Modellierung in der Computergraphik

Heidrun Schumann

Ziel der Computergraphik ist es, aus der Beschreibung von künstlichen Szenen Bilder zu generieren. Diese Beschreibungen werden durch einen Modellierungsprozess erzeugt. Heutige Szenen setzen sich aus mehreren Millionen von Primitiven zusammen, so dass es bei der Modellierung insbesondere um das effektive Handling sehr großer Datenmengen geht. Ein grundlegender Ansatz ist die Verwendung von Level of Detail Methoden, die während der Bildausgabe in Abhängigkeit verschiedener Kriterien, geeignete Approximationen eines Objektes bereitstellen. Daneben gibt es weitere Ansätze, wie die punkt-basierte Modellierung oder die Verwendung von so bezeichneten Impostern. Die genannten Ansätze sollen in diesem Beitrag genauer vorgestellt werden.

1. Einleitung

Im Laufe der Zeit und mit zunehmender Leistungsfähigkeit von Hard- und Software entwickelte sich die Computergraphik zu einem eigenständigen und wichtigen Zweig der Informatik mit vielen Anwendungen in den unterschiedlichsten Bereichen. Hauptanliegen ist es, visuell wahrnehmbare Objekte möglichst realitätsgetreu wiederzugeben. Dazu ist es notwendig, die darzustellenden Objekte in einem hohen Detaillierungsgrad zu modellieren. Dies steht im Widerspruch zu den Anforderungen an eine schnelle Bildausgabe, die möglichst in Echtzeit erfolgen soll. Deshalb wurden spezifische Modellierungsansätze entwickelt, die Vereinfachungen vornehmen, um so die Bilderzeugung zu beschleunigen. Hierzu gehören:

- Level of Detail Modelle
- Punktbasierte Modelle und
- Imposter.

Alle 3 Ansätze gehen von Dreiecksnetzen zur Objektbeschreibung aus, basieren aber auf unterschiedlichen Strategien. Level of Detail-Modelle stellen verschiedene Genauigkeitsstufen eines Objektes bereit, die sich in der Bilderzeugung austauschen lassen. Punkt-basierte Modelle ersetzen Dreiecksnetze durch eine Menge von Punkten, die effizient mit modernen Graphikkarten dargestellt werden können. Imposter sind bild-basierte Einheiten, die in einem Präprozess erzeugt und anstelle komplexer Szenenteile in die Bildausgabe eingespeist werden. Jeder Ansatz hat seine Vor- und Nachteile. Darum gibt es zunehmend auch hybride Vorgehensweisen, die verschiedene Techniken kombinieren.

2. Level of Detail Modelle

Bei der Bildausgabe werden 3-dimensionale Objekte mit einer perspektivischen Projektion auf einen 2-dimensionalen diskreten Raum abgebildet.

Die Größe des Objektes im Bildraum hängt vom Abstand des Betrachters zu diesem Objekt ab. Mit zunehmender Entfernung des Betrachters wird das Objekt immer kleiner, so dass mehrere Dreiecke in einem Bildpunkt zusammenfallen. In diesem Fall können gröbere Modelle verwendet und Rechenzeit gespart werden. Hierfür ist es sinnvoll, verschiedene Auflösungsstufen eines Objektes bereitzustellen und je nach Bedarf in die Ausgabepipeline einzuspeisen (vgl. Abb.1). Dieses Vorgehen wird unter dem Begriff „Level of Detail Techniken“ (kurz LoD) zusammengefasst.

