„Interactive Multiresolution Mesh Editing“

Seminararbeit für das Fachseminar
„Aktuelle Themen der Graphischen Datenverarbeitung“
basierend auf einem Paper von
Denis Zorin, Peter Schröder und Wim Sweldens
(voraussichtlich SIGGRAPH ‘97).

Leitung: Prof. Dr.-Ing Markus Gross

vorgelegt von
Christoph Burkhalter

SS 1997

Inhaltsangabe / Disposition

1. Einleitung

Das vorgestellte Tool und die beiden grundlegenden Algorithmen werden ganz kurz motiviert und beschrieben.

2. Bisherige Ansätze

H-Splines und Wavelets stellen zwei wichtige Techniken zur Darstellung und Approximation von Oberflächen dar. Einige ihrer Vor- und Nachteile werden hier diskutiert.

3. Datenstruktur

Die Darstellung von Oberflächen kann mittels Patches oder polygonalen Netzen geschehen. In diesem Kapitel werden diese Möglichkeiten sowie die im Paper verwendete Notation und die Implementierung behandelt.

4. Zentrale Transformationen

Dieses Kapitel stellt den Kern dieser Arbeit dar. Es erklärt die beiden zentralen Algorithmen: Synthesis und Analysis. Kurz wird auf die beiden Schemen von Loop und Taubin eingegangen und das Zusammenspiel der Basisalgorithmen erklärt.

5. Struktur des Editors

Neben dem Ablaufdiagramm des Editors wird in diesem Kapitel auf den Vorteil der adaptiven und lokalen Algorithmen eingegangen.

6. Resultate, Schlussfolgerungen und Ausblick

Neben einigen Testresultaten findet man hier eine Bