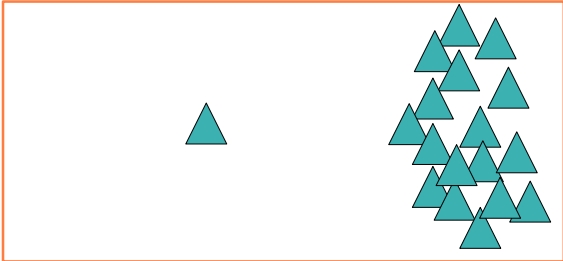


Zur Wahl der Splitting-Plane

- **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

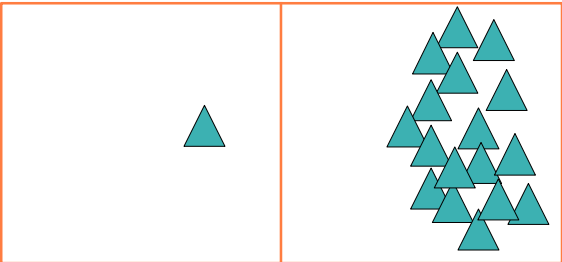
G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 45

Motivation der Kostenfunktion



G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 46

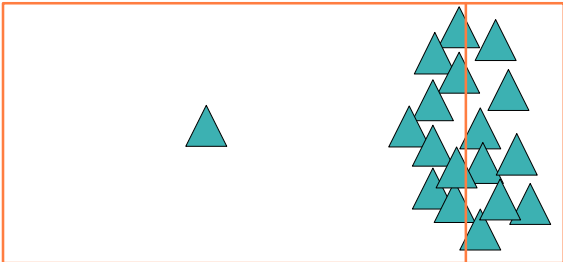
Split in der Mitte:



- 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 07 Ray-Tracing Acceleration 47

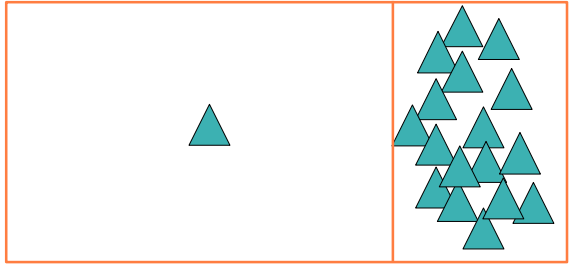
Split am Median:



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

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 48

▪ Kosten-optimierte Heuristik:



▪ Ungefähr gleiche erwartete Kosten
- Wahrscheinlichkeit für Hit links größer, dafür sind dort weniger Polygone

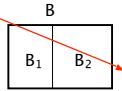
G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 49

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

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

▪ Erwartete Kosten eines Strahltests:

- bei der Transversierung ist man bei Zelle B angekommen
- Zelle B habe Kinder B_1, B_2
- Erwartete Kosten (~ 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 07 Ray-Tracing Acceleration 50

▪ Wahrscheinlichkeit:

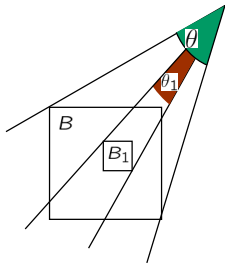
$$P[\text{Schnitt mit } B_1 \mid \text{Schnitt mit } B] = \frac{\theta_1}{\theta} \approx \frac{\text{Area}(B_1)}{\text{Area}(B)}$$

wobei $\theta \mid \theta_1$ 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 07 Ray-Tracing Acceleration 51

▪ Auflösung der "rekursiven" Formel:

- Wie berechnet man $C(B_1)$ bzw. $C(B_2)$?
- Einfache Heuristik: setze

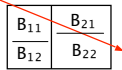
$$C(B_i) \approx \text{Anzahl Dreiecke in } B_i$$

▪ 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 07 Ray-Tracing Acceleration 52

- **Achtung:** für andere Queries (z.B. Punkte, Boxes,...) ist die Fläche **kein** Maß für die Wahrscheinlichkeit!
- Naheliegende, verbesserte(?) Heuristik: mache „Look-Ahead“

