

Überblick

- Einführung
- Beschleunigungsverfahren für Raytracing
- Grids
- Hierarchische Grids
- Bewertung
- Fazit

Einführung

Gewöhnliche Beschleunigungsverfahren & Raumunterteilung

- entworfen und optimiert für effizientes Traversieren von statischen Szenen
- zufriedenstellende Leistung nur auf Basis eines Vorverarbeitungs-Schrittes, in dem die Raumunterteilung erzeugt wird
- kompliziertes Update nach jeder Änderung der Geometrie

Einführung

Neue Anforderungen an Interaktives Raytracing

- schnelle Updates
- Einfügen & Löschen von Objekten an beliebiger Stelle in konstanter Zeit

Einführung

Anpassung aktueller Techniken der Raumunterteilung

- Anpassung bestehender Lösungen an sich verändernde Geometrie
- Problem: hoher Aufwand für Einfügen & Löschen von Objekten
 - *Spatial Subdivision*, die sowohl eine effiziente Strahl-Traversierung, als auch schnelles Einfügen & Löschen bei wachsender Ausdehnung der Szene ermöglicht

Beschleunigungsverfahren für Raytracing

Aktuelle Verfahren der Raumunterteilung

- bisher wenig Augenmerk auf bewegte Objekte
- A.S.Glassner „*Spacetime ray tracing for animation*“
Ansatz für das Raytracing animierter Objekte, der sich allerdings auf Vorkenntnisse über den Animationspfad stützt und somit keinen praktischen Nutzen für interaktive Anwendungen bietet
- Hauptanforderungen nach Gaede & Günther:
 - schnelle Strahl-Traversierung
 - Anpassungsfähigkeit an ungleichmäßig verteilte Daten

Beschleunigungsverfahren für Raytracing

Kategorien der Raumunterteilung

- *Bounding volume hierarchies*
 - Baumstrukturen
 - Objekte sind in Knoten abgelegt
 - zusätzliche Infos zur Optimierung der Traversierung und Einebnung zur Repräsentation in einem Array
 - nichttrivialer Aufwand für Einfügen & Löschen
 - Ineffizienz durch nicht ausbalancierte Baumstruktur
 - Ausbalancierungsschritt nach Einfügen & Löschen

Beschleunigungsverfahren für Raytracing

Kategorien der Raumunterteilung

- *Voxel based structures*

- *Grids* oder Baumstrukturen
- Konstruktion der Raumunterteilung mit Aufwand $O(n)$
- Aufwand für das Einfügen eines Objekt abhängig von dessen Größe
- Hauptproblem:
 - Statisches *Grid* erfordert Neuberechnung, wenn sich ein Objekt über dessen Grenzen hinaus bewegt
 - Lösung im nächsten Abschnitt

Grids

Grid Raumunterteilung für statische Szenen

- schnelle Traversierung
- unkompliziertes Einfügen & Löschen
- Einfügen:
 - Mappen der achsenparallelen *Bounding Box* des Objekts auf das *Grid*
 - Eintragen des Objekts in alle geschnittenen Voxel
- Löschen:
 - analog

Grids

Bewegen eines Objekts aus der Raumunterteilung

- Replizieren des *Grids*:
Objekte, die die Grenze des *Grids* überschreiten, werden somit auf der gegenüberliegenden Seite einfach wieder eingefügt
- Differenzierung:
Objekte, die außerhalb des physikalischen *Grids* liegen, werden speziell gekennzeichnet, um zwischen den verschiedenen Ebenen eines *Voxels* unterscheiden zu können
- Aufwand für Traversierung:
erhöht sich leicht, da nun pro Strahl eventuell mehr *Voxel* durchlaufen werden müssen

