GENVIS — Eine Bibliothek für Räumliche Strukturierungen auf OpenSG

Christoph Fünfzig

TU Braunschweig, Institut für ComputerGraphik
Mühlenpfordtstr. 23
38106 Braunschweig
Email: c.fuenfzig@cg.cs.tu-bs.de

ABSTRACT

Ein Szenengraph für denselben Szenengraphinhalt kann auf mehrere Arten strukturiert sein, abhängig von der Anwendung. In diesem Artikel wird die Bibliothek GENVIS für räumliche Strukturierungen auf dem Szenengraph OpenSG mit einigen Anwendungen vorgestellt. Bei der Organisation von GENVIS spielt die flexible und effiziente Anbindung an OpenSG eine wichtige Rolle. Beispielhaft werden die Strukturierung mit k-DOP Hüllkörpern und die reguläre Raumunterteilung vorgestellt. Insbesondere erstere kann zur Kollisionserkennung und zur Umstrukturierung des Szenengraph verwendet werden.

Schlüsselworte: *Szenengraph, OpenSG, Räumliche Strukturierung, Hüllkörper Hierarchie, Reguläres Gitter, Culling, Collision Detection*

Einleitung

Der Szenengraph hat sich für die schnelle Anzeige von allgemeinen Szeneninhalten durchgesetzt. Der Szeneninhalt liegt dabei als gerichteter azyklischer Graph (*DAG*)[1] vor, wobei die inneren Knoten Anzeigeattribute (Transformationen, Materialien, Umgebungseinflüsse wie Licht, Nebel usw.) verändern und die Blattknoten erst geometrische Primitive enthalten [Blythe *et al.*, 2000]. Der Vorteil der *DAG*-Struktur liegt darin, daß auf einfache Weise das Anzeigeverhalten definiert werden kann. Im allgemeinen vererben innere Knoten ihre Attribute auf den Teilbaum unterhalb.

Dieses einfache Vererbungsverhalten bringt es jedoch mit sich, daß nicht alle Sortierungen effizient realisierbar sind. Für statische Szeneninhalte mit vielen verschiedenen Materialien kann es zum Beispiel sinnvoll sein nach Materialien zu sortieren und die geometrische Primitive mit gleichem Material zusammenzufassen. Für dynamische Szeneninhalte ist dagegen eine logische Struktur mit Transformationen über den sich bewegenden Szenenteilen notwendig. Für ein effizientes Culling von statischen Szenen muß eine hinreichend feine räumliche Strukturierung im Szenengraph vorliegen.

In den meisten Fällen wird der Szeneninhalt aus einer Szenendatei eingelesen, in der überhaupt keine oder keine geeignete Strukturierung vorliegt. Ein Beispiel zeigt Abbildung 1.

In diesem Artikel wird die Bibliothek GENVIS vorgestellt, die für solche Fälle räumlich/logische Strukturierungen auf dem Szenengraph OpenSG [Reiners *et al.*, 2000] zur Verfügung stellt. Dabei ist die Verbindung der beiden Strukturen effizient und flexibel über einen Cache realisiert. Die Darstellung in diesem Artikel beschränkt sich auf die Strukturierung mit k-DOP Hüllkörpern und die reguläre Raumunterteilung.

Anbindung an den Szenengraph OpenSG

Wie in der Einleitung dargestellt, wird die Szenengraphstruktur von mehreren Faktoren beeinflußt. Um vielfältigste geometrische Anfragen effizient behandeln zu können, werden geeignete Datenstrukturen

[1] Im Szenengraph OpenSG ist ein Knoten aufgeteilt in zwei Teile *Node* und *NodeCore*. Die *Node*-Objekte bilden einen Baum von beliebigem Grad. Jedes *Node*-Objekt verweist auf ein *NodeCore*-Objekt, das den Knotentyp festlegt. Die *Node*-Objekte stehen also in einer n-zu-1 Beziehung zu den *NodeCore*-Objekten. Von den *NodeCore*-Objekten aus betrachtet, liegt also eine *DAG*-Struktur vor.