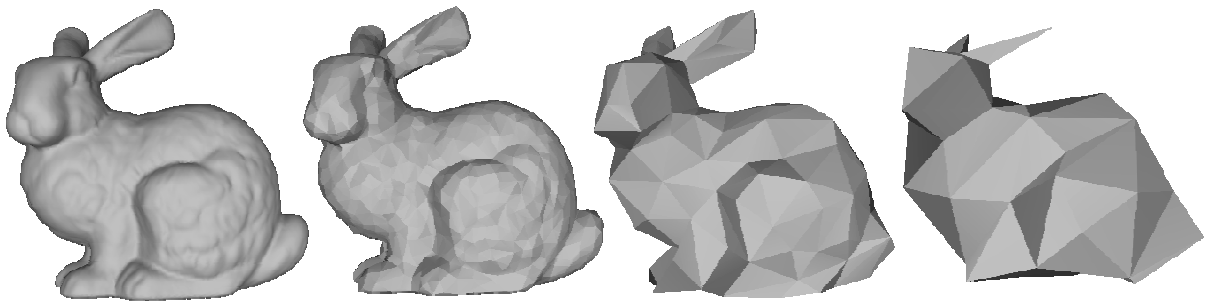

Abbildung 1: Der „Stanford Bunny“ in diskreten LoD-Stufen
(69.451; 2.502; 251 und 76 Dreiecke) aus [1]

Mit Level of Detail Techniken werden 2 Konflikte ausbalanciert:

- Detailreiche Modellierung vs. Echtzeitrendering und
- Detailreiche Modellierung vs. Bildauflösung.

Nur die dem Betrachter nahe gelegenen Objekte werden in einer hohen Auflösung dargestellt, weiter entfernte Objekte werden vereinfacht. Die Vereinfachung entspricht dabei der Bildauflösung, so dass keine Fehler im Bild entstehen. Durch dieses Vorgehen kann die Anzahl der darzustellenden Dreiecke drastisch reduziert und bei gleicher Bildqualität der Ausgabeprozess extrem beschleunigt werden.

Man unterscheidet zwischen diskreten und kontinuierlichen LoD-Techniken.:

- **Diskretes LoD:**
  Hierbei werden verschiedene Auflösungsstufen eines Objektes in einem Präprozess berechnet und zur Laufzeit ein passendes LoD für die Darstellung ausgewählt. In der Regel sind die einzelnen Objektabstufungen sehr grob, um die Anzahl der zu speichernden Objektvereinfachungen in Grenzen zu halten. Dieses Vorgehen wurde bereits 1976 von Clark vorgeschlagen (vgl. [2]) und ist heute in den meisten Graphiksystemen integriert, wobei die Auswahl der LoD-Stufen anhand des Abstandes des Betrachters zum Objekt erfolgt. Das diskrete LoD-Verfahren ist einfach und effizient. Allerdings treten oft so bezeichnete „Popping-Artefakte“ auf, wenn ein Wechsel zwischen unterschiedlichen LoD-Stufen eines Objektes erforderlich ist. Außerdem sind keine lokalen Vereinfachungen möglich.
- **Kontinuierliches LoD:**
  Hierbei wird in einem Präprozess eine effiziente hierarchische Datenstruktur aufgebaut, die das gesamte LoD-Spektrum des darzustellenden Objektes

enthält. Dazu gehören sowohl das Ausgangsnetz in der höchsten Genauigkeit sowie eine Reihe lokaler Vereinfachungsoperationen mit zugehörigen Fehlermaßen. Dieses Vorgehen wurde von Hoppe 1996 vorgeschlagen [3]. Zur Laufzeit wird aus der hierarchischen Struktur eine passende LoD-Stufe extrahiert und dargestellt. Die Vereinfachungen im Dreiecksnetz werden in der Regel durch den „Edge Collapse Operator“ erzielt. Dabei werden 2 durch eine Kante verbundene Punkte im Dreiecksnetz verschmolzen und so die Anzahl der Dreiecke reduziert.

Während beim diskreten LoD Fehler in der Bildausgabe kaum zu vermeiden sind, kann bei kontinuierlichen LoD-Verfahren garantiert werden, dass keine Fehler im Bild entstehen, d.h. die Darstellung eines Objektes in gröberer Auflösung unterscheidet sich nicht von der Darstellung seines Originals. Hierzu werden 2 Fehler ausgewertet: der Objektraumfehler und der Bildraumfehler. Der Objektraumfehler, auch als Geometriefehler bezeichnet, erfasst die Abweichungen zwischen den einzelnen Approximationen eines Objekts im Objektraum und wird im Allgemeinen über die Quadric Error Metric abgeschätzt. Der Bildraumfehler wird in Pixeln angegeben und misst die Abweichungen zwischen Original und dargestellter Approximation im Bildraum. Der Objektraumfehler dient dazu, den Vereinfachungsprozess zu steuern und die Hierarchie aufzubauen. Mit der Abschätzung des Bildraumfehlers wird zur Laufzeit eine geeignete Vereinfachungsstufe eines Objektes, ein so bezeichneter Cut, aus der Hierarchie für die Darstellung ausgewählt.

