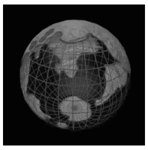


Partitionierung von Szenen

Teile und herrsche - wie Informatiker
die Welt unter sich aufteilen

Inhalt

- Räumliche Aufteilung von Szenen
  - LODs
  - Bounding Boxes
  - Binary Space Partitioning Trees
  - Cells und Portals
- Räumliche Aufteilung für Mehrbenutzerwelten
  - Filtering
  - Locals und Beacons
  - Aura

Räumliche Aufteilung

- Normalfall: VRML-Browser muß ganze Szene laden und rendern
- Spational Partitioning (räumliche Aufteilung) = teile 3D-Raum in Teilräume und berechne oder definiere Sichtbarkeitsrelation (Raum-zu-Raum)

➔ nur sichtbare Teilräume müssen geladen und gerendert werden

LOD

- In Abhängigkeit von der Entfernung des Betrachters vom Objekt, wird der Szenengraph für das Objekt ausgewählt. Je näher, um so genauer.
  - Zur Auswahl der Detailgenauigkeit, bzw. Sichtbarkeit innerhalb von Gebäuden unbrauchbar.

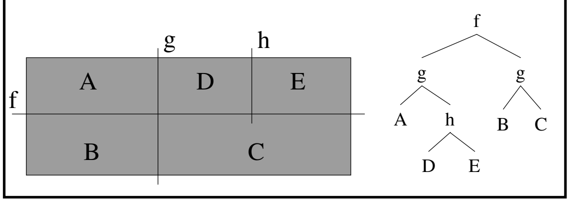
```
LOD { range [ 10 100 1000 ]
      level [ USE O1  USE O2
              USE O3  USE O4 ]}
```

Bounding Boxes

- Teilraum ist in einem SWITCH-Knoten und wird durch einen ProximitySensor an- bzw. ausgeschaltet.
  - Die Bounding Box des ProximitySensors und des Teilraums stimmen überein, d.h. erst wenn, der Betrachter sich in diesen Teilraum hineinbewegt, wird dieser gerendert.
  - Eine Staffelung (ähnlich LOD) von ProximitySensoren und Bounding Boxes ist denkbar.
- BBs auch für Culling verwendbar!
  - Konservativ: Abschätzung der sichbaren Polygone

BSP Trees

- BSP = Binary Space Partitioning
  - Teilt Raum hierarchisch durch Ebenen in Teilräume auf
- Hier: 2D Beispiel (Geraden statt Ebenen)



BSP Trees

- Zwischenknoten: Ebenen, gegeben durch Parameter `a,b,c,d` für Ebenengleichung
  - `a*x+b*y+c*z+d=0`
  - linker Teilbaum für alle Punkte `(x,y,z)`, für die `a*x+b*y+c*z+d<0`
  - linker Teilbaum für alle Punkte `(x,y,z)`, für die `a*x+b*y+c*z+d>0`
- Blattknoten: Teilräume
- Suche in BSP Tree:
  - geg.: Punkt (x,y,z), ges.: Teilraum, in dem dieser Punkt liegt

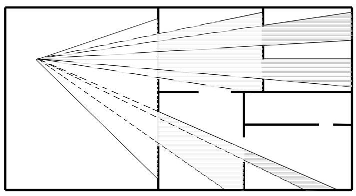
BSP Trees und Sichtbarkeit

- Die folgende Technik wurde so z.B. im Computerspiel Quake verwendet.
- Blattknoten: Teilraum + Liste aller Teilräume, die von diesem aus sichtbar sind. (statisch, von Hand eingetragen!)
- Technik: Berechne mit Hilfe des BSP Trees, in welchem Teilraum sich der Betrachter befindet, rendere dann diesen Teilraum und alle von diesem aus sichtbaren Teilräume.
- Implementierung in VRML: Betrachterposition durch ProximitySensor ermittelt an Script routen, in dem BSP Tree durchsucht wird, dann zu rendernde Teilräume über SWITCH-Knoten anschalten.

Cells and Portals

- Konservative Berechnung der Sichtbarkeit
- Anwendung: Gebäude, Fahrzeuge, Höhlen, etc.:
  - bestehen aus Teilräumen (Cells=konvexe Polyhedrons) und sind durch Fenster, Türen oder sonstige Öffnungen (Portals=Polygone) miteinander verbunden.
  - Eine Zelle ist von einer anderen nur durch eine Folge von Portals sichtbar.
- Zwei Lösungen:
  - Statische Raumaufteilung: langsame, exakte Vorberechnung (Jones, Teller)
  - Dynamische Raumaufteilung: schnelle, approximative Sichtbarkeitsberechnung (Luebke&Georges)

Cells and Portals: Ein Beispiel



Cells and Portals: Basisalgorithmus

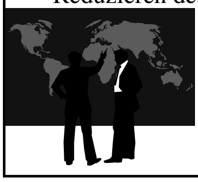
Portal: `(p,c,d)` mit `p` Polygon des Portals, verbindet Zelle `c` mit Zelle `d`.
`VisCells(c,V)` berechnet alle von `c` im Blickfeld `V` sichtbaren Zellen.
`Restrict(V,p)` berechnet Einschränkung des Blickfelds `V` durch das Polygon `p`
`VisPortals(c,V)` berechnet alle Portals, die in Zelle `c` im Blickfeld `V` liegen.