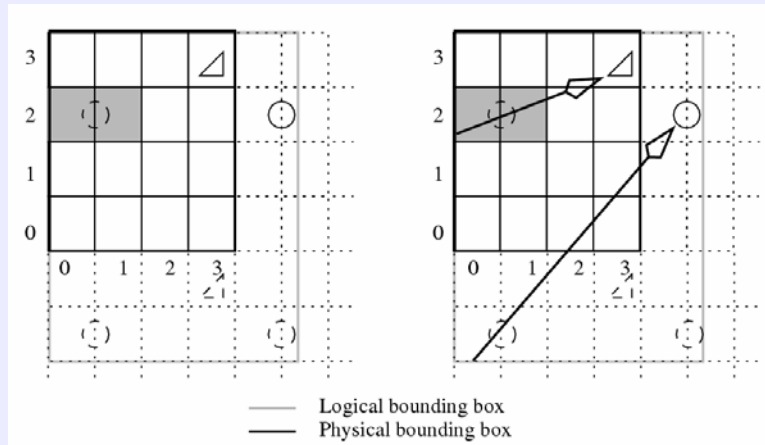
Grids

Bewegen eines Objekts aus der Raumunterteilung

- *Physical Bounding Box* (PBB):
Initiale *Bounding Box* um das gesamte *Grid*, die im Vorverarbeitungs-Schritt berechnet wird.
- *Logical Bounding Box* (LBB):
Bounding Box, die mitgeführt wird und zu jederzeit auch die Objekte einschließt, die das ursprüngliche Ausmaß der Szene überschritten haben. Sie deckt sich anfangs mit der PBB und bildet das Abbruchkriterium für die Traversierung.

Grids

Bewegen eines Objekts aus der Raumunterteilung



Fortgeschrittene Raytracing Techniken & Augmentierte Bildsynthese – Timo Wirtz – 13.06.2003

Grids

Bewegen eines Objekts aus der Raumunterteilung

- Tradeoff:

Falls die LBB viel größer wird als die PBB entsteht ein *Tradeoff* zwischen Traversierungs-Geschwindigkeit und Neuaufbau des *Grids*.

→ Neuberechnung des *Grids*, wenn sich die Länge der Diagonalen der Bounding Boxen um den Faktor 2 unterscheiden

Fortgeschrittene Raytracing Techniken & Augmentierte Bildsynthese – Timo Wirtz – 13.06.2003

Grids

Fazit

- elegante Lösung für das Problem der Neuberechnung bei wachsenden Szenen
- weiterhin bestehendes Problem:
Der Aufwand für das Einfügen & Löschen von Objekten hängt stark von der Relation zwischen Objektgröße und Szenengröße ab.
→ Lösungsansatz im nächsten Abschnitt

Hierarchische *Grids*

Idee:

Raumunterteilung deren Voxel sich der Größe der Objekte anpassen. Somit wäre das Einfügen & Löschen von Objekten unabhängig von ihrer Größe und daher in konstanter Zeit möglich.

Lösung:

Hierarchische Raumunterteilung (Octrees, Bintrees, hierarchische *Grids*)

Erweiterung:

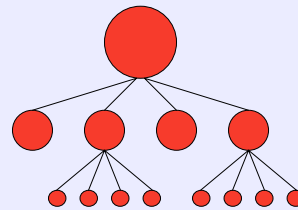
Im Vergleich zu üblichen Baumstrukturen, wird das Ablegen von Objekte nicht nur in den Blättern, sondern in allen Knoten zugelassen, um einen konstanten Aufwand beim Einfügen & Löschen zu erhalten.

Hierarchische Grids

- Ordnung:

Beim Einfügen von Objekten ist nun vorab das entsprechende *Grid*-Level zu bestimmen, welches von der Objektgröße abhängt.

$$\text{Grid-Level} = \frac{\text{Länge der Diagonalen des Grids}}{\text{Länge der Objektdiagonalen}}$$



Hierarchische Grids

- Modifikation der Traversierung

- die Traversierung startet immer an einem Blatt
- der Strahl wird mit dem entsprechenden Voxel und allen Eltern geschnitten

- Problem:

Verschwenderisches Vorgehen, da gemeinsame Elternknoten für ein und denselben Strahl wiederholt traversiert werden.

```

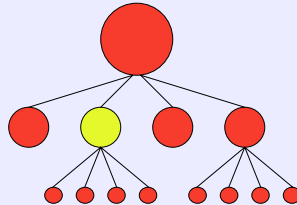
Bitmask = (raydir_x > 0) ? x : x + 1
forall levels in hierarchical grid {
    cell = hgrid[level][x>>level][y>>level][z>>level]
    forall objects in cell intersect (ray, object)
    if (bitmask & 1) return
    bitmask >>= 1
}

```

Hierarchische *Grids*

- weiteres Problem:

Um unnötiges Durchlaufen der Hierarchien zu vermeiden, kann der Aufstieg im Baum beim größten Voxel beendet werden, das noch Objekte enthält.