Die Multi-Triangulation-Struktur von Floriani et al [4] ist eine effiziente Datenstruktur zum Handling einer kontinuierlichen LoD-Beschreibung für ein Objekt. Sie ist ein gerichteter azyklischer Graph (DAG) $G = (N, A)$, wobei jeder Knoten n_i der Knotenmenge N eine Vereinfachungsoperation mit dem dazugehörigen Geometriefehler enthält, und jeder Bogen a_j der Bogenmenge A ein Dreiecksnetz repräsentiert. Die MT-Hierarchie enthält 2 spezielle Knoten: den *Source Node* n_s, dessen ausgehende Knoten das niedrigste LoD enthalten, und den *Destination Node* n_d, dessen eingehende Kanten die höchste LoD-Stufe beschreiben. Abbildung 2 veranschaulicht die Bildung der MT-Hierarchie und die Auswahl von LoD-Stufen (Cuts) an einem kleinen Beispiel.

Analog zu diskreten LoD-Modellen erfolgt auch bei kontinuierlichen LoD-Modellen die Auswahl einer bestimmten Genauigkeitsstufe zur Laufzeit in der Regel anhand des Abstandes vom Betrachter zum Objekt. Daneben können aber auch weitere Kriterien betrachtet werden, wie beispielsweise Beleuchtungsverhältnisse oder die Wichtigkeit von Objekten in einem bestimmten Kontext. Unbeleuchtete oder weniger wichtige Objekte lassen sich in einer gröberen Genauigkeitsstufe darstellen, als beleuchtete oder für die Anwendung besonders interessante Objekte.

Level of Detail Konzepte sind ausführlich in [1] beschrieben. In [6] wird speziell die Problematik zur Bildung und Echtzeitdarstellung von LoD-Modellen für „Out-of-Core-Netze“ behandelt, d.h. für Dreiecksnetze, die so groß sind, dass sie nicht mehr vollständig in den Hauptspeicher passen.