Sei `c` Zelle, in der der Betrachter ist und `V` sein Blickfeld.
Berechne `VisibleCells(c,V)`.

$$\mathrm{VisCells}(c,V) = \{c\} \cup \bigcup_{(p,c,d)\in \mathrm{VisPortals}(c,V)} \mathrm{VisCells}(d, \mathrm{Restrict}(V,p))$$

Räumliche Aufteilung für Mehrbenutzerwelten

- Skalierbarkeit
  - Reduzieren der erforderlichen Bandbreite
  - Verteilen der Teilräume auf unterschiedliche Server
- Reduzieren des Renderingaufwands



Einsicht: Ein Benutzer ist in der Regel nur an dem interessiert, was in seiner nächsten Umgebung passiert.

Filtern (z.B. in DIS)

- Ein Rechner dient als Filter und leitet nur bestimmte Nachrichten an seine Clienten weiter.
- Parameter
  - Schärfe (acuity) = Größe/Entfernung vom Betrachter
    - Nur Objekte mit für die dieses Verhältnis größer ist, sind sichtbar und ihr Szenengraph muß an den Client geschickt werden.
  - Max. Anzahl der Objekte, für die der Client Änderungen empfangen kann/will.
  - Teilraum, in dem der Client ist. Nur Änderungen in Teilräumen, die von diesem aus sichtbar sind, müssen an den Client geschickt werden.

Locales and Beacons

- Verwendet in SPLINE (Scalable Platform for Large Interactive Networked Environments) von MERL (Mitsubishi Electric Research Laboratory) --> Diamond Park
- Locale (Schauplatz)
  - eigene Menge von Multicast-Adressen
  - eigenes Koordinatensystem
  - Benutzer sieht das Locale, in dem er sich befindet und die benachbarten Locales. (können überlappen!)
- Beacon (Leuchtfeuer)
  - Namensserver liefert Locale, in dem das Objekt mit diesem Namen (tag) sich gerade befindet.

Locale

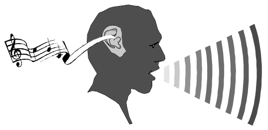
- Instanzvariablen einer Locale L
  - eindeutige ID für L
  - Vaterobjekt
    - Verschachtelung von Locales möglich
    - z.B. Auto hat eigenes Locale und bewegt sich in einem anderen
  - 3D Volumen von L
  - Multicast-Adressen von L
  - Für jedes benachbarte Locale N:
    - ID von N
    - Transformationsmatrix: N nach L
    - Transformationsmatrix: L nach N

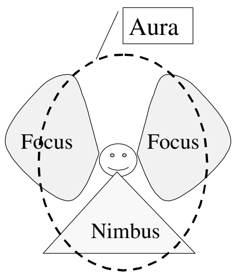
Aura

- Verwendet in den Systemen MASSIVE und DIVE
- Jedes Objekt hat pro Ereignisklasse (Medium) folgende Eigenschaften:
  - **Aura** = 3D-Volumen, in dem Objekt mit anderen Objekten interagieren kann, d.h. wenn sich die Auras zweier Objekte schneiden, können die beiden Objekte miteinander interagieren.
  - **Focus** = 3D-Volumen, in dem das Objekt Ereignisse (akustisch, visuell, …) wahrnimmt.
  - **Nimbus** = 3D-Volumen, in dem von dem Objekt verursachte Ereignisse wahrnehmbar sind.

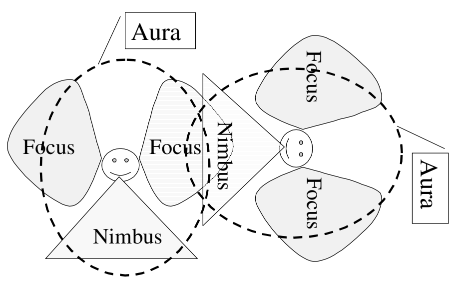
Aura

- Beispiel für Audio





Aura: Gegenseitige Wahrnehmung



Aura: Gegenseitige Wahrnehmung

Objekt A		Objekt B	
Focus	Nimbus	Focus	Nimbus
-	-	-	-
B	-	-	-
-	B	-	-
B	B	-	-
-	-	A	-
B	-	A	-
-	B	A	-
B	B	A	-
-	-	-	A
B	-	-	A
-	B	-	A
B	B	-	A
-	-	A	A
B	-	A	A
-	B	A	A
B	B	A	A

10 nicht symmetrische Fälle

Aura: Gegenseitige Wahrnehmung

- Für jeden Fall (Wahrnehmungsgrad) kann man unterschiedliche Darstellung des Objekts oder gar verschiedene Räume verwenden.
- Funktionsweise:
  – Sobald ein Objekt A mit dem Focus eines Objekts B kollidiert, erhöht sich die A's Wahrnehmung von B.
  – Kollidiert A mit dem Nimbus von B, so erhöht dies B's Wahrnehmung von A.
- Beispiel: A nimmt B wahr --> A lächelt

Aura-Manager

- AMs zuständig für Bereiche (Quadranten) der verteilten Szene
- Informiert Objekt A, wenn ein anderes Objekt B mit seiner Aura kollidiert. Objekt A kann nun B kontaktieren und z.B. dessen Szenengraph laden.
- hierarchisch: Master AM übergibt Objekt an anderen AM, sobald es einen Quadranten verläßt und in einen anderen eintritt.