Levels of Detail

Christian Troger

Institute of Computer Graphics and Algorithms
Vienna University of Technology
e9526703@stud3.tuwien.ac.at

Abstract

Dieses Paper soll überblicksmäßig die vielseitigen Einsatzmöglichkeiten von *Levels of Detail (LOD)* in der Computergraphik beleuchten. Über die bereits seit längerem benützten geometrischen *Levels of Detail* hinaus, sollen vor allem deren Schwachpunkte aufgedeckt und bessere Lösungen bzw. Erweiterungen vorgestellt werden. Neben der Verwendung im 3D-Bereich wird in einem Kapitel auch die Einsatzmöglichkeit von *Levels of Detail* bei Rastergraphiken beschrieben. Als Abschluß werden die Zukunft von *Levels of Detail* und notwendige Forschungsarbeiten auf diesem Gebiet erläutert.

Keywords:
Levels of Detail, geometrische LOD, simulation LOD, adaptive LOD, MipMapping, Regions of Interest.

1. Einführung

1.1. LOD Definition

Unter *Levels of Detail* versteht man im Allgemeinen die Verarbeitung von Objekten mit unterschiedlichen Detailstufen. Die tatsächlich verwendete Abstufung eines Objekts soll dann durch definierte Bedingungen ermittelt werden. Obwohl die LOD-Methodik in vielen Bereichen angewendet werden kann, so hat sie sich vor allem in der Computergraphik durchgesetzt und findet dort weite Verbreitung. Die Grundidee dabei ist, visuell unwichtigere Objekte in geringerer Detailstufe zu verwenden, um die Darstellungsgeschwindigkeit zu erhöhen. Durch Verwendung von LOD darf jedoch das Verhalten der Objekte und der visuelle Eindruck auf den Betrachter nicht verändert werden, wobei jedoch das Problem auftritt, daß diese zwei Kriterien nicht immer eindeutig messbar sind.

1.2. Warum werden LOD verwendet?

Im Bereich Visualisierung und Virtual Reality war man immer mit dem Problem konfrontiert, daß die zu verarbeitende Datenmenge bei weitem zu groß war, um sie von der Hardware interaktiv darstellen zu lassen. Um die Lücke zwischen benötigter und zur Verfügung stehender Hardware zu schließen, müssen bei interaktiven Darstellungen Abstriche gemacht werden. Dabei werden nun jedoch nicht "globale" Maßnahmen getroffen (z.B. halbieren der Bildschirmauflösung), sondern durch Verwendung von *Levels of Detail* intelligentere Lösungen angewandt, um die bestmögliche Darstellung mit der zur Verfügung stehenden Hardware zu gewährleisten.

1.3. Arten von LOD

1.3.1. Geometric Levels of Detail

Für jedes Objekt in einer Szene werden mehrere LOD-Repräsentationen erzeugt, zur Laufzeit wird dann je nach Entfernung des Objekts vom Betrachter entschieden, welches LOD-Objekt verwendet werden soll. Dieses Verfahren funktioniert zufriedenstellend für VR-Walktroughs in großen, komplexen Strukturen, bei wissenschaftlichen Visualisierungen ist es jedoch meistens unwahrscheinlich, daß sich die Objekte sehr weit vom Betrachter entfernen um ein LOD-Objekt mit geringerer Detailstufe darzustellen (z.B. medizinische Visualisierung). Deshalb werden oft adaptive LOD verwendet, wo verschiedene LOD auf unterschiedlichen Regionen *eines* Objektes erlaubt sind.

1.3.2. Simulation Levels of Detail

Bei geometrischen *Levels of Detail* wird zwar eine Performance-Steigerung durch die Darstellung von einfacheren Objekten erreicht, das Verhalten der Objekte wird jedoch bei den unterschiedlichen Detailstufen nicht berücksichtigt, Simulationsberechnungen sind also in allen Stufen identisch. Auch die Regel für das Wechseln zwischen den Stufen, die nur die Entfernung des Objekts vom Betrachter beinhaltet, ist relativ primitiv. In [1] und [2] wird ein Verfahren vorgestellt wo neben den geometrischen LOD auch *behavior levels of detail* eingeführt werden. Das Prinzip dabei ist, die notwendigen Berechnungen für das Simulationsverhalten der Objekte je nach Stufe unterschiedlich zu definieren, um eine Reduzierung der benötigten Rechenleistung zu erreichen. Zusätzlich wurden auch die Bedingungen für das Wechseln zwischen LOD-Stufen verfeinert und neue Kriterien vorgestellt.

Das größte Problem bei *Simulation Levels of Detail* ist zur Zeit die Tatsache, daß die einzelnen Stufen für jede Simulation meistens individuell zu definieren sind, d.h. dieser Algorithmus ist schwer generalisierbar.

1.3.3. Animated Smooth Level of Detail (ASLOD)

In [4] wird ein ASLOD-Verfahren vorgestellt, das, aufbauend auf einer Kombination von progressiven LOD-Modellen und herkömmlichen Szene-Graphen, speziell für die Real-Time-Animation von menschlichen Körpern ausgelegt ist.

1.3.4. Levels of Detail für Rastergraphiken

- **Texturen:**
  Werden in einer VR-Umgebung Texturen verwendet, so müssen diese bei Entfernung vom Betrachter in ihrer Größe angepasst werden. Die dadurch entstehenden Artefakte können durch Verwendung von *Multiple Lev-*

els of Detail eliminiert werden, indem man je nach Entfernung verschieden große Texturen verwendet. Für diese Texturen in unterschiedlicher Auflösung, hat sich der Begriff *mipmaps* durchgesetzt. [14] [16]

- **Regions of Interest (ROI):**

  In der Bildverarbeitung werden sehr oft ROI verwendet, um zeitaufwendige Operationen auf bestimmte Bereiche im Bild zu beschränken. In [8] wird die ROI/LOD-basierte Methode *Rectangular fish eye view* für eine optimierte Bildbetrachtung und Bildübertragung vorgestellt.

- **Wavelets:**

  Verwendet als Bildkomprimierungs-Verfahren anstatt dem JPEG-Standard, da Wavelets zwar mehr Rechenleistung beanspruchen, aber eine bessere PSNR (Peak Signal Noise Ratio) als JPEG aufweisen, und bei gleicher Kompressionsrate JPEG-typische Blockartefakte vermieden werden. Da Wavelets an sich kein eigenes LOD-Verfahren darstellen, sondern nur gewisse Ähnlichkeiten aufweisen, wird in diesem Paper nicht näher darauf eingegangen.

2. Geometrische Levels of Detail

Bei geometrischen LOD werden Vereinfachungsalgorithmen iterativ auf ein Objekt angewandt, um eine schrittweise Hierarchie von immer gröber werdenden Objekten zu erhalten. Diese Hierarchien werden in LOD-basierten Rendering-Verfahren verwendet, um höhere Frame-Update Raten bei gleichbleibendem Realismusgrad zu erhalten. Dazu wird die aktuelle visuelle Bedeutung eines Objekts ermittelt, um so den entsprechenden *level of detail* auszuwählen. Je wichtiger ein Objekt für den User dabei ist, desto höher wird die Detailstufe gewählt.

2.1. Erzeugung von LOD

2.1.1. Manuell (vorberechnet)

Dabei werden zur Design-Zeit Objekte mit unterschiedlicher Polygonzahl erstellt. Es können optimale Detailstufen erzeugt werden, der Aufwand dafür ist aber sehr hoch.

2.1.2. Automatisch (vorberechnet)

Ebenfalls zur Design-Zeit werden, ausgehend von einem Originalobjekt, automatisch gröbere LOD-Objekte erzeugt. Hierfür existieren verschiedene Algorithmen, wobei das Ziel ein guter Kompromiß zwischen Schnelligkeit und Genauigkeit ist. Unter Umständen müssen allerdings Abstriche bei der Schnelligkeit in Kauf genommen werden.

2.1.3. Dynamisch

Die unterschiedlichen LOD-Objekte werden auch hier automatisch erzeugt, allerdings zur Laufzeit. Dynamische Berechnungen dauern allerdings etwas länger als die Berechnungen zur Design-Zeit.

2.2. Uniform Levels of Detail

Dabei wird ein universeller LOD-Erzeugungs-Algorithmus verwendet, d.h. es werden keine spezifischen Eigenschaften der Objekte berücksichtigt, sondern im Prinzip nur ein Objekt durch immer gröbere Objekte ersetzt und eher wahllos die Anzahl der Polygone verringert. Die Algorithmen können keine selektive Zerlegung eines Objekts vornehmen und beziehen sich immer auf das komplette Objekt. Dabei kann es vorkommen, daß wichtige geometrische Eigenschaften verloren gehen, die auch bei der qualitativ schlechtesten Stufe vorhanden sein sollten.

2.3. Adaptive Levels of Detail

Uniform Levels of Detail sind nur optimal für VR-Walkthroughs und große, komplexe Strukturen mit vielen Objekten. Bei wissenschaftlichen Visualisierungen, wo meist nur ein Objekt mit vielen Polygonen (mit der immer gleichen Entfernung vom Betrachter) dargestellt wird, bringt dieser Ansatz keine Verbesserungen. Hier wäre es von Vorteil, wenn unterschiedliche *levels of detail* auf verschiedenen Regionen *eines* Objekts definiert werden könnten. In [15] wird ein Verfahren vorgestellt, mit dem adaptive Vereinfachungen auf unterschiedlichen Regionen eines Objekts konstruiert werden können.

2.3.1. Wahl der Detailstufe

Normalerweise wird hier nur die aktuelle Bildschirmgröße des Objekts zur Wahl des darzustellenden LOD herangezogen. Da beim adaptiven LOD-Verfahren aber davon ausgegangen wird, daß sich das Objekt nur selten vom Betrachter entfernt, müssen hier neue Kriterien für die Wahl der Detailstufe definiert werden:

- **Local Illumination**

  Die Erhöhung der Detailierung ist proportional zum Beleuchtungsgrad auf der Oberfläche des Objekts, d.h. bei scharfen Lichtübergängen müssen mehr Details dargestellt werden, die Anzahl der Polygone in dieser Region also erhöht werden.