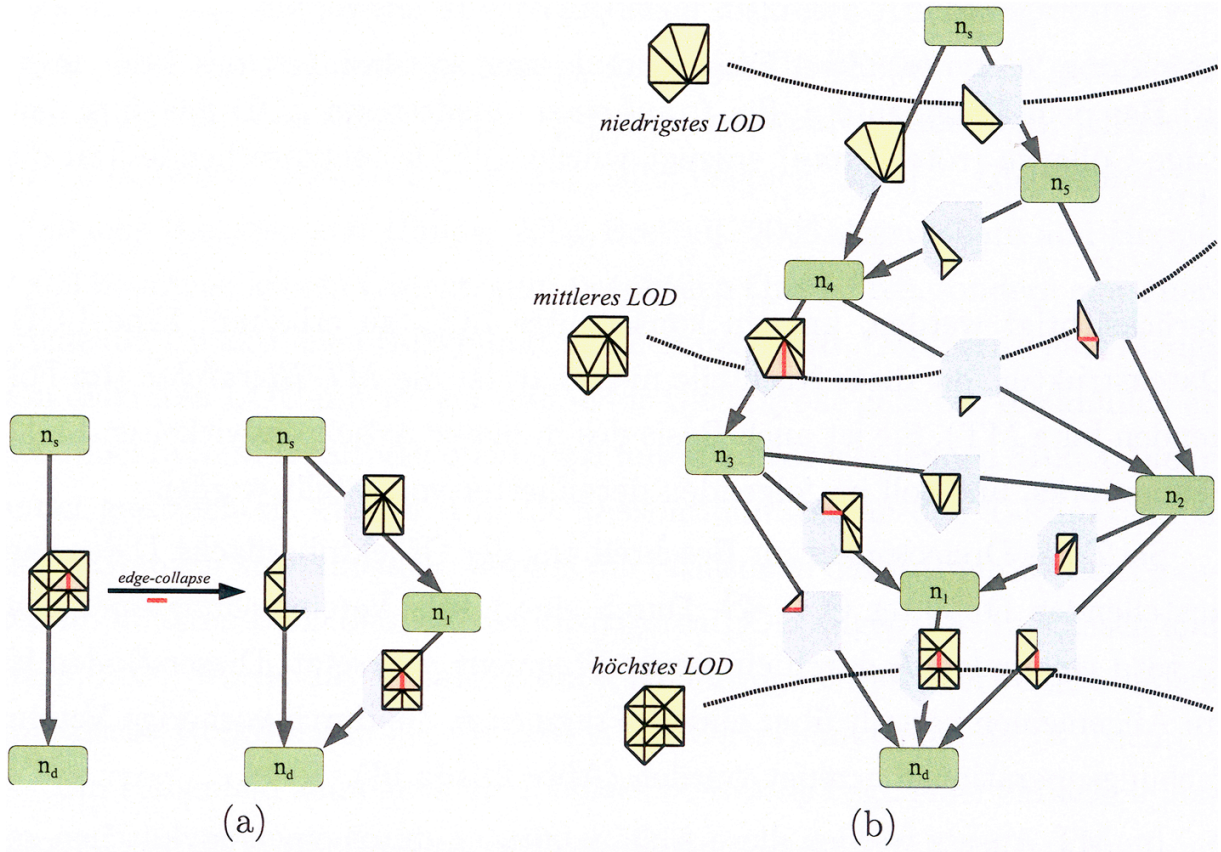

Abbildung 2: Schritte zur Erzeugung einer MT-Hierarchie aus [5] : Ausgangsnetz und erste Vereinfachung (a) sowie MT-Hierarchie (b)

3. Punkt-basierte Modelle

Bei punkt-basierten Ansätzen wird die Oberfläche eines Objektes nicht durch Dreiecke, sondern durch Punkte beschrieben, denen eine spezielle Darstellungsform zugeordnet wird. Punktmodelle wurden bereits 1985 als allgemeines Primitiv für die Oberflächenbeschreibung von 3D-Objekten vorgeschlagen [7], fanden aber erst in neuester Zeit vor allem aus folgenden 2 Gründen eine breitere Anwendung:

- Heutige Graphikkarten unterstützen das Rendern von Punkten, so dass eine schnelle Bildausgabe komplexer punkt-basierter Szenen möglich ist.
- Heutige Modelle sind zum Teil so hoch-aufgelöst, dass mehrere Dreiecke nur auf ein einzelnes Pixel abgebildet werden. Das heißt, in diesem Fall ist die Auflösung des Modells höher als die Auflösung des Bildes, und damit wird der Bilderzeugungsprozess für Dreiecke ineffizient.

Für Punkt-Modelle gibt es 2 Ansatzpunkte: Entweder sind die Objektbeschreibungen von vornherein als Punktmodelle gegeben (das ist z.B. der Fall, wenn Objektbeschreibungen mit einem 3D-Scanner erzeugt wurden) oder die Punktmodelle werden aus Dreiecksnetzen gewonnen. In diesem Fall werden Dreiecke sukzessiv

durch Punkte ersetzt. Die Punkte werden dann stufenweise reduziert und somit eine LoD-Hierarchie aufgebaut. Für die Darstellung werden den Punkten entweder im Objektraum Kreisscheiben oder im Bildraum Pixelbereiche zugeordnet in der Form, dass bestimmte Kriterien, wie z.B. eine geschlossene Objektoberfläche, eingehalten werden können (vgl. Abbildung 3).