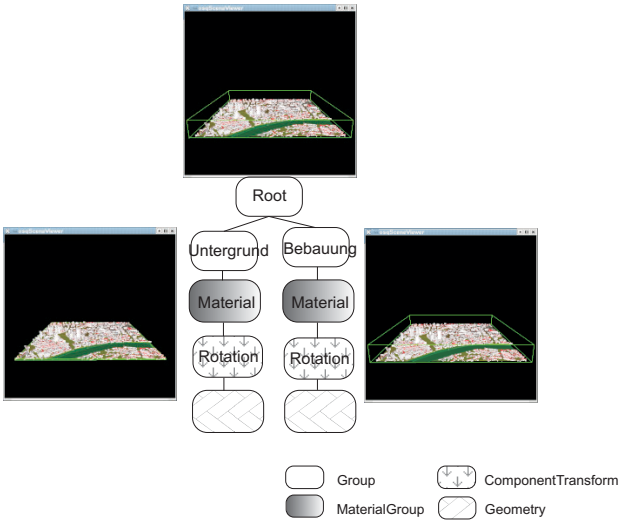

Abbildung 1: Beispiel für einen grob, logisch strukturierten Szenengraph, der sich aber aufgrund fehlender räumlicher Strukturierung schlecht zum Culling eignet.

in einer separaten Bibliothek GENVIS implementiert (Abbildung 2).

Die Datenstrukturen müssen einmal aus dem Szenengraph OpenSG aufgebaut und anschließend zu diesem konsistent gehalten werden (Weg OpenSG zu GENVIS). Als Brücke dazwischen fungiert der *OpenSGCache*, der einmal aus dem Szenengraph abgeleitet wird (*GenvisPreprocAction*). GENVIS benutzt anschließend den *OpenSGCache*.

Für Anfragestellung und Anfragereaktion muß es die Möglichkeit geben, das Szenengraph-Objekt zum Anfrageergebnis zu finden (Weg GENVIS zu OpenSG).

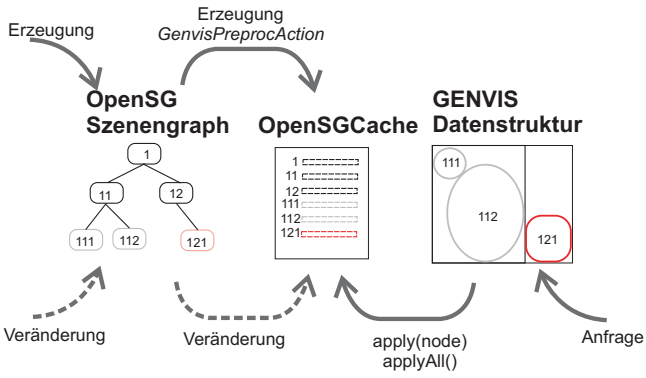

Abbildung 2: Beziehungen von OpenSG und GENVIS über den OpenSGCache.

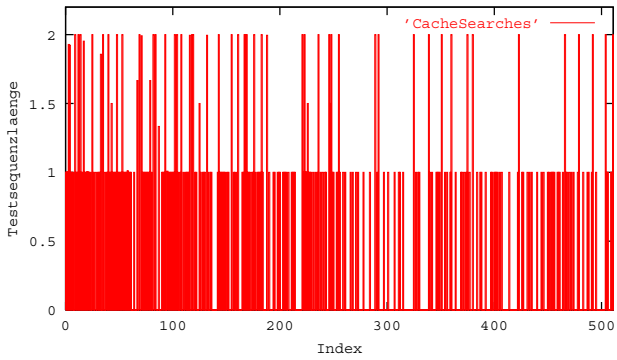

Abbildung 4: Histogramm der Testsequenzlängen für einen Cache mit 277 Datensätzen (insgesamt Platz für 512 Datensätze). Die Statistik wurde über 250 Bilder des Frankfurt-Modells gewonnen.

OpenSG zu GENVIS

Der für geometrische Anfragen relevante Inhalt des Szenengraphen (im wesentlichen Transform und abgeleitete NodeCore, Geometry) wird zur Konstruktion von Adapterobjekten benutzt. Diese werden im *OpenSGCache* gesammelt.

Der Cache erlaubt es zu einem Szenengraphknoten (identifiziert durch einen *NodePtr*) in Zeit $O(1)$ den zugehörigen Datensatz zu finden. Dazu wird eine Hashtabelle mit doppeltem Hashing $(h_1(\text{node}) = int(\text{node}), h_2(\text{node}) = 2h_1(\text{node}) + 1)$ eingesetzt. Wie in Abbildung 4 zu sehen, werden für den Zugriff auf einen Datensatz höchstens 2 Tabellenzugriffe benötigt.

