DIE VERWENDUNG VORAUSBERECHNETER DREIECKS-CLUSTER FÜR BESCHLEUNIGTES RAYTRACING IN DYNAMISCHEN SZENEN

Seminarvortrag im Seminar Computergrafik

20.12.2011

Simon Schütz

Fakultät für Physik

Universität Göttingen

simon.schuetz@stud.uni-goettingen.de

Nach: Kirill Garanzha. The use of precomputed triangle clusters for accelerated ray tracing in dynamic scenes. Computer Graphics Forum, 28:1199–1206, 2009.

ABLAUF

- Einführung
  - Raytracing in dynamischen Szenen
- Räumlich sortierende Datenstrukturen
  - Bounding Volume Hierarchy
  - Oberflächenheuristik
- Beschleunigungsalgorithmus
  - Heuristik zum Clustern von Dreiecken
  - Erzeugung von Clustern
- Ergebnisse
  - Bestimmung der idealen Clustergröße
  - Vergleich mit Referenzmodellen

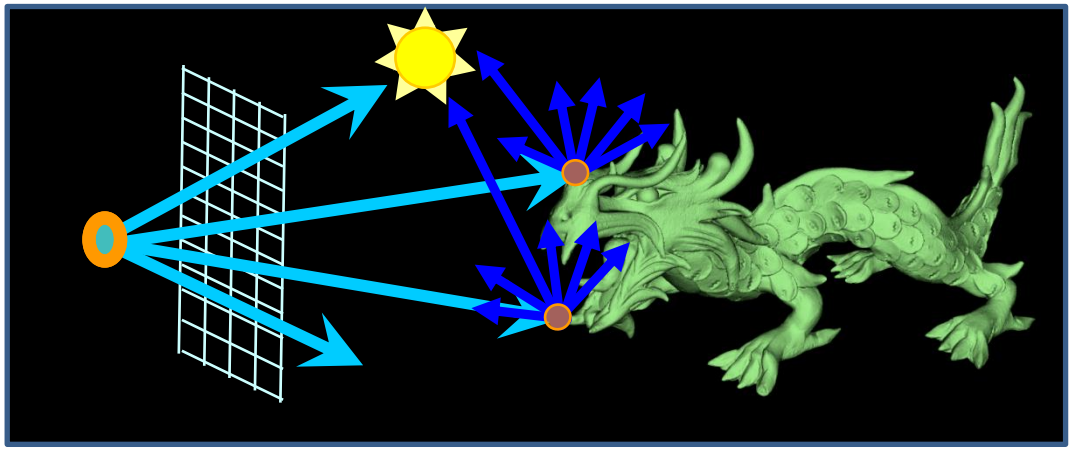
ABLAUF

- Einführung
  - Raytracing in dynamischen Szenen
- Räumlich sortierende Datenstrukturen
  - Bounding Volume Hierarchy
  - Oberflächenheuristik
- Beschleunigungsalgorithmus
  - Heuristik zum Clustern von Dreiecken
  - Erzeugung von Clustern
- Ergebnisse
  - Bestimmung der idealen Clustergröße
  - Vergleich mit Referenzmodellen

Einführung
Raytracing in dynamischen Szenen

Ablauf beim Raytracing

- Strahl wird in Szene geschickt
- Bei Kollision mit Objekt wird Farbe (Beleuchtung) berechnet
- Dazu möglicherweise Aussendung von Sekundärstrahlen (Spiegel/diffuse Streuung)



Quelle: [2]

Einführung
Raytracing in dynamischen Szenen

Beschleunigung der Kollisionserkennung

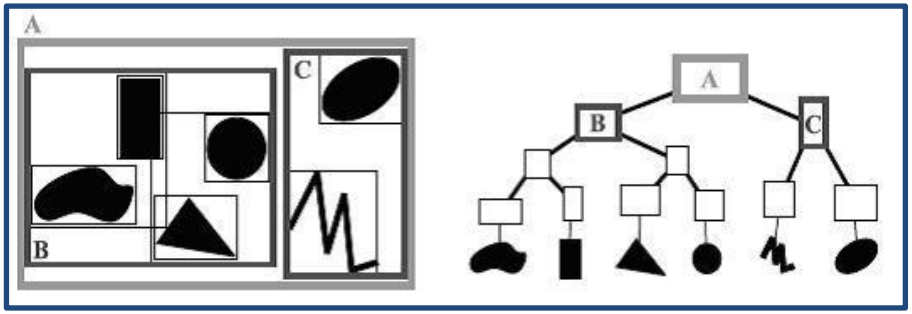
- Räumlicher Suchbaum der Objekte
- Muss in statischen Szenen nur einmal aufgebaut werden, kann dann mit unterschiedlichen Kamerapositionen/Beleuchtungen wiederverwendet werden
- Problem in animierten Szenen: Räumliche Anordnung verändert sich
  - Baum für jeden Frame neu aufbauen: teuer
  - Baum behalten und aktualisieren: kompliziert, nur bei bestimmten Animationen möglich