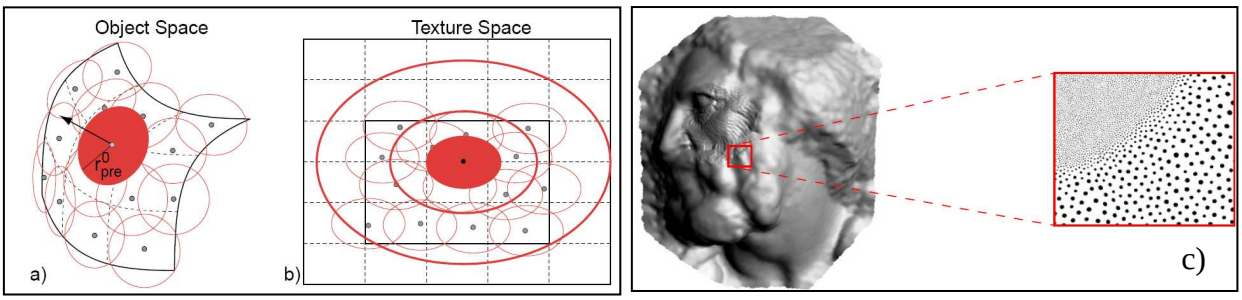

Abbildung 3: Definition von punkt-basierten Oberflächen im Objektraum (a), im Bildraum (b) und zur Darstellung eines Objektes (c).

Analog zu den im Abschnitt 2 eingeführten LoD-Konzepten werden auch für Punktmodelle Fehlermaße eingeführt und ausgewertet, um eine bestimmte Darstellungsqualität zu garantieren. Abbildung 4 zeigt ein Beispiel einer Punkthierarchie für ein 3D-Objekt und die daraus erzeugten Bilder.

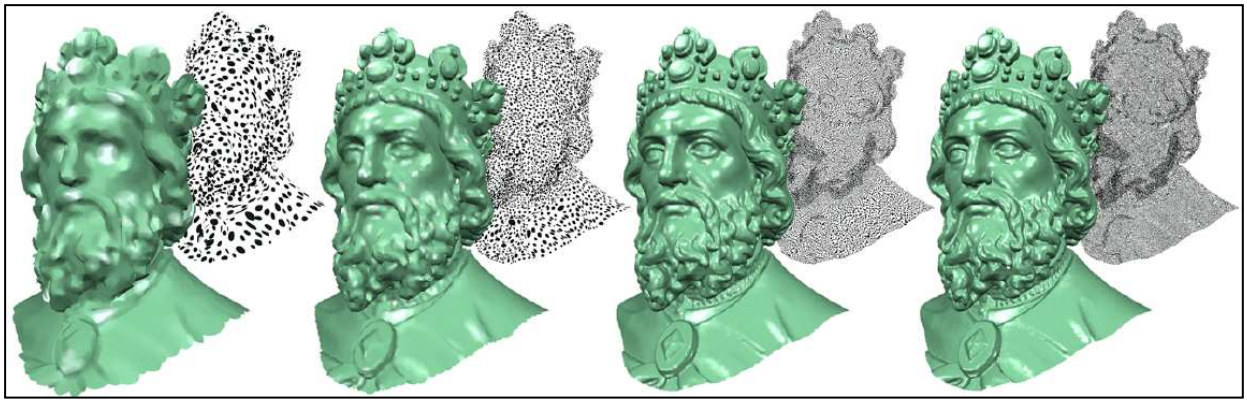

Abbildung 4: Beispiel einer Punkthierarchie aus [8]

Punkt-basierte Modelle sind ausführlich in [9] beschrieben. Sie werden auch häufig in Kombination mit den oben beschriebenen LoD-Modellen für Dreiecksnetze eingesetzt. Hochaufgelöste Modelle werden dann durch Dreiecke repräsentiert und dargestellt. Einfachere Modelle werden mit Punkten repräsentiert. In [5] wird zudem vorgeschlagen, für die Repräsentation von Punkten so bezeichnete Surfel-Billboards zu verwenden und damit den punkt-basierten Ansatz zusätzlich auch noch mit einem bild-basierten Ansatz zu verknüpfen.