Hierarchische *Grids*

Fazit

- Vorteile:

- Einfügen & Löschen in konstanter Zeit
- Traversierung nur geringfügig aufwendiger als bei Standard-*Grids*
- Verfahren der Replizierung weiterhin anwendbar

- Nachteil:

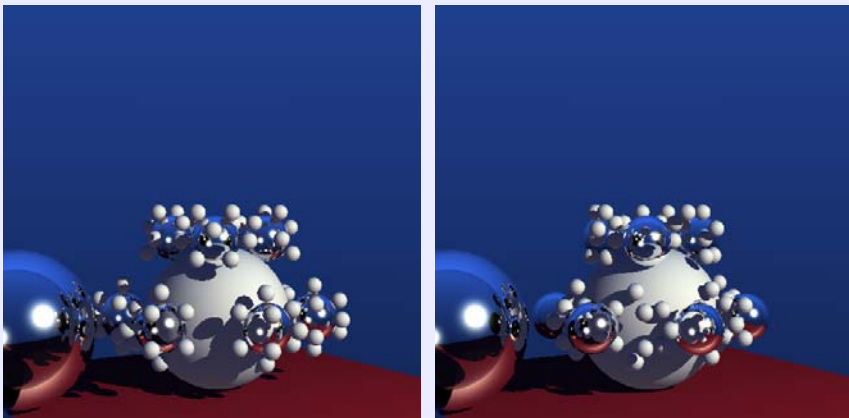
- Langsam, falls alle Objekte ähnlich groß sind

Bewertung

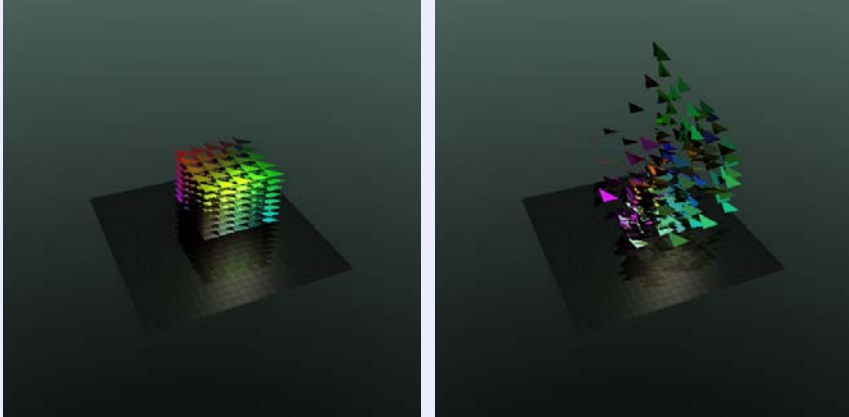
Daten

- SGI Origin 2000 mit 32 Prozessoren (genutzt 30)
- 2 Testszenen mit animierten Objekten (vordefinierte Animationspfade)
- Auflösung: 512 x 512 Pixel

Bewertung

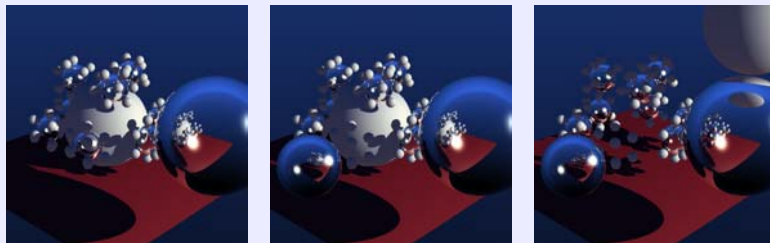


Bewertung



Fortgeschrittene Raytracing Techniken & Augmentierte Bildsynthese – Timo Wirtz – 13.06.2003

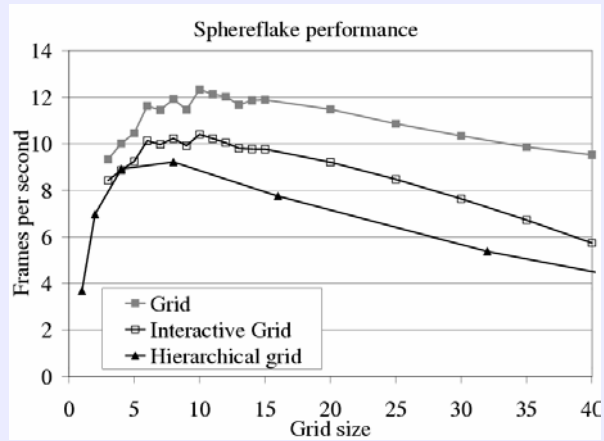
Bewertung



Bilder während der interaktiven Manipulation

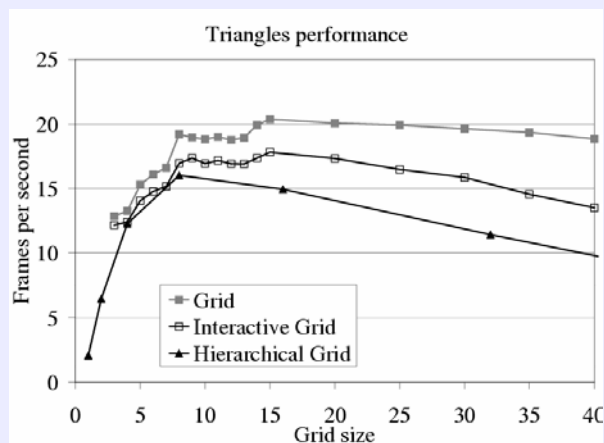
Fortgeschrittene Raytracing Techniken & Augmentierte Bildsynthese – Timo Wirtz – 13.06.2003

Bewertung



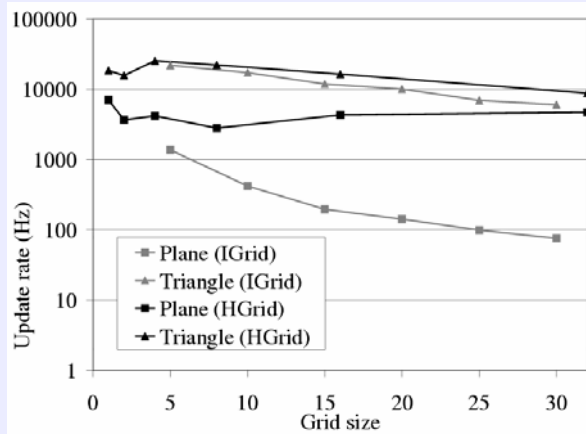
- Performanz bei statischer Szene; *Bounding Volume* Hierarchie erreicht 8.5 fps

Bewertung



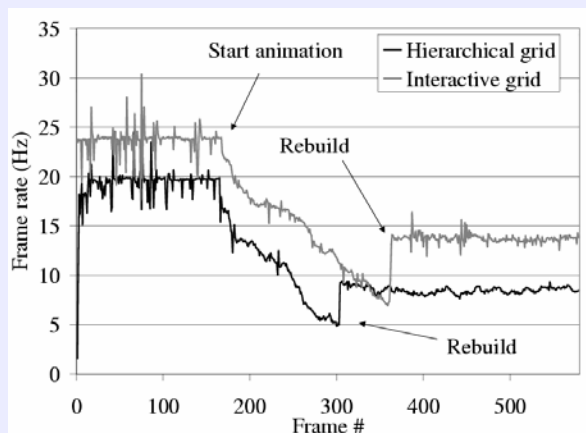
- Performanz bei statischer Szene; *Bounding Volume* Hierarchie erreicht 16.4 fps

Bewertung



- Update-Rate als Funktion über die *Grid*-Größe

Bewertung



- Frame-Rate als Funktion über die Zeit der Animation

Fazit

- Probleme mit interaktiven Anwendungen:

Arbeit, die bei statischen Anwendungen im Vorverarbeitungs-Schritt gemacht werden kann, beschränkt die erreichbare Leistung.

Im Besonderen ist dies die Raumunterteilung, die üblicherweise einmalig vor dem Start der Traversierung berechnet wird.

Der Aufwand für das Einfügen & Löschen von Objekten kann unvorhersehbar variable sein.

Bei großer Ausweitung der Szene wird Neuaufbau der Raumunterteilung nötig. Da diese Berechnung einige Zeit in Anspruch nehmen kann, sollte sie parallel in einen separaten Buffer geschehen.

Ende