O84	~1000	~80000	89800
BVH	~1000	~400000	410000

■ Achtung: mit Vorsicht genießen!

G. Zachmann Computer-Graphik 2 - SS 08 Ray-Tracing Acceleration 108