4. Imposter

Imposter sind bild-basierte Einheiten, die als alternative Repräsentation für Szenenteile mit hoher geometrischer Komplexität benutzt werden, um so die Anzahl der darzustellenden Primitive zu reduzieren und die Ausgabe zu beschleunigen. Zur Imposterersterstellung wird vom entsprechenden Szenenteil zunächst ein Bild erzeugt und dieses dann in die Ausgabepipeline eingespeist (vgl. Abbildung 5).

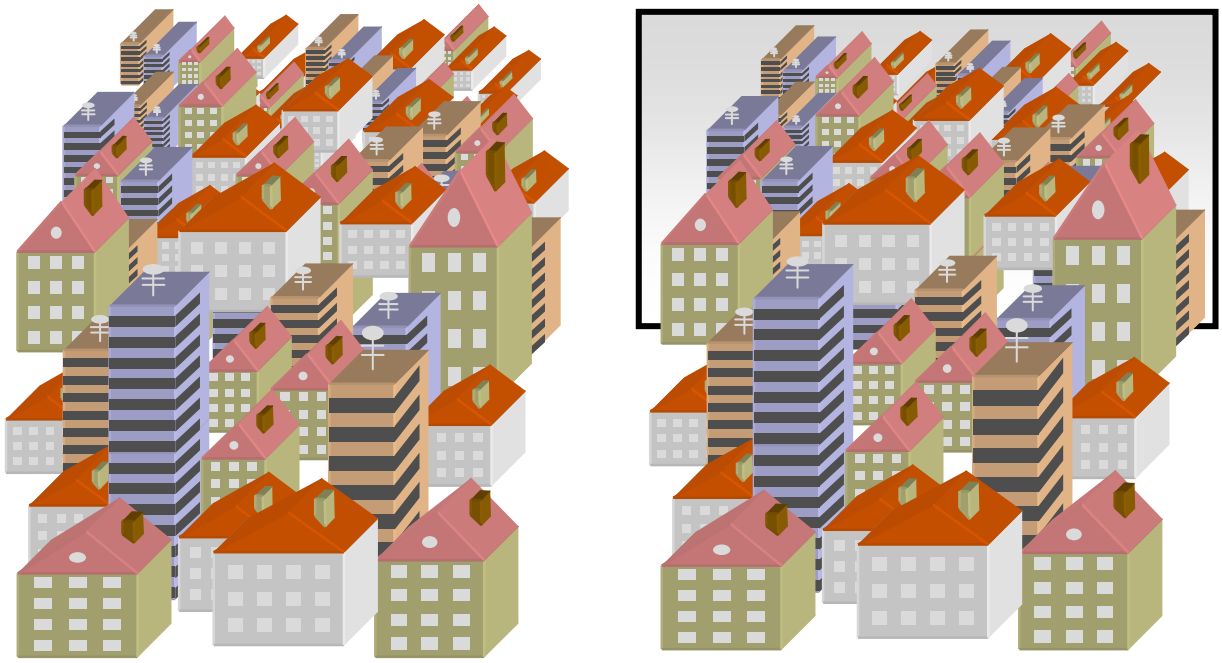

Abbildung 5: Veranschaulichung der Imposterdarstellung mit einem Beispiel von W. Purgathofer: Darstellung der Originalgeometrie (links) und Darstellung mit Imposter für die schwarz eingerahmte Geometrie (rechts)

Der Vorteil einer bild-basierten Repräsentation ist die geringe geometrische Komplexität bei gleichzeitig hoher Detailwiedergabe. Bereits 1976 wurde in [10] ein erster bild-basierter Ansatz zur Oberflächendarstellung eingeführt, um die Darstellungsgeschwindigkeit einer Oberfläche von ihrer Komplexität zu entkoppeln. Heute werden bild-basierte Einheiten, auch als Billboards bezeichnet, nahezu in allen Computerspielen eingesetzt. In Abbildung 6 ist das prinzipielle Vorgehen der Bilderzeugung mit Impostern dargestellt.

Die Ersetzungsstrategie für Imposter geht in der Regel folgendermaßen vor: Die Szene wird in disjunkte Sichtbereiche unterteilt, in denen sich der Betrachter frei bewegen kann, und in denen dieselben Imposter gültig sind. Für jede Sichtzelle werden von einem gegebenen Referenzpunkt aus weiter entfernt liegende Objektteile als Bild aufgenommen und in den Image Cache geladen. Objekte in der näheren Umgebung einer Sichtzelle werden als Geometrie ausgegeben. Beim Wechsel von einer Sichtzelle zur nächsten wird der Image Cache aktualisiert und enthält nun die Imposter der neuen Sichtzelle. Um die Gültigkeit der Imposter zu vergrößern, d.h. um möglichst größere Sichtzellen definieren zu können, nutzt man

auch so bezeichnete *„layered Imposter“*. Hierbei werden die entfernten Szenenteile nicht nur in einem Bild zusammengefasst, sondern in Schichten unterteilt, und für jede Schicht wird ein separates Bild erzeugt.

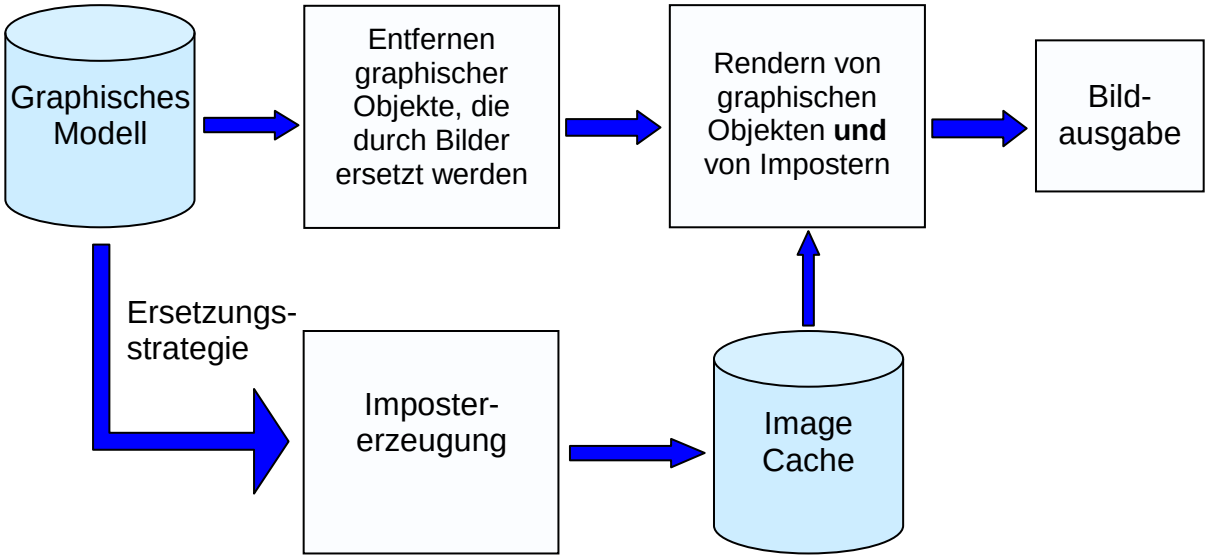

Abbildung 6: Bildausgabe unter Verwendung von Impostern

Beim Einsatz von Impostern sind 3 Anforderungen zu erfüllen bezogen auf:

- Rechengeschwindigkeit:
  Die Darstellung einer Szene mit Impostern muß schneller sein als die Darstellung der entsprechenden Originalgeometrie.
- Bildqualität:
  Im Bild darf ein Imposter nicht von der Darstellung der Originalgeometrie zu unterscheiden sein.
- Speicheraufwand:
  Die Erzeugung von Impostern muss so erfolgen, dass sich der Speicherbedarf in vorgegebenen Grenzen bewegt.

Die erste Anforderung wird in der Regel von allen Ansätzen zur Imposterdarstellung erfüllt. Schwieriger ist es, die Bildqualität vorherzusagen und zu garantieren. In [11] wurden deshalb neue Impostertechniken entwickelt, die auf der Basis spezieller Fehlermetriken erstellt werden, und hierdurch die visuelle Qualität vorhersagbar und quantifizierbar machen. Wenn dann eine Fehlerschranke in der Pixelauflösung angegeben wird, kann garantiert werden, dass die Anforderung 2 vollständig erfüllt ist. Der Speicherbedarf für den Image Cache hängt von der Größe der Sichtzellen und bei geschichteten Impostern zusätzlich von der Anzahl der definierten Schichten ab. Um die Anforderung 3 zu erfüllen, wurden deshalb in [11] Imposter nicht generell für entfernt liegende Objekte erzeugt, sondern es wurde eine allgemeine Strategie entwickelt zur automatischen Auswahl von komplexen Szenenteilen, die sich für die Impostererstellung besonders eignen. Dadurch können zudem interaktive Bildwiederholraten für jede mögliche Betrachterposition und -richtung garantiert werden, da

eine gewisse Szenenkomplexität nicht überschritten wird. Außerdem können die gesamten Kosten zur Impostergenerierung reduziert werden. Damit lässt sich die praktische Nutzbarkeit der Impostertechnik hinsichtlich des Berechnungsaufwandes, des Speicherplatzbedarfes, der Bildqualität und der Anforderungen an verschiedene Anwendungsszenarien anpassen.

4. Schlussbemerkungen

Die vorgestellten Ansätze haben unterschiedliche Eigenschaften. LoD-Techniken beziehen sich in der Regel nur auf einzelne Objekte. Punkt-basierte Modelle sind sehr effizient für natürliche Objekte, wie z.B. Bäume oder für weiter entfernt liegende Objekte. Bei nahen Objekten mit geschlossenen Oberflächen eignen sich aber Dreiecke besser für die Darstellung. Impostertechniken können Szenenteile mit mehreren Objektteilen zusammenfassen, erfordern aber einen nicht unerheblichen zusätzlichen Speicheraufwand. Auf Grund dieser unterschiedlichen Eigenschaften muss anwendungsbezogen entschieden werden, welche Technik zum Einsatz kommen soll.

Referenzen:

[1] D. Luebke, B. Hallen: Perceptually Driven Simplification for Interactive Rendering. Proceedings Eurographics Rendering Workshop, 2001, S. 223-234

[2] J. H. Clark: Hierarchical Geometric Models for Visible Surface Algorithms. Communications of the ACM, 19 (10) 1976, S. 547 - 554

[3] H. Hoppe : Progressive Meshes. Proceedings SIGGRAPH'96, ACM Press, 1996, S. 99-108

[4] L. D. Floriani et al: Building and traversing a surface at variable resolution. Proceedings IEEE Visualization'97 conference, IEEE Computer Society Press, 1997, S. 103-110

[5] M. Holst: Effiziente auflösungsvariable Oberflächenbeschreibung mit hybriden Modellen. Eingereichte Dissertation, Universität Rostock, IEF, 2007

[6] H. Birkholz: Level of Detail und hardware-unterstütztes Out of Core Rendering. Eingereichte Dissertation, Universität Rostock, IEF, 2007

[7] M. Levoy, T. Whitted: The use of points as a display primitive. Technical Report 85-022, Computer Science Department, University of North Carolina at Chapel Hill, 1985

[8] S. Rusinkiewicz ; M. Levoy : QSplat: A Multiresolution Point Rendering System for Large Meshes, Proceedings SIGGRAPH'2000, ACM Press, S. 343 -352

[9] M. Gross et al : Point-based Computer Graphics. Tutorial T6, Eurographics 2002, Saarbrücken, 2002

[10] J. F. Blinn, M. E. Newell : Texture and reflection in computer generated images. Communications of the ACM, 19 (10) 1976, S. 542-547

[11] S. Jeschke : Accelerating the Rendering Process Using Impostors. Dissertation, Universität Rostock, IEF, 2005