Der Cache-Datensatz zum Knoten **node* umfaßt die Welt-Matrix, die Menge der Kindknoten und eine Liste der Adapterobjekte

```
class OpenSGCacheData<CONTAINER>
  OSG::NodePtr node;
  CONTAINER children;
        // = node->getMFChildren()
  OSG::Matrix matrix;
        // = node->getToWorld()
  stl::vector<Adapter*> adapter;
```

Das Argument *CONTAINER* bezeichnet dabei eine Containerklasse, die zur Speicherung der Menge der Kindknoten verwendet wird. Im einfachsten Fall kann hier ein Vektor verwendet werden. Für die Sortierung zum Beispiel nach dem Kameraabstand bietet sich eine Prioritätsliste (*map*) an [Staneker, 2001]. Die erforderlichen Aktualisierungen der Container werden mit der Methode *void sort (const SORTFUNCTOR& s)* im

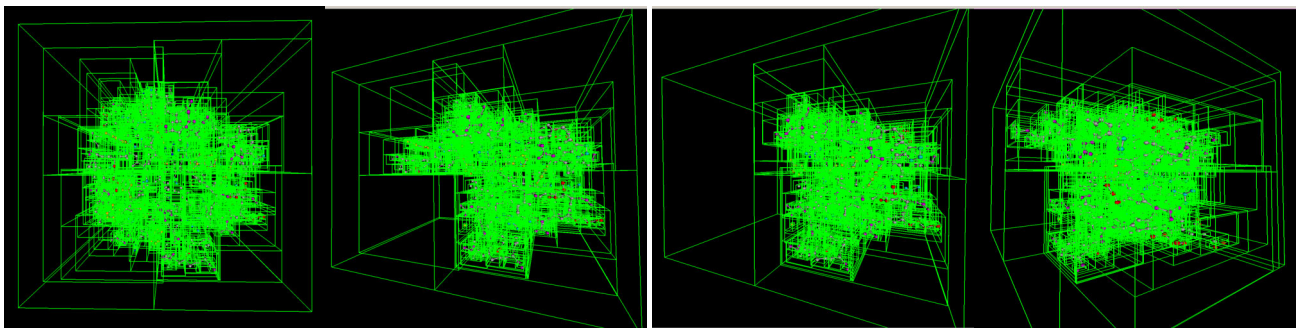

Abbildung 3: Abarbeitung der Kindknoten bis zum Abstand 45.

OpenSGCache durchgeführt, wobei *s* einen zur Containerklasse passenden Sortfunktor bezeichnet. Wegen der Lokalität des OpenSGCaches kann der Sortfunktor sehr effizient auf alle Container in den Datensätzen angewendet werden. Dies gilt insbesondere bei Verwendung von Vektor als Containerklasse.

In einer Draw-Aktion *DepthSortedDrawAction*, die auf den Cache zugreift, ist dann ein nach dem Kameraabstand sortiertes Abarbeiten der Kindknoten möglich. Für jeden Frame wird zunächst die Aktualisierung des gesamten Caches mit *sort* durchgeführt. In der Rekursion über alle Szenengraphknoten können dann die Kindknoten jeweils sortiert weiterverarbeitet (also dem Viewfrustum-Test, Occlusion-Test unterzogen) werden. Abbildung 3 zeigt eine Bildsequenz, bei der die Abarbeitung der Container bei einem festgelegten Maximalabstand abgebrochen wird.

GENVIS zu OpenSG

Generell gibt es in GENVIS eine Adapterklasse für einen Geometry-Knoten und für eine einzelne Face (Dreieck, Viereck) innerhalb eines Geometry-Knotens

```
class GeometryAdapter<BasicTraits,..>
  typedef ... TransformType;
  typedef ... GeomObjType;

  GeometryAdapter (const GeomObjectType&);
  GeometryAdapter (const TransformType&,
                   const GeomObjectType&);

  GeomObjectType getOriginal () const;

  // Spezifische Daten für die Strukturierung z.B.
  const KDop<REAL,K>& getBoundingBox () const;
```

```
class FaceAdapter<BasicTraits,..>
  typedef ... TransformType;
  typedef ... GeomFaceType;

  GeometryAdapter (const GeomFaceType&);
  GeometryAdapter (const TransformType&,
                   const GeomFaceType&);

  GeomFaceType getOriginal () const;

  // Spezifische Daten für die Strukturierung z.B.
  const KDop<REAL,K>& getBoundingBox () const;
```

Mit einem zusätzlichen Adapterklassentyp können auch Faces des Meshes mit Zusammenhangsinformationen [Botsch and Bischoff, 2002] verarbeitet werden.

Adapterobjekte müssen aus einem Geometry-Knoten bzw. einer Face zusammen mit der Welt-Matrix konstruierbar sein. Für die Rückmeldung von Ergebnissen existiert die Methode *getOriginal*, die das zum Adapterobjekt gehörende OpenSG-Original liefert.

Strukturierung mit Hüllkörpern

Die automatische Strukturierung von allgemeinen Szenen erfordert die Identifikation von sinnvollen, kleinsten Teilen innerhalb der Szene. Werden diese sukzessive zu größeren Teilen zusammengefaßt, so entsteht eine räumliche Hierarchie.

Um eine solche Strukturierung auf allgemeinen Szenen schnell durchführen zu können, werden statt der Teile geometrisch einfache Hüllkörper verwendet, die die Teile jeweils enthalten.

Die Qualität der berechneten Hierarchie hängt dabei unter anderem vom Hüllkörpertyp ab:

- Kugel [Omohundro, 1989]
- Achsenparalleler Quader bzw. seine Verallgemeinerung Diskret-Orientiertes Polyeder (*k-DOP*) [Klosowski *et al.*, 1998]
- Beliebig-Orientierter Quader [Gottschalk *et al.*, 1996]
- Konvexes Polyeder

Außerdem erlauben Hüllkörper sowohl Strukturierungsalgorithmen, die von-oben-nach-unten (*top-down*), und solche, die von unten-nach-oben (*bottom-up*) vorgehen.

GENVIS verwendet binäre Hierarchien aus k-DOPs, die mit verschiedenen top-down Strukturierungsalgorithmen erzeugt werden können. Gegeben einen Hüllkörper, der n Primitive enthält, muß dieser in zwei Teilgruppen mit eigenen Hüllkörpern unterteilt werden. Die Bewertung aller möglichen Unterteilungen in zwei Teilgruppen verbietet sich wegen der exponentiellen Anzahl ($\frac{1}{2}(2^n-2)$) von Möglichkeiten [Klosowski *et al.*, 1998]. Aus diesem Grund wird die Unterteilung anhand von Projektionen auf eine kleine Anzahl von Achsen durchgeführt.

LongestSideMedian Als Achsen werden die Koordinatenachsen x,y,z verwendet. Die Unterteilung erfolgt entlang der Achse, bezüglich der der Hüllkörper am längsten ist. Unter den $n-2$ Unterteilungspositionen wird stets die Position $\lfloor \frac{n}{2} \rfloor$ gewählt.

LongestSideMean Die Wahl der Achse bezüglich der unterteilt wird erfolgt wie bei *LongestSideMedian*. Als Unterteilungsposition wird das arithmetische Mittel der Hüllkörperzentren (projiziert auf die Unterteilungsachse) verwendet.

WeightedSurfaceArea Als Unterteilungsachse $axis$ und Unterteilungsposition p werden Werte $axis \in \{x,y,z\}$, $p \in \{1,\ldots,n-1\}$ mit minimalen Kosten

$$C(axis,p) = \frac{1}{\text{area}(H_{1,\ldots,n})}(\text{area}(H_{1,\ldots,p}) \cdot p + \text{area}(H_{p+1,\ldots,n}) \cdot (n-p))$$

gewählt. Diese Kosten $C(axis,p)$ sind eine obere Schranke für die Kosten eines Strahlschnittes mit dem Hüllkörper $H_{1,\ldots,n}$, unterteilt in $H_{1,\ldots,p}$ und $H_{p+1,\ldots,n}$ [Müller and Fellner, 1999].

WeightedVolume Für den Schnitt mit einem Testkörper ist eine Unterteilung entsprechend der Hüllkörpervolumina sinnvoll. In [Omohundro, 1989] wird gezeigt, daß die mittlere Anfragezeit $O(\sum_{\{H\text{Hüllkörper der Hierarchie}\}} vol(H))$ für einen (im Szenenhüllkörper) zufällig gewählten Testpunkt ist. Die Schnittkosten mit dem Hüllkörper $H_{1,\ldots,n}$, unterteilt in $H_{1,\ldots,p}$ und $H_{p+1,\ldots,n}$ sind

$$C(axis,p) = \text{vol}(H_{1,\ldots,p}) + \text{vol}(H_{p+1,\ldots,n})$$

Ob ein Hüllkörper unterteilt wird, wird über Schranken wie die maximale Tiefe und die maximale Anzahl von Primitiven im Hierarchieknoten gesteuert.

Abbildung 5 zeigt die Ebenen $2,4,6,8$ der 6-DOP Hierarchie (achsenparallele Quader), berechnet mit *LongestSideMean*, *WeightedSurfaceArea*. Um von einer binären zu einer 2^i-ären Hierarchie $(i>1)$ zu kommen, besteht die Möglichkeit jeweils $i-1$ Hierarchieebenen zusammenzufassen. Eine Hierarchie mit Tiefe d hat nach der Modifikation die Tiefe $\frac{d}{i}$. Dies ist insbesondere bei der Umstrukturierung des Szenengraph ein erwünschter Effekt, weil eine flachere Hierarchie geringere Traversierungskosten hat.

Reguläre Raumunterteilung

Neben der vom Szeneninhalt getriebenen Strukturierung mit Hüllkörpern gibt es Anwendungen für eine reguläre Raumunterteilung. Mittels einer regulären Raumunterteilung wird der Szeneninhalt in einen nahen Szenenteil und einen umgebenden Szenenteil unterteilt [Wimmer *et al.*, 1999]. Für den umgebenden Szenenteil werden dann bildbasierte Render-Techniken angewendet. Die reguläre Raumunterteilung kann auch für das Vorausladen von Szenengraphteilen eingesetzt werden.

Gegeben den 6-DOP Hüllkörper der Szene, wird dieser regulär in Quader unterteilt. Die Anzahl m der Quader kann spezifiziert werden über

- Quaderanzahl l_{max} entlang der längsten Seite
- Quaderanzahl l_{min} entlang der kürzesten Seite
- Quaderanzahl u pro Modelleinheit

Der Hüllkörper wird jeweils entlang der längsten Achse halbiert und der Inhalt in die beiden Teilkörper einsortiert. Bei Speicherung des Inhalts in einem Vektor ist die erwartete Laufzeit $O(n \log n)$ [Müller and Fellner, 1999]. Liegt eine Hüllkörperhierarchie auf dem Inhalt vor, dann ist die erwartete Laufzeit $O(n)$ [Müller and Fellner, 1999].

Abbildung 6 zeigt die reguläre Unterteilung eines Molekülmodells.

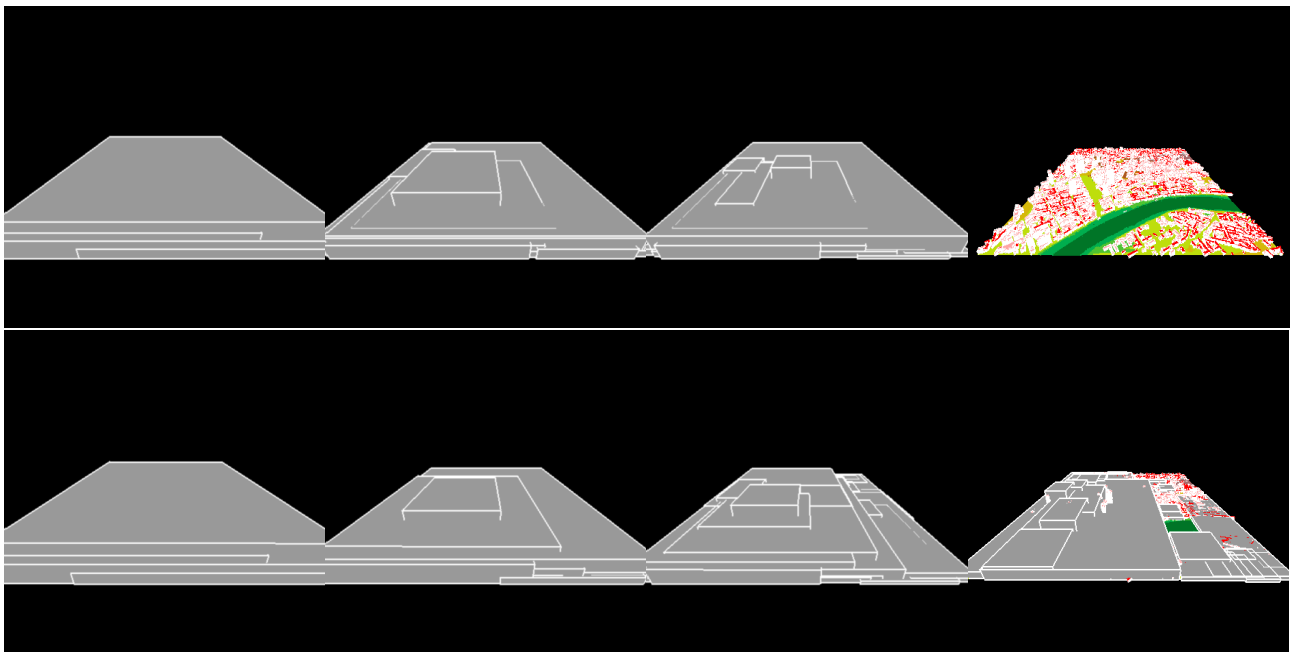

Abbildung 5: Ebenen $2,4,6,8$ der 6-DOP Hierarchie auf dem Stadtmodell von Frankfurt. Oben berechnet mit *LongestSideMean*, unten berechnet mit *WeightedSurfaceArea*, jeweils $maxTiefe = 50$, $maxAnzahlPrimitive = 1024$.

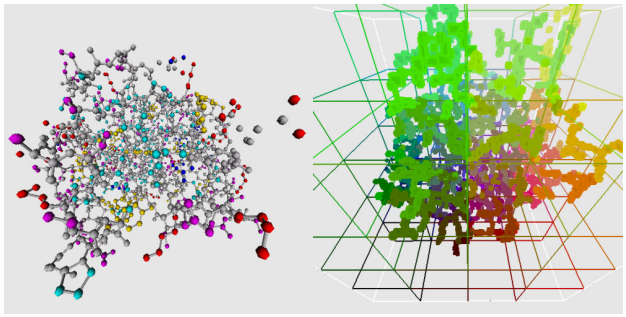

Abbildung 6: Ein Phospholipase-Molekülmodell, strukturiert in ein reguläres Gitter mit 4 Gitterwürfeln pro Modelleinheit.

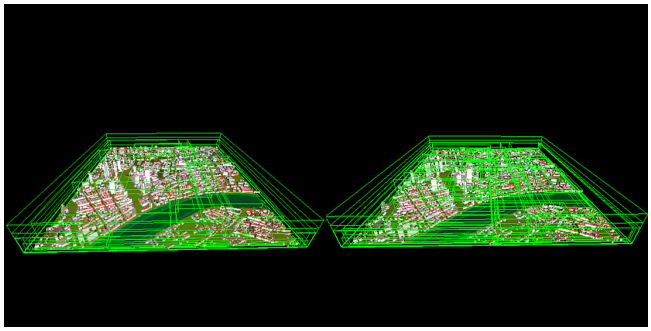

Abbildung 7: Frankfurt-Modell mit der Hüllkörperhierarchie übernommen bis Ebene 4 bzw. Ebene 6. Die Gruppierung nach gleichen Materialien wurde durch Pointervergleich vorgenommen.

Anwendungen

Räumliche Umstrukturierung des Szenengraph

Aus einer einmal berechneten k-DOP Hüllkörperhierarchie kann mittels des Konverters *BVol2OpenSG* ein OpenSG Szenengraph dieser räumlichen Struktur abgeleitet werden. Die Hierarchieknoten werden dabei in OpenSG Group-Knoten und die Faces innerhalb des Hierarchieknoten nach Materialien gruppiert in jeweils einen Geometry-Knoten umgesetzt (Abbildung 7, für die Hierarchie vergleiche Abbildung 5 unten).

Kollisionserkennung

Eine Hauptanwendung der k-DOP Hüllkörperhierarchien ist die schnelle Filterung von Facemengen vor dem eigentlichen Schnittest Punkt–Face bzw. Face–Face [Klosowski *et al.*, 1998].

Abbildung 8 zeigt die Anwendung von GENVIS in der Simulation eines chaotischen Wasserrads. Entlang der Strahlbahn werden Wasserpartikel auf Kollision mit den Reservoirs getestet. Bei Kollision wird ein Wasservolumen zum Füllstand aufaddiert, worauf die Bewegung simuliert wird. Zudem fließt ständig ein eingestelltes Wasservolumen aus allen Reservoirs

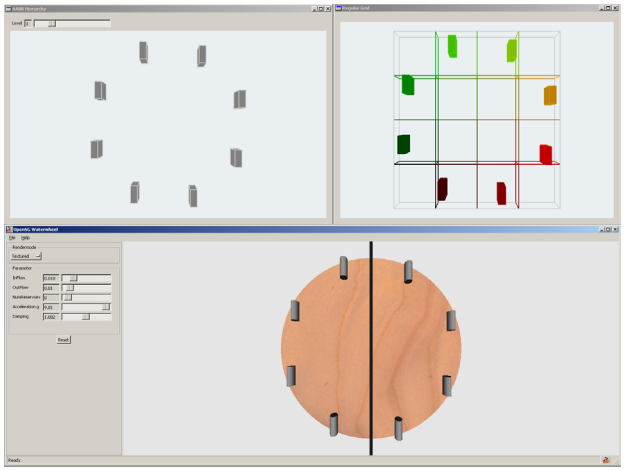

Abbildung 8: Anwendung der Hüllkörperhierarchie bei der Kollisionserkennung in einer physikalischen Simulation des chaotischen Wasserrads.

wieder ab. Für gewisse Modellparameter ist ein chaotisches Verhalten zu beobachten [Fischel, 1997].

Erweiterungen

Bottom-Up Konstruktion von Hüllkörperhierarchien höheren Grades

Bottom-Up Konstruktionsalgorithmen bieten die Möglichkeit jeweils eine Szenen-angepasste Anzahl von Gruppen zu bilden. Die entstehende Hierarchie hat dann höheren, wechselnden Knotengrad [Omohundro, 1989].

BSP-Baum zur Unterteilung in Zellen/Portale

In Innenraummodellen besteht die Möglichkeit mit einer Unterteilung in Zellen/Portale die Zell-Zell und die Kamera-Zell Sichtbarkeiten vorzuberechnen [Teller and Séquin, 1991][Luebke and Georges, 1995]. Für die Vorberechnung müssen die Zellen und Portale der Szene bekannt sein.

Danksagung

Vielen Dank an Gordon Müller für seine Arbeiten zur Hüllkörperstrukturierung. Außerdem gilt mein Dank der Firma T-Mobil für die Erlaubnis zur Benutzung des Frankfurt-Modells.

Literatur

[Blythe *et al.*, 2000] D. Blythe, S. Ghali, L. Kettner, and H. Sowizral. Migrating to an object-oriented graphics api, Jun 2000.

[Botsch and Bischoff, 2002] M. Botsch and S. Bischoff. Online documentation of the openmesh package. Technical report, RWTH Aachen, 2002.

[Fischel, 1997] Georg Fischel. Obervation of the lorenz systems using scientific visualization. Technical report, TU Wien, 1997.

[Gottschalk *et al.*, 1996] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30(Annual Conference Series):171–180, 1996.

[Klosowski *et al.*, 1998] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, January/March 1998.

[Luebke and Georges, 1995] David P. Luebke and Chris Georges. Portals and mirrors: Simple, fast evaluation of potentially visible sets. *Proceedings of the 1995 Symposium on Interactive 3-D Graphics*, April 1995.

[Müller and Fellner, 1999] Gordon Müller and Dieter W. Fellner. Hybrid scene structuring with application to ray tracing. In S. P. Mudur, D. Shikhare, J. L. Encarancao, and J. Rossignac, editors, *Intl. Conf. on Visual Computing ICVC '99*, pages 19–26, Goa, India, February 1999.

[Omohundro, 1989] Stephen M. Omohundro. Five balltree construction algorithms. Technical report, International Computer Science Institute, University of Berkeley, Nov 1989.

[Reiners *et al.*, 2000] D. Reiners, A. Rettig, C. Knoepfle, G. Voss, and J. Behr. Opensg design 0.2alpha, May 2000.

[Staneker, 2001] Dirk Staneker. Ein hybrider Ansatz zur effizienten Verdeckungsrechnung. Master's thesis, University of Tübingen, Computer Science, 2001.

[Teller and Séquin, 1991] Seth J. Teller and Carlo H. Séquin. Visibility preprocessing for interactive walkthroughs. *Computer Graphics*, 25(4):61–68, July 1991.

[Wimmer *et al.*, 1999] M. Wimmer, M. Giegl, and D. Schmalstieg. Fast walkthroughs with image caches and ray casting. *Proceedings of the Eurographics Workshop Virtual Environments*, June 1999.