- **Visibility Culling**

  Nicht sichtbare Bereiche des Objekts (in vielen Fällen nahezu gleich groß wie die sichtbaren Bereiche) können stark vereinfacht in der schlechtesten Detailstufe (oder auch gar nicht) dargestellt werden.

- **Silhouette boundaries**

  Silhouetten spielen eine entscheidende Rolle bei der Wahl der Detailstufe. Die projezierten Längen der Kanten der Silhouette können verwendet werden, um die Glätte der Silhouette-Grenzen zu bestimmen und die Anzahl der Polygone in diesem Bereich festzulegen.

2.4. Beispiele

Anhand eines sphärischen Objekts soll der Vorteil von adaptiven LOD gegenüber herkömmlichen (uniform) LOD aufgezeigt werden. Das in *figure1* dargestellt Originalobjekt wird in ein Objekt mit ca. 500 Polygonen umgewandelt (*figure2* und *figure3*). Dabei ist deutlich zu sehen, daß bei den adaptiven LOD ein wesentlich besseres Ergebnis durch Erhöhung der Polygonzahl im hellen Punkt erreicht wird.

3. Simulation Levels of Detail

Bei ausschließlicher Verwendung von geometrischen LOD tritt das Problem auf, daß bei Objekten die mit geringerer Detailstufe dargestellt werden, Simulationsberechnungen (z.B. Bewegung) immer gleich durchgeführt werden wie beim Objekt mit geringster Entfernung vom Betrachter. Es wird daher entweder der Simulationsgrad für Objekte eingeschränkt, oder es wird

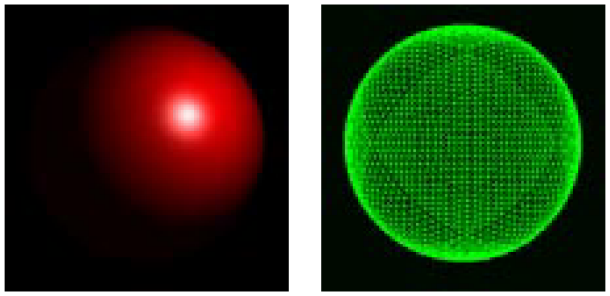

Figure 1: *8192 Polygone (Originalobjekt)*

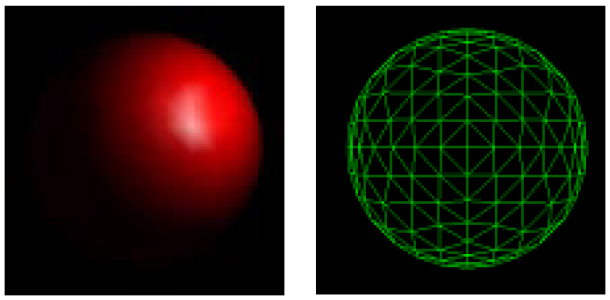

Figure 2: *(b) 512 Polygone (uniform LOD)*

mehr Rechenleistung für exakte Simulationen beansprucht. Als Lösung für dieses Problem wird in [1] und [2] Simulation LOD vorgestellt, die das Simulationsverhalten der Detailstufen anpassen und auch Verbesserungen bezüglich Verwendung von geometrischen LOD aufweisen.

3.1. Erzeugung von LOD-Stufen

Es wird nicht einfach die Anzahl der Polygone verringert, sondern die Abstufung für Geometrie *und* für das Simulationsverhalten vorgenommen. Beim geometrischen Vereinfachen wird außerdem darauf geachtet, daß wichtige Elemente auch beim gröbsten Modell noch vorhanden sind. Dadurch ist auch sichergestellt, daß beim Wechsel in eine schlechtere Detailstufe die Silhouette des Objekts erhalten bleibt, was bei einer automatischen Verringerung der Polygonzahl manchmal ein Problem darstellt.

Im Gegensatz zu den meisten *"bottom-up"*-Implementierungen von LOD (ausgehend vom detailiertesten Objekt werden immer mehr Polygone entfernt), wird eine *"top-down"*-Annäherung verwendet, d.h. aufbauend auf ein sehr einfaches Modell eines Objekts, werden immer mehr geometrische Details hinzugefügt und das Simulationsverhalten erweitert.

Die Wahl, wie genau die Verfeinerung von Verhalten und Geometrie sein soll, ist applikationsspezifisch und muß berücksichtigen, ob das Hauptziel realistische Visualisierung oder exaktes dynamisches Verhalten ist.

3.2. Formale Beschreibung der Verfeinerung

Mit den Operationen aus *table1* kann definiert werden:

- $G_n = G_i(G_{n-1})$
  G_n ist ein geometrischer LOD, das durch Anwendung der drei Verfeinerungsoperationen aus der um eine Stufe weniger detailierten geometrischen G_{n-1} erzeugt wird. G_0 kann ein leeres Objekt sein.
- $B_n = B_j(B_{n-1})$
  B_n ist ein Verhaltensspezifischer LOD, das durch Anwendung der drei Verfeinerungsoperationen aus der um eine Stufe weniger detailierteren verhaltensspezifischen LOD B_{n-1} erzeugt wird. B_0 kann ein Objekt sein, daß noch kein Simulationsverhalten aufweist.
- $L_n = (G_i, B_j)$
  Ein LOD L_n eines Objekts ist ein Tuple von G_i und B_j. Jede Detailstufe besitzt also eine visuelle Darstellung des Objekts und ein Simulationsverhalten. G_i und B_j müssen kompatibel sein, d.h. B_j kann sich nicht auf ein nicht-existierendes geometrisches Objekt in G_i beziehen, und G_i nicht auf ein nicht-existierendes Verhalten in B_j.

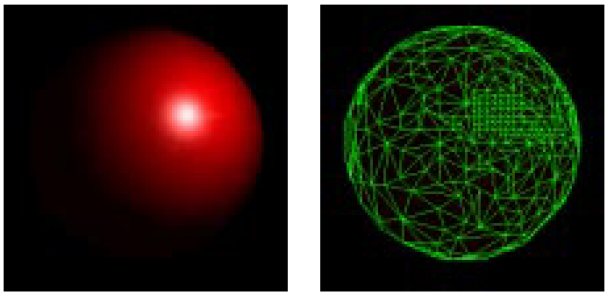

Figure 3: *537 Polygone (adaptive LOD)*

Table 1: *Operationen für die Erzeugung von LOD-Stufen.*

Verhalten	Geometrie
Eigenschaften verfeinern	Objekte zerlegen
Eigenschaften hinzufügen	Objekte hinzufügen
Eigenschaften ersetzen	Objekte ersetzen

3.3. Wechsel zwischen Detailstufen

Normalerweise wird nur die Entfernung eines Objekts als Bedingung für einen LOD-Wechsel verwendet. Um hier mehr Flexibilität zu erreichen, werden folgende Kriterien verwendet:

- Entfernung des Objekts vom Betrachter
- Betrachtungswinkel des Objekts
- Wichtigkeit des dynamischen Verhaltens des Objekts
- Einfluß anderer Objekte auf das Objekt (Wichtig beim Simulationsverhalten)

Zur Vermeidung oszillierender Effekte, wird eine Hysterese bei der Definition der Stufenübergänge verwendet. Um den Wechsel zwischen Detailstufen für den User möglichst transparent zu halten und zur Elimination von "popping"-Effekten, werden spezielle Überblendungen (Farbüberblendungen, Interpolation,...) verwendet.

Beim Wechsel des Simulationsverhalten muß darauf geachtet werden, daß es zu keinen unerwarteten Auswirkungen (aus Usersicht) kommt. Das kann dadurch erreicht werden, indem man den Wechsel nur bei bestimmten Applikationszuständen zuläßt. Es ist jedoch schwierig, hier eine allgemein gültige Lösung für dieses Problem zu finden.

3.4. Beispiel

Zur Veranschaulichung von Simulation-LOD sollen anhand eines Automodells die notwendigen Schritte für die Erzeugung der LOD-Stufen erklärt werden.

3.4.1. LOD-Stufen

- **Stufe 1**

  Geometrie G_0: Das Auto besteht nur aus einem simplen "Gerüst" (24 Polygone)

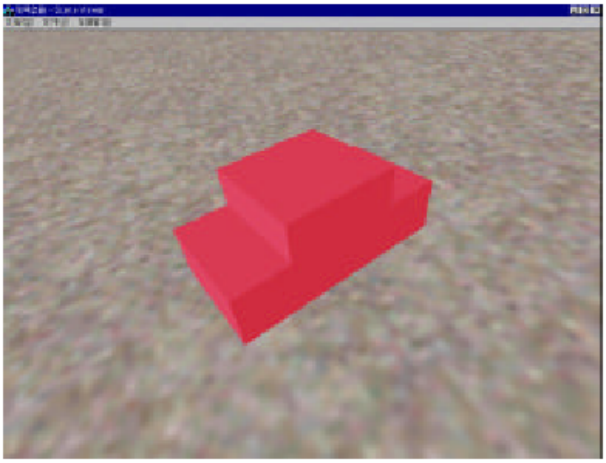

Figure 4: *Geometrie Stufe 1.*

Simulationsverhalten B_0: Das Auto fährt auf vorgegebenem Pfad konstant vor und zurück, das Umdrehen erfolgt "abrupt" (90 Grad). Es findet keine Kollisionsabfrage mit anderen Objekten statt.