ABLAUF

- Einführung
  - Raytracing in dynamischen Szenen
- Räumlich sortierende Datenstrukturen
  - Bounding Volume Hierarchy
  - Oberflächenheuristik
- Beschleunigungsalgorithmus
  - Heuristik zum Clustern von Dreiecken
  - Erzeugung von Clustern
- Ergebnisse
  - Bestimmung der idealen Clustergröße
  - Vergleich mit Referenzmodellen

Räumlich sortierende Datenstrukturen
Bounding Volume Hierarchy (BVH)

- Binärer Baum
  - Hierarchie von Bounding Boxes
  - Innere Knoten sind ausgerichtete Quader
  - Daten in den Blättern gespeichert
- Aufbau
  - Mindestens O(n log n)
  - Heuristik optimiert Anordnung
- Auswertung
  - Traversierung entlang des Strahls
  - Nur geschnittene Blätter müssen auf Kollision geprüft werden



Quelle: [5]

Räumlich sortierende Datenstrukturen
Oberflächenheuristik (Surface Area Heuristic, SAH)

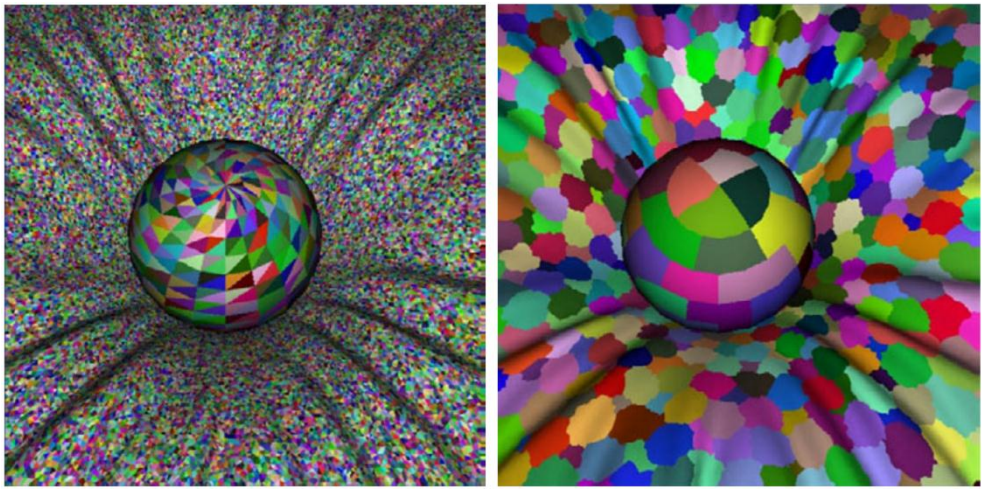
- Ziel: BVH so schachteln, dass möglichst kleine Bounding Boxes entstehen
- Annahme: Strahlen laufen in alle Richtungen
  - Dann gilt: Wahrscheinlichkeit $P_{hit}(X)$, dass Volumen X getroffen wird, wächst proportional zu seiner Oberfläche $SA(X)$
  - Bei Aufteilung von Volumen X in Teilvolumen Y_1, $Y_2 \subset X$ (wobei $Y_1 \cup Y_2$ alle Dreiecke in X enthält) ist die Wahrscheinlichkeit, dass Volumen Y_i getroffen wird, wenn der Strahl X trifft, gegeben durch: $P_{hit}(Y_i|X) = \frac{SA(Y_i)}{SA(X)}$
  - Gewichtung der Trefferwahrscheinlichkeiten nach Anzahl der in den Teilvolumen befindlichen Dreiecke ergibt die Kosten, die für eine Traversierung bei der Unterteilung von X in Y_1 und Y_2 anfallen
  - Unterteilung mit niedrigsten Kosten ist am besten

ABLAUF

- Einführung
  - Raytracing in dynamischen Szenen
- Räumlich sortierende Datenstrukturen
  - Bounding Volume Hierarchy
  - Oberflächenheuristik
- Beschleunigungsalgorithmus
  - Heuristik zum Clustern von Dreiecken
  - Erzeugung von Clustern
- Ergebnisse
  - Bestimmung der idealen Clustergröße
  - Vergleich mit Referenzmodellen

Beschleunigungsalgorithmus
Heuristik zum Clustern von Dreiecken

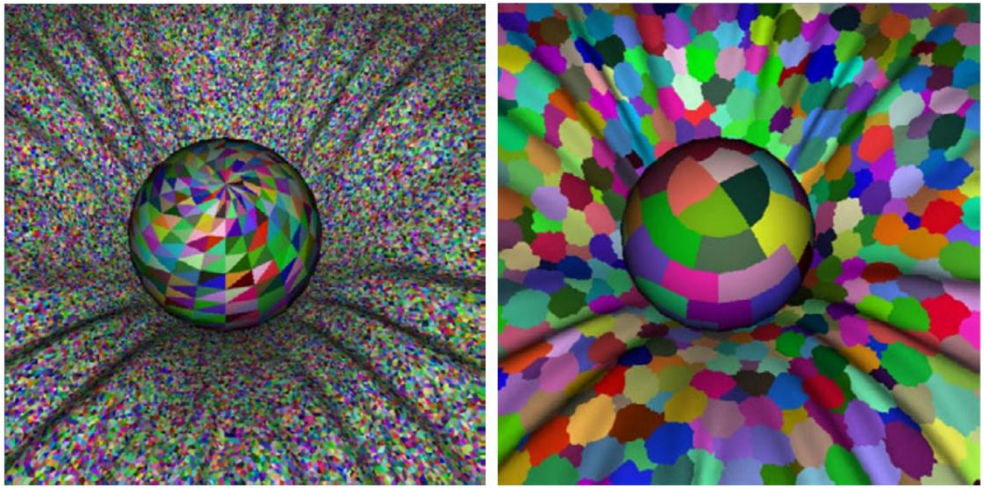
- Idee: BVH wird schneller aufgebaut, wenn weniger Objekte sortiert werden müssen
- Umsetzung: Fasse benachbarte Dreiecke in Dreiecksnetzen zu Clustern zusammen, die jeweils eine kleine Bounding Box haben
- Annahme: In Animationen bleiben Ausmaße der Cluster nahezu unverändert



Quelle: [1]

Beschleunigungsalgorithmus
Heuristik zum Clustern von Dreiecken

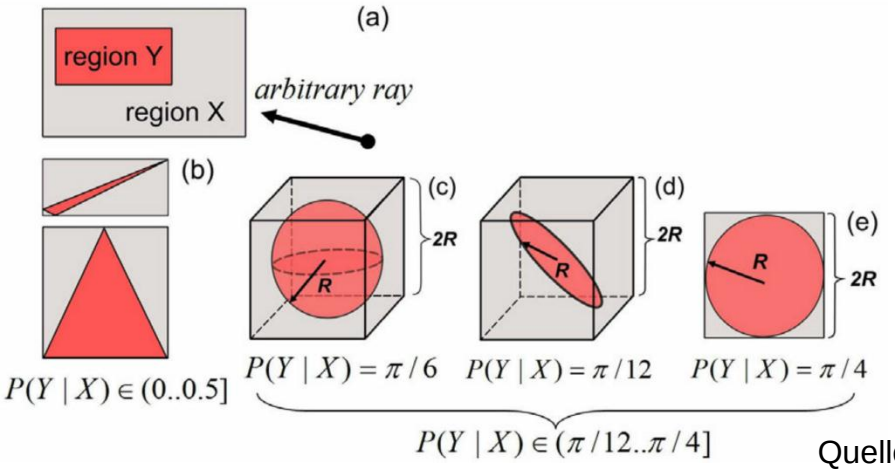
- Kriterien bei der Wahl der Cluster
  - Form soll rund oder sphärisch sein
  - Konnektivität des Dreiecksnetzes soll hoch sein
  - Gleichmäßige geometrische Abmessungen und Zahl der Vertizes



Quelle: [1]