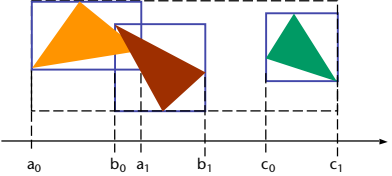
$$\begin{aligned}
 C(B) &= P[\text{Schnitt mit } B_1] \cdot C(B_1) \\
 &\quad + P[\text{Schnitt mit } B_2] \cdot C(B_2) \\
 &= P[B_1] \cdot (P[B_{11}]C(B_{11}) + P[B_{12}]C(B_{12})) \\
 &\quad + P[B_1] \cdot (P[B_{21}]C(B_{21}) + P[B_{22}]C(B_{22})) \\
 &\quad \dots
 \end{aligned}$$


Diplomarbeit ...

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 53

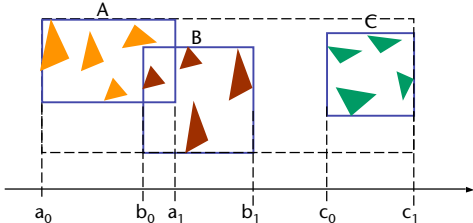
Bemerkungen

- Es genügt, die Kostenfunktion (SAH) nur an einer endlichen Folge von Stellen auszuwerten
 - Nämlich an den Rändern der Bboxes der Dreiecke
 - Dazwischen ist die 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 07 Ray-Tracing Acceleration 54

- Falls Anzahl Polygone groß (> 500,000 z.B.) → suche nur nach **ungefähr** Minimum [Havran et al., 2006]:
 - Sortiere Polygone in "Buckets"
 - Werte SAH nur an den Bucket-Grenzen aus



G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 55

Zusätzliche Kriterien [2005]

- Teste vor der SAH folgende Regel:
 - Falls eine leere Kind-Zelle abgespalten werden kann, dann erzeuge diese (überspringe SAH)
- Teste folgendes zusätzliches Abbruchkriterium:
 - Falls das Volumen der aktuellen Zelle zu klein ist, dann keine Aufteilung
 - Kriterium für "zu klein" (z.B.): $\text{Vol}(\text{Zelle}) < 0.1 \cdot \text{Vol}(\text{Root})$
 - Sinn: solche Zellen werden wahrscheinlich sowieso nicht getroffen
 - Spart Speicherplatz, ohne Laufzeit zu kosten
- Für Architekturmodelle:
 - Falls es eine Splitting-Plane gibt, die komplett von Polygonen bedeckt wird, dann verwende diese; schlage diese Polygone der kleineren Zelle zu
 - Sinn: dadurch passen sich die Zellen eher den "Räumen" an (s.a. *portal culling*)

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 56

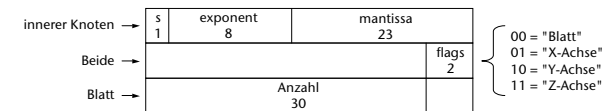
Speicherung eines kd-Tree

- 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 07

Ray-Tracing Acceleration 57

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



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

G. Zachmann Computer-Graphik 2 - SS 07

Ray-Tracing Acceleration 58

- 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` → `m_onePrim`

Falls `m_nPrims > 1` → `m_primitives`

G. Zachmann Computer-Graphik 2 - SS 07

Ray-Tracing Acceleration 59

- 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!

G. Zachmann Computer-Graphik 2 - SS 07

Ray-Tracing Acceleration 60

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 = 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 07 Ray-Tracing Acceleration 61

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

Diplomarbeit ...

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 62

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 07 Ray-Tracing Acceleration 63

- 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 07 Ray-Tracing Acceleration 64