- **Stufe 2**

  Geometrie G_1: Das Gerüst wird gegliedert, Räder und Windschutzscheibe werden hinzugefügt (˜242 Polygone)

Figure 5: *Geometrie Stufe 2.*

Simulationsverhalten B_1: Eine Kollisionsabfrage findet nun statt, außerdem kann das Auto für eine bestimmte Zeit stehenbleiben und wieder losfahren. Um den Realismus-Grad zu erhöhen, wird zusätzlich das Umdrehen verfeinert. Das konstante vor- und zurückfahren bleibt unverändert, wird also von Stufe 1 "geerbt".

- **Stufe 3**

  Geometrie G_2: Nummernschilder, Radkappen und Scheinwerfer werden hinzugefügt. (˜432 Polygone)

Figure 6: *Geometrie Stufe 3.*

Simulationsverhalten B_2: Hinzugefügt werden Geschwindigkeitskontrolle basierend auf der Kollisionsabfrage, Animation der laufenden Räder, Berechnung der korrekten Steuerungswinkel und die Erzeugung von dynamischen Pfaden.

3.4.2. Bildung von LOD

Bei der Bildung von L_i muß beachtet werden, welche Eigenschaften von G_i und B_j sich gegenseitig bedingen. (G_0, B_0) wäre zum Beispiel erlaubt, während (G_2, B_0) nicht möglich ist. Als einfachste Lösung können die LOD-Stufen (G_0, B_0), (G_1, B_1) und (G_2, B_2) gebildet werden.

4. Levels of Detail bei Animation (ASLOD)

Das in [4] vorgestelle ASLOD-Verfahren wurde entwickelt, um entfernungsabhängige Detailstufen bei Körperbewegungen und deformierbaren Modelle zu verwenden.

4.1. Merkmale von ASLOD

- Individuelle Detailstufenwahl mit Hilfe von hierarchischen Graphen.
- Approximierung der adaptiven Detailstufenwahl anstatt exakter Berechnung für höhere Performance.
- Rendering von animierten, deformierbaren Modellen (Skeleton Animation).

4.2. Wahl der Detailstufe

Die Wahl der Stufe wird formuliert als Optimierungsproblem:

Maximiere *benefit*, während der damit verbundene *cost*-Aufwand innerhalb eines bestimmten Limits liegen muß. *cost* ist bedingt durch die Graphik-Hardware des Systems, *benefit* wird ermittelt durch die aktuelle Darstellungsgröße des Objekts und die Anzahl der Polygone.

Wie bereits bei adaptiven LOD erwähnt, wird die Wahl der LOD-Stufe nicht pro Objekt, sondern für alle Oberflächen des Objekts getroffen. Da der Rechenaufwand dabei aber sehr hoch ist, wird bei ASLOD das Objekt in Regionen unterteilt, die dann je nach Detailstufe in gröbere Strukturen zerlegt werden (siehe *figure 7*).

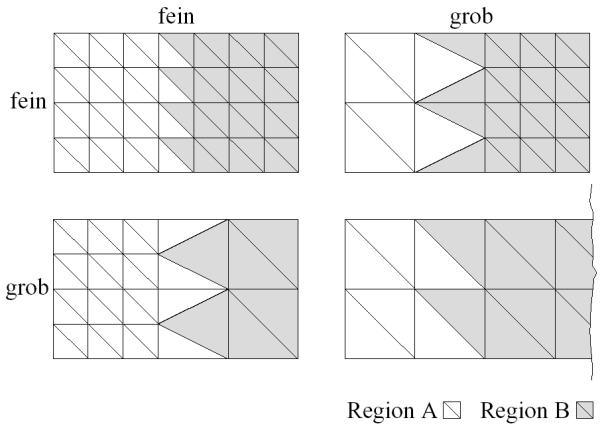

Figure 7: *Erzeugung von LOD-Stufen*

5. Levels of Detail für Rastergraphiken

5.1. Texturen

Der Begriff MipMap wurde zum ersten Mal 1983 von Lance Williams in [16] erwähnt und zählt als fortgeschrittene Technik in Zusammenhang mit Texturen. Dabei wird eine Textur in verschiedenen Größen zur Verfügung gestellt, und zwar in der Grundgröße 1x1, der Originalgröße und allen dazwischenliegenden Größen in Potenzen von zwei. Ein Beispiel soll verdeutlichen, unter welchen Umständen mipmapping verwendet wird. In einer Szene wird eine Textur mit der Größe 64 x 64 Texel verwendet. Die Szene wird animiert und das mit dieser Textur versehene Objekt entfernt sich immer weiter vom Augpunkt des Betrachters - wird somit immer kleiner. Zu einem gewissen Zeitpunkt wird die 64 x 64 Texel große Textur nicht mehr auf das Objekt passen. Eine Möglichkeit wäre sie zu skalieren und anzupassen, doch dabei können störende Artefakte und Performanceeinbussen entstehen. Wenn die Textur in unterschiedlichen Auflösungen zur Verfügung steht, werden diese Nachteile umgangen und die Ausführungsgeschwindigkeit wächst. Mipmapping kann sich allerdings auch als störend erweisen, wenn das Umschalten von einer Texturgröße auf eine andere allzu deutlich sichtbar ist. Es ist daher darauf zu achten, daß genügend viele unterschiedlich große Texturen zur Verfügung stehen.

5.2. Regions of Interest (ROI)

Bei Darstellung von Rastergraphiken ist neben der Performance der Graphik-Hardware, oft auch die Übertragungsbandbreite bei der Übertragung von Bildern ein Problem. Darum müssen Komprimierungstechniken und effiziente Übertragungsmethoden kombiniert werden, um die Antwortzeit bei Bildübertragungssystemen möglichst gering zu halten. [8] bietet einen Überblick über die Probleme, die bei Verwendung von LOD und ROI für eine progressive Bildübertragung auftreten.

Sehr oft ist eine grobe Annäherung (Preview) für den Benützer ausreichend um zu entscheiden, ob das Bild detailierter übertragen werden soll (*Detail on Demand*). Diese Verfeinerung kann auch automatisch erfolgen (*Progressive refinement*).

Als Lösungsvarianten kommen hier zum Einsatz:

- **Regions of Interest:**

  Nur einige Teile des Bildes werden am Anfang detailiert übertragen, der Rest wird automatisch oder auf Wunsch des Users verfeinert. *Beispiel:* das Gesicht in einem Porträt wird detailierter übertragen als der Rest.

- **Levels of Detail:**

  Auf dem Server werden hier mehrere, qualitativ unterschiedliche Bilder gespeichert. Als erstes wird nur das qualitativ schlechteste und damit speichersparendste Bild übertragen. Erst auf Wunsch des Users (bzw. automatisch) werden detailiertere Bilder gesendet. *Beispiel:* Bei Bildgallerien im WWW wird oft eine Liste mit qualitativ minderwertigeren Bildern dargestellt und erst nach Anklicken der Preview wird das detailgetreue Originalbild übertragen.

Basierend auf diesen zwei Varianten wird in [8] die *Rectangular Fish Eye View*-Methode zur Übertragung von Bildern in Netzwerken mit geringer Bandbreite vorgestellt.

- **Grundidee:**

  Zu jedem Zeitpunkt während der Übertragung ist das bereits übertragene Bild in einer bestimmten Detailstufe verfügbar (*global LOD*). Diese globale LOD kann mehrere *regions of interest* beinhalten, wobei jeder Region eine bestimmte *local LOD* zugewiesen wird. Es gibt zwei Arten dieser lokalen LODs. *transmission target* beschreibt den Zustand, wann die Übertragung der entsprechenden ROI gestoppt wird und *transmission state* beschreibt die bereits übertragenen Daten für die ROI. Die Verfeinerung einer ROI wird durch Senden von mehr Daten für die ROI erreicht. (d.h. *transmission state* nähert sich *transmission target*. Die Verfeinerung einer *global LOD* wird erreicht durch Hinzufügen einer neuen ROI oder durch Verfeinern einer bereits existierenden ROI.

- **Rectangular Fish Eye View:**

  Der Darstellungsbreich wird in rechteckige Teile unterteilt, wobei jeder Teil verbundene, nicht überlappende Bildregionen enthält. Im mittleren Rechteck wird die *focus region* des Bilds mit voller Auflösung dargestellt. Die umgebenden Rechtecke (*context area*) stellen die restlichen Teile des Bildes dar, wobei diese Teile in X- und/oder Y-Richtung mit unterschiedlichen Faktoren (Potenzen von zwei) zusammengestaucht werden (*Subsampling*).

Meistens werden bei *regions of interest* nur rechteckige Regionen verwendet. Es wäre aber besser, Rastergraphiken in polygonale Regionen zu unterteilen, um eine optimalere Ausnützung des Verfahrens zu gewährleisten.

6. Levels of Detail bei der Übertragung von Objektgeometrie

Bei den heutzutage verwendeten verteilten VR-Systemen stößt man oft auf performante CPU's, Graphikprozessoren und Speicherelemente, im Gegensatz dazu jedoch auf relativ

langsame Netzwerksysteme. Das führt dazu, daß manche VR-Applikationen nahezu unbrauchbar werden, wenn die Antwortzeit auf Interaktionen des Users mehrere Sekunden beträgt. Um dieses Manko zu beheben, wird in [12] ein Verfahren vorgestellt, das *levels of detail* verwendet um einen äußerst sparsamen Umgang mit der zur Verfügung stehenden Bandbreite zu gewährleisten. Es wird davon ausgegangen, daß nur ein *level of detail* eines bestimmten Objekts zu einer bestimmten Zeit dargestellt werden kann. Daher kann eine Performance-Steigerung erreicht werden, wenn nicht ganze Objekte, sondern nur ihre einzelnen *level of detail* als Übertragungseinheiten angesehen werden. Das Hauptziel des Verfahrens ist, alle Daten in jener Zeit zur Verfügung zu stellen, sodaß für den User kein visueller Unterschied zwischen verteilter und nicht-verteilter VR-Umgebung besteht.