Beschleunigungsalgorithmus
Heuristik zum Clustern von Dreiecken

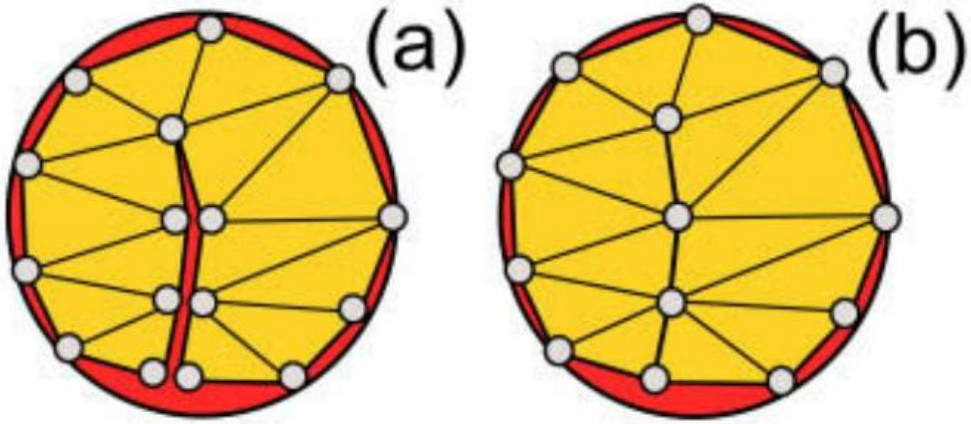
- Motivation für runde/sphärische Form: Maximiere(!) $P_{hit}(Y|X)$ für Form Y in Bounding Box X:
  - Wenn Box ausgewertet wird, soll das Ergebnis ein Treffer sein, damit die Traversierung enden kann
  - Für beliebige Dreiecke: $P(Y|X) \leq 0.5$
  - Für Kugeln und Kreise: Im Mittel höhere Trefferwahrscheinlichkeit



Quelle: [1]

Beschleunigungsalgorithmus
Heuristik zum Clustern von Dreiecken

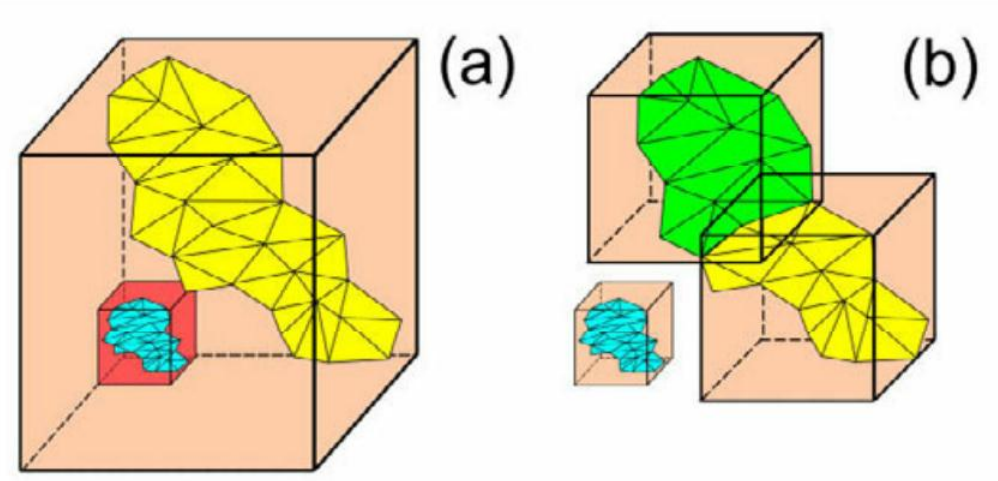
- Motivation für hohe Konnektivität des Dreiecksnetzes: Im Verlauf der Animation werden einige Kanten verzerrt
  - Bei eng verbundenen Clustern ist die Verzerrung eher gleichmäßig
  - Lose verbundene Cluster können auch „auseinanderklappen“ und wesentlich größer werden



Quelle: [1]

Beschleunigungsalgorithmus
Heuristik zum Clustern von Dreiecken

- Motivation für gleichmäßige Abmessungen: Sehr große Cluster haben auch große Bounding Boxes, die in der BVH-Struktur für Mehraufwand sorgen
  - Ein großer Cluster wird durch Bounding Box weniger genau approximiert als mehrere kleine Cluster
  - Überlappende Bounding Boxes erzeugen Mehraufwand



Quelle: [1]

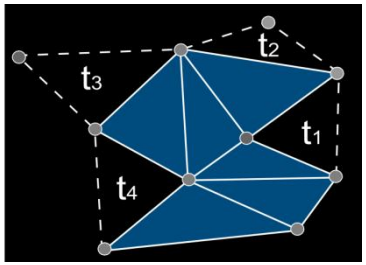
Beschleunigungsalgorithmus
Erzeugen von Clustern

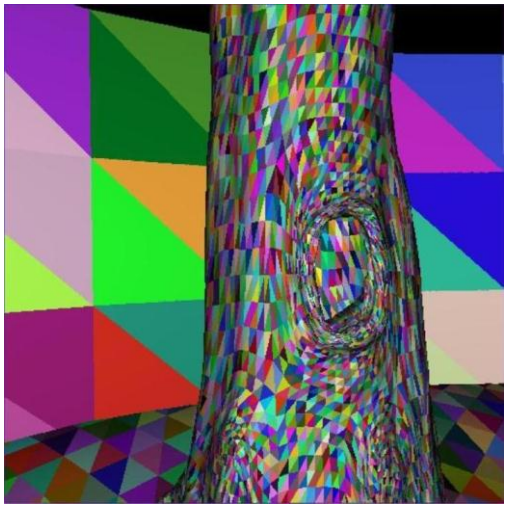
- Zwei Methoden zur Aufteilung eines Dreiecksnetzes in Cluster
  - Iteratives Wachstum
  - Iterative Kontraktion
- Basieren auf Funktion $Acc(k)$ für ein Cluster k, deren Wert umso höher ist, je besser das Cluster die Bedingungen (runde Form, hohe Konnektivität, beschränkte Ausmaße) erfüllt
- Weitere Methoden sind denkbar. Methoden sind je nach Aufbau der Szene unterschiedlich gut geeignet.

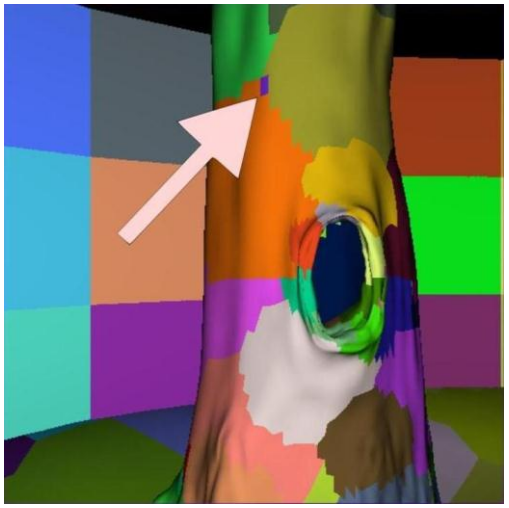
Beschleunigungsalgorithmus
Erzeugen von Clustern

Iteratives Wachstum

- Starte mit einem Dreieck d_0 als Startcluster $k = d_0$. Füge diesem Cluster solange immer denjenigen Nachbarn d_n zu, der das maximale $Acc(k + d_n)$ erzeugt, bis $Acc(k + d_n) < 0 \; \forall \; d_n$.
  - Geringe Laufzeit
  - Die ersten Cluster sind sehr gut; spätere Cluster werden zunehmend schlechter





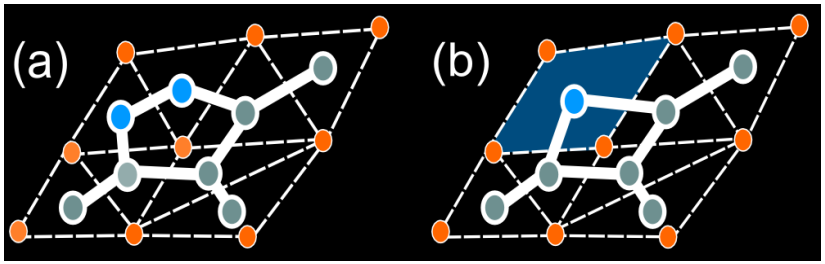


Quellen: [2]

Beschleunigungsalgorithmus
Erzeugen von Clustern

Iterative Kontraktion

- Betrachte die Dreiecke als Cluster c_i. Berechne für alle benachbarten Cluster c_i, c_j den Wert $Acc(c_i + c_j)$. Finde das (i, j) mit dem höchsten Wert und verschmelze diese zu einem gemeinsamen Cluster. Iteriere, bis $Acc(c_i + c_j) < 0 \; \forall \; (i, j)$.
  - Hoher Speicherbedarf
  - Clusterkanten sind weniger regelmäßig, aber alle Cluster sind ähnlich gut aufgebaut



Quellen: [2]

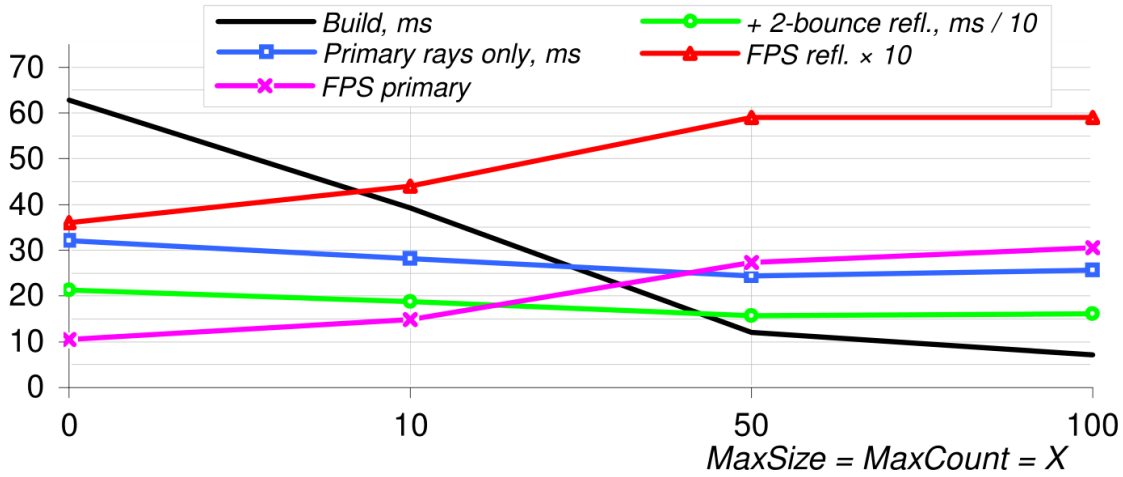
ABLAUF

- Einführung
  - Raytracing in dynamischen Szenen
- Räumlich sortierende Datenstrukturen
  - Bounding Volume Hierarchy
  - Oberflächenheuristik
- Beschleunigungsalgorithmus
  - Heuristik zum Clustern von Dreiecken
  - Erzeugung von Clustern
- Ergebnisse
  - Bestimmung der idealen Clustergröße
  - Vergleich mit Referenzmodellen

Ergebnisse
Bestimmung der idealen Clustergröße

- Kriterium „gleichmäßige Abmessungen“ benötigt gewünschte Zahl der Vertizes pro Cluster
- Tests ergeben: Bei 50 Vertizes pro Cluster arbeitet BVH-Auswertung am schnellsten

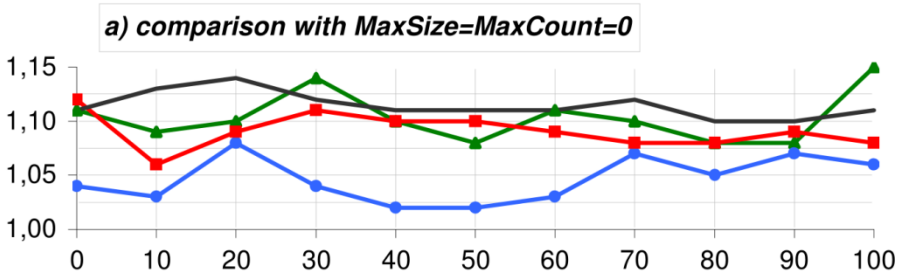
Quelle: [2]

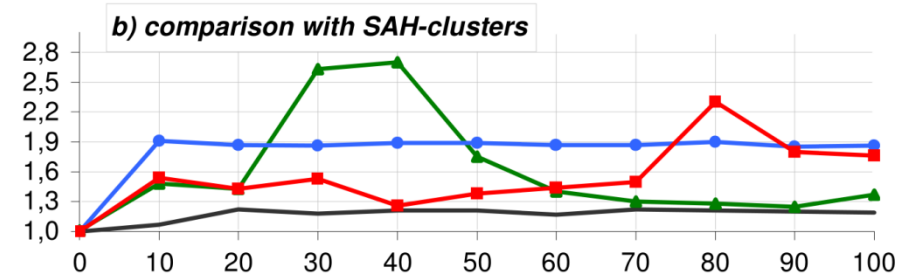


Quelle: [1]

Ergebnisse
Vergleich mit Referenzmodellen

- Vergleich der Bildrate bei Animation über 100 Frames mit 4 verschiedenen Animationen
- Standardansatz: BVH-Baum mit einem Dreieck pro Blatt
  - Neues Modell um bis zu 15% schneller
- SAH-Clustering: Cluster werden auf Basis der Oberflächenheuristik gebildet
  - Im ersten Frame vergleichbar; danach ist neues Modell teilw. doppelt so schnell





Quelle: [1]

Ergebnisse
Verbesserungsansätze

Folgende Verbesserungen sind noch angedacht oder möglich:

- Aktualisierung der Clusterstruktur durch Umgruppierung von Dreiecken während der Animation
- Behandlung von Situationen, in denen Indizierung oder Konnektivität der Dreiecksnetze sich im Lauf der Animation ändert
- Nutzung verbesserter BVH-Algorithmen oder anderer Datenstrukturen

Quellen

[1] Kirill Garanzha. The use of precomputed triangle clusters for accelerated ray tracing in dynamic scenes. Computer Graphics Forum, 28:1199–1206, 2009. doi: 10.1111/j.1467-8659.2009.01497.x.

[2] Kirill Garanzha. Vortrag: The use of precomputed triangle clusters for accelerated ray tracing in dynamic scenes. 20-th Eurographics Symposium on Rendering, 2009. URL http://garanzha.com/Documents/UPTC-ART-DS.ppt.

[3] Michael Garland, Andrew Wilmott, and Paul S. Heckbert. Hierachical face clustering on polygonal surfaces. Proc. of the Symposium of Interactive 3D Graphics, 2001.

[4] Johannes Günther, Heiko Friedrich, Ingo Wald, Hans-Peter Seidel, and Philipp Slusallek. Ray tracing animated scenes using motion decomposition. Eurographics, 25:3, 2006.

[5] Houjun8022. Example of bounding volume hierarchy, December 2009. URL http://commons.wikimedia.org/wiki/File:Example_of_bounding_volume_hierarchy.JPG.

[6] I. Wald, W. Mark, J. Günther, S. Boulos, T. Ize, W. Hunt, S. Parker, and P. Shirley. State of the art in ray tracing animated scenes. In State of the Art Reports, Eurographics, 2007.

[7] Ingo Wald. On fast construction of sah-based bounding volume hierarchies. Proc. of the IEEE/EG Symposium on Interactive Ray Tracing, 2007.

Beschleunigungsalgorithmus
Heuristik zum Clustern von Dreiecken

- Betrachte Bounding Sphere $S(k)$ mit Radius $R(k)$. Fläche des einbeschriebenen Kreises:

$$AreaC(k) = \pi \cdot R(k)^2$$

- Berechne mittleren Normalenvektor

$$\vec{N}_{avg}(k) = Normalisiert\left(\sum_{i=1}^{k} \vec{n}_i\right)$$

- Projiziere Fläche auf Kreis C

$$AreaP(k) = \sum_{i=1}^{k} Area_i \cdot \left\|\vec{N}_{avg}(k) \cdot \vec{n}_i\right\|$$

- Bilde Quotient der Flächen

$$reg(k) = \frac{AreaP(k)}{AreaC(k)}$$

- Teile aktuelle Vertexzahl durch gewünschte Vertexzahl

$$regBound(k) = \frac{VertexCount(k)}{MaxCount}$$

- Teile Oberfläche von $S(k)$ durch gewünschte mittlere Fläche

$$sizeBound(k) = \max\left(1, \frac{SA(S(k))}{MaxSize \cdot AvgSA}\right)$$

- $reg(k)$, meist in $(0,1]$, soll groß sein
- $regBound(k)$ in $(0,1]$ soll klein sein
- $sizeBound(k)$ in $(0,1]$ soll klein sein:

$$Acc(k) = reg(k) - regB(k) \cdot sizeB(k)$$