

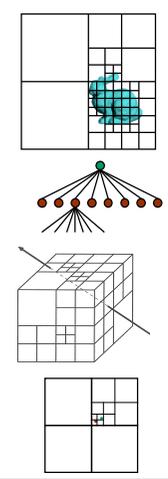
### Allgemeine Regeln zur Optimierung

- "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 07 Ray-Tracing Acceleration 28

### Octree / Quadtree

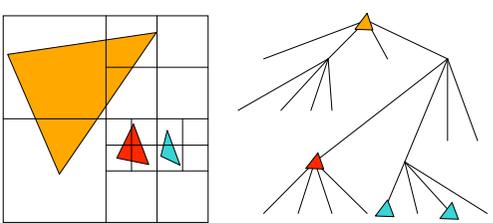
- Idee: extreme Variante der rekursiven Gitter
- Aufbau:
  - Mit BBox der gesamten Szene beginnen
  - Voxel in 8 gleiche Sub-Voxels rekursiv unterteilen
  - Abbruchkriterien: Zahl der restlichen Primitive und maximalen Tiefe
- Vorteil: lässt große Traversal-Schritte in den leeren Regionen zu („empty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



G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 29

### Primitive in adaptiven Gittern / Octrees

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



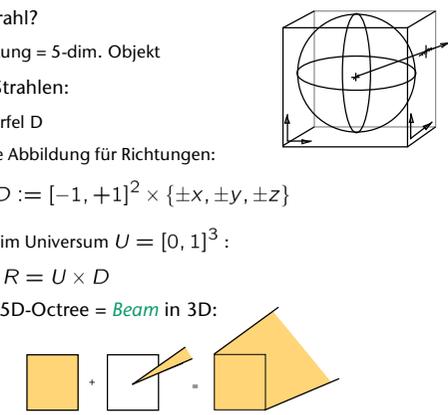
Octree/(Quadtree)

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 30

### 5D-Octree für Strahlen

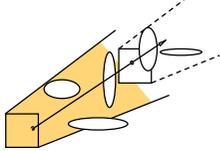
[Arvo u. Kirk 1987]

- Was ist ein Strahl?
  - Punkt + Richtung = 5-dim. Objekt
- Octree über 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 07 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 07 Ray-Tracing Acceleration 32

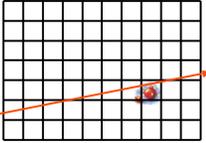
### Bemerkungen

- Die Methode führt im Prinzip eine ungefähre 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 07 Ray-Tracing Acceleration 33

### kD-Trees

- Problem der Gitter: "teapot in a stadium"
- Problem der Octrees:
  - zu starr bei der Plazierung der Unterteilung (immer Mittelpunkt)
  - Unterteilung in allen Richtungen nicht immer nötig
- Lösung: hierarchische Raumunterteilung, die die lokale "Auflösung" 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 07 Ray-Tracing Acceleration 34

- **Informelle Definition:**
  - Binärer Baum:
    - Blätter: enthalten einzelne Objekte oder Objektliste
    - Innere Knoten: *Splitting Plane* (senkrecht zu einer Achse) und Kindzeiger
  - 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 07 Ray-Tracing Acceleration 35

### Beispiel

[Slide courtesy Martin Eisemann]

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 36

### 3D-Beispiel

G. Zachmann Computer-Graphik 2 - SS 07 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  $\rightarrow t_{split}$
  - Fallunterscheidung:
    - Erst "near", dann "far" Teilbaum traversieren
    - Nur "near" traversieren
    - Nur "far" traversieren

G. Zachmann Computer-Graphik 2 - SS 07 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: // (a)
    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 07 Ray-Tracing Acceleration 39

### Optimierte Traversierung [1999]

- Beobachtung:
  - 90% aller Strahlen sind Schattenstrahlen
  - Irgendein Hit genügt (nicht notw. der nächste)
- Konsequenz:
  - Egal, in welcher Reihenfolge Kinder besucht → reines DFS machen
- Idee: Rekursion durch Iteration ersetzen
- Dazu Baum transformieren:

G. Zachmann Computer-Graphik 2 - SS 07 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 07 Ray-Tracing Acceleration 41

- Anwendungsfelder:
  - Ray Tracing
  - Photon Mapping
  - Mehrdimensionale Suchabfragen (z.B. k-nächste Nachbarn)
  - Kollisionserkennung

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 42

### Aufbau eines kD-Trees

- Gegeben:
  - Achsenparallele BBox der Szene ("Zelle")
  - Liste der Geometrieprimitive in dieser Zelle
- Ablauf:
  - Wähle eine achsenparallele Fläche, um die Zelle in zwei aufzuspalten
  - Verteile die Geometrie auf die beiden Kinder
    - evtl. einige Polygone (konzeptionell) aufspalten
  - 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 07 Ray-Tracing Acceleration 43

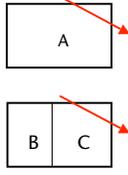
Ein Abbruchkriterium

- 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
- $t_j : t_t = 80 : 1$  (festgestellt durch Experimente)
- $p_B$  werden wir später ermitteln



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

G. Zachmann Computer-Graphik 2 - SS 07 Ray-Tracing Acceleration 44