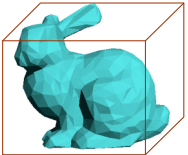
Raumunterteilung vs. Objektunterteilung

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

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 65

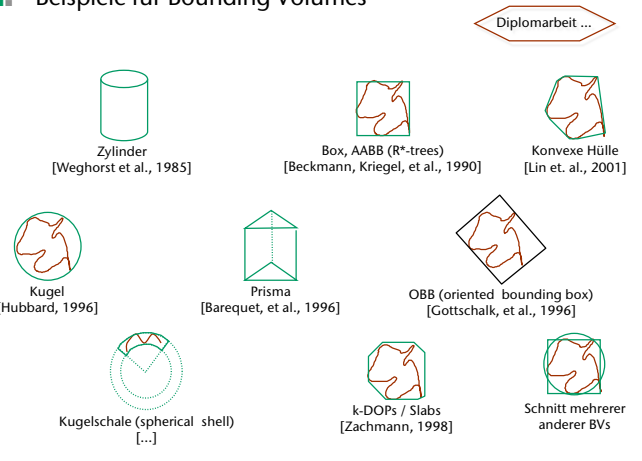
Bounding Volumes (BVs)

- 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 07 Ray-Tracing Acceleration 66

Beispiele für Bounding Volumes

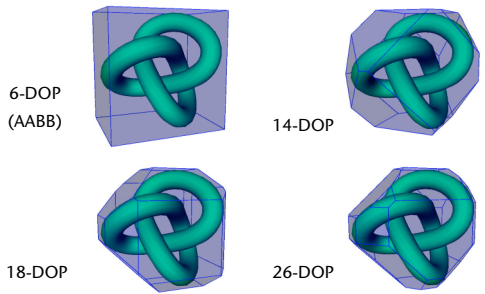


- Zylinder [Weghorst et al., 1985]
- Box, AABB (R*-trees) [Beckmann, Kriegel, et al., 1990]
- Konvexe Hülle [Lin et al., 2001]
- Kugel [Hubbard, 1996]
- Prisma [Barequet, et al., 1996]
- OBB (oriented bounding box) [Gottschalk, et al., 1996]
- Kugelschale (spherical shell) [...]
- k-DOPs / Slabs [Zachmann,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

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 67

k-DOPs

- Beispiele:



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

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 68

Kosten eines BVs

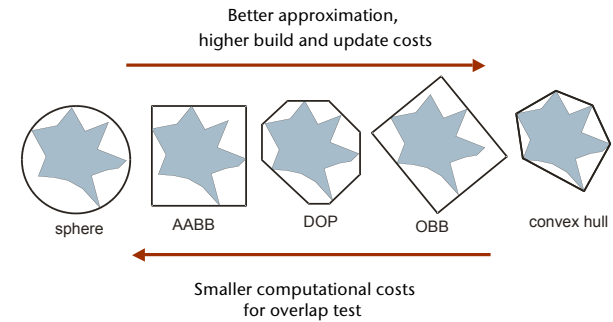
- **Kosten** einer Schnittpunktberechnung eines Strahls mit einer Teilszene:

$$T = n \cdot B + m \cdot I$$

- 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 treffen
- I = 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

- Qualitativer Vergleich:



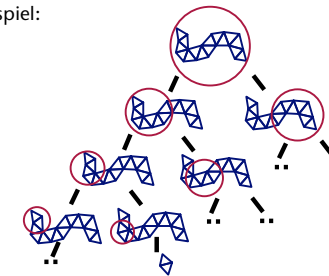
Bounding Volume Hierarchy (BVH)

- **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

$$\mathcal{P} = \mathcal{P}_1 \cup \dots \cup \mathcal{P}_n$$
 wobei \mathcal{P}_i die den Kindern zugeordneten Primitive sind

- Schematisches Beispiel:



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

Beispiele

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 73

Beispiel für Traversierung einer BVH für Strahlschnittest

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 74

- 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 07 Ray-Tracing Acceleration 75

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 07 Ray-Tracing Acceleration 76

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 07 Ray-Tracing Acceleration 77

Beispiel

- 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 07 Ray-Tracing Acceleration 78

Anmerkungen

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

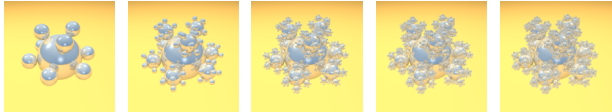
- Für die Priority Queue ist keine vollständige Ordnung notwendig, da in jedem Schleifendurchlauf nur das Element mit der kleinsten estimated distance benötigt wird
- Effiziente Umsetzung mit einem **Heap**

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 79

- **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 07 Ray-Tracing Acceleration 80

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 07 Ray-Tracing Acceleration 81