6.1. Data Management

Auf dem Server werden die Daten (Objekte) der VR-Umgebung gespeichert. Da diese Datenmenge sehr groß werden kann, ist es nicht sinnvoll, wenn jeder Client alle Daten zu sich überträgt, sondern nur jene Teile die dargestellt werden sollen. Dazu wird für jeden Client ein spherischer *Area of Interest (AOI)* definiert, wobei sich die Objekte im AOI ändern, wenn sich ein anderes Objekt oder der Client selbst bewegt. Der Client weiß, welche Objekte sich in seinem AOI befinden und kann Update-Requests für diese Objekte an den Server schicken.

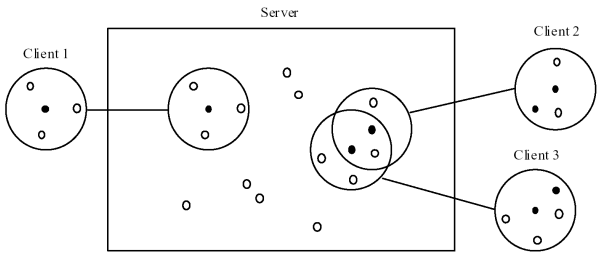

Figure 8: *Verteilte Geometrie-Datenbank: Der Server speichert geographisch verteilt Objekte (kleine weiße Punkte). Jeder Client kennt nur die Objekte aus seiner AOI (große Kreise). Wenn sich AOIs überlappen, können sich die Clients gegenseitig sehen (kleine schwarze Punkte).*

Bewegt sich ein Objekt in ein AOI oder entfernt sich ein Objekt aus dem AOI eines Clients, so wird der entsprechende Client vom Server verständigt.

Um nicht immer komplette Objekte übertragen zu müssen, wird immer nur ein *level of detail* an den Client geschickt. Objekte können dann auch dargestellt werden, wenn nur der gröbste *level of detail* fertig übertragen wurde. Da jedoch auch die Übertragung eines einzelnen *level of detail* zeitintensiv sein kann, wird *prefetching* verwendet. Wenn zu einer bestimmten Detailstufe gewechselt werden soll, wird die nächst feinere Stufe vom Server angefordert. Sollte das *prefetching* fehlschlagen, kann immer noch eine beim Client gespeicherte gröbere Detailstufe dargestellt werden (*graceful degradation*). Wird eine geringe Aktivität am Netzwerk festgestellt, können die noch fehlenden Stufen angefordert werden (*progressive refinement*).

6.2. Geometrische Datenstruktur

Die Datenstruktur wird in zwei Teile geteilt:

- die eigentlichen Daten als Subgraph einer LOD-Node.
- als *control structure* einen *trunk* mit allen LOD-Nodes.

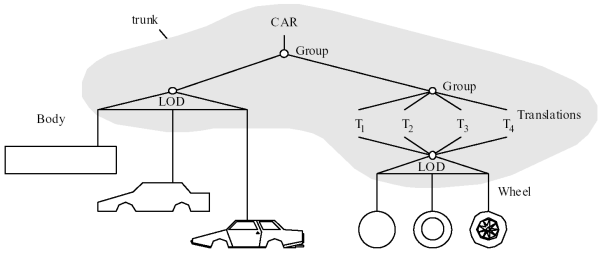

Figure 9: *Auto-Modell mit Levels of Detail. Das Modell ist unterteilt in die LOD-Geometrie und in einen "trunk" (grauer Bereich).*

Bei der Netzwerkübertragung wird zuerst der *trunk* und dann erst die einzelnen LOD übertragen, da der *trunk* und irgendein beliebiges LOD bereits ausreichen, um das Objekt beim Client darzustellen, wenn vielleicht auch nicht in der gewünschten LOD-Qualität.

6.3. Strategie des Clients

Durch den *prefetching*-Algorithmus wird die nächsthöhere Detailstufe bereits angefordert, obwohl noch eine gröbere Stufe dargestellt wird. Dabei kann es zu Problemen kommen, wenn sich der User oder ein anderes Objekt zu schnell bewegt. Auch bei Applikationen die "Objekte aus dem Nichts" zulassen kann es vorkommen, daß die gewünschte Detailstufe nicht rechtzeitig verfügbar ist. Es muß daher eine gröbere Darstellung gewählt werden, die im Normalfall aber vorhanden sein sollte. Das kann jedoch fehlschlagen, wenn es zu einem totalen Netzwerkausfall oder zu einer Netzwerküberlastung kommt. Gerade bei Überlastung ist es für den Client am Wichtigsten, zumindest die gröbste Detailstufe aller Objekte zu holen und erst bei Nachlassen der Netzwerkaktivität die noch fehlenden Stufen anzufordern, da fehlende Details eher zu tolerieren sind als ein Stocken in der Bewegung.

6.4. Cache-Strategie des Clients

Im Cache befinden sich alle für den Client relevanten LOD. Verläßt ein Objekt den AOI, so müssen alle LOD dieses Objekts aus dem Cache dieses Client gelöscht werden. Alternativ könnten auch nur die LOD mit den höchsten Detailstufen des Objekts gelöscht werden um eine Performancesteigerung zu erzielen.

Es muß außerdem darauf geachtet werden, daß der Abstand zwischen Objekt-Eintrittsrand und Objekt-Austrittsrand genügend groß ist, um oszillierende Effekte (bei sich an der AOI-Grenze bewegenden Objekten) zu vermeiden.

6.5. Verbesserungen

Um den Server zu entlasten, könnte die virtuelle Umgebung auf mehrere Server aufgeteilt werden, bzw. eine Parallelverarbeitung durch den Einsatz von mehreren CPUs realisiert werden. Eine Kompression der geometrischen Daten vor der Übertragung über das Netzwerk würde eine zusätzliche Verringerung der benötigten Bandbreite bringen.

7. Ausblick

Das größte Problem bei geometrischen LOD und vor allem bei Simulation-LOD ist zur Zeit noch die Tatsache, daß die einzelnen Detailstufen entweder von Hand, oder automatisch nur sehr applikationsspezifisch erzeugt werden können. Generell gültige Algorithmen sind bei geometrischen LOD zum Teil schon im Einsatz, bei Simulation-LOD werden im Moment Verfahren entwickelt, die eine vom Anwendungsgebiet unabhängige, automatische Wahl der Berechnungsgenauigkeit ermöglichen.

Ein weiteres Forschungsgebiet ist die Loslösung des LOD-Begriffes von der Computergraphik, um die Anwendbarkeit in anderen Gebieten zu untersuchen.

8. References

[1] **Deborah A. Carlson, Jessica K. Hodgins.** Simulation levels of detail for real-time animation. In *Graphics Interface*, S. 1-8, Mai 1997.

[2] **Jinseok Seo, Gerard Jounghyun Kim, Kyo Chul Kang.** Levels of Detail Engineering of VR Objects. Department of Computer Science and Engineering, Pohang University of Science and Technology. 12/1999.

[3] **Chris Faisstnauer, Dieter Schmalstieg, Werner Purgathofer.** Priority Round-Robin Scheduling for Very Large Virtual Environments and Networked Games. Vienna University of Technology.

[4] **Anton Fuhrmann, Dieter Schmalstieg.** Course View-Dependent Levels of Detail for Hierarchical and Deformable Models. Technischer Bericht, Vienna University of Technology.

[5] **Paul S. Heckbert, Michael Garland.** Survey of Polygonal Surface Simplification Algorithms. Technischer Bericht, CS Dept., Carnegie Mellon U., to appear.

[6] **Lee Markosian, Barbara J. Meier, Michael A. Kowalski, John F. Hughes, Loring S. Holden, J.D. Northrup, John F. Hughes.** Art-based Rendering with Continuous levels of Detail. In *Proceedings of NPAR2000.*

[7] **A. E. W. Mason, E. H. Blake.** Automatic Hierachical Level of Detail Optimization in Computer Animation. *Computer Graphics Forum*, 16(3):191-200. Proceedings of Eurographics '97. ISSN 1067-7055.

[8] **U. Rauschenbach, H. Schumann.** Flexible Embedded Image Communication using Levels of Detail and Regions of Interest. In *Proceedings of IMC '98 - Interactive Applications of Mobile Computing*, Rostock Germany, November 1998.

[9] **Uwe Rauschenbach.** Bedarfsgesteuerte Bildübertragung mit Regions of Interest und Levels of Detail für mobile Umgebungen, Kapitel 3: Regions of Interest und LOD in der Bildübertragung. *Dissertation*, Universität Rostock, 2000

[10] **G. Schaufler, W. Sturzlinger, J. Volkert.** Generating Multiple Levels of Detail for Polygonal Geometry. In *Virtual Environments '95 (Eurographics Workshop in Virtual Environments 1995)*, S. 33-41, Berlin, Germany, Januar 1995. Springer Verlag.

[11] **G. Schaufler, W. Stuerzlinger.** Generating Multiple Levels of Detail from Polygonal Geometry Models. *Proc. Second EUROGRAPHICS Workshop on Virtual Environments.*

[12] **Dieter Schmalstieg, Michael Gervautz.** Demand-Driven Geometry Transmission for Distributed Virtual Environments. *Research paper,* Institute of Computer Graphics, Vienna University of Technology, 1996.

[13] **William J. Schroeder, Jonathan A. Zarge, William E. Lorensen.** Decimation of triangle meshes. Volume 26, S. 65-70, Juli 1992.

[14] **Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner.** OpenGL programming guide: the official guide to learning OpenGL, version 1.2. Addison-Wesley, Reading, MA, USA, third Edition.

[15] **Julie C.Xia, Jihad El-Sana, Amitabh Varshney.** Adaptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models. *IEEE Transactions on Visualization and Computer Graphics*, 3(2). ISSN 1077-2626.

[16] **Lance Williams.** Pyramidal Parametrics. *Proceedings of SIGGRAPH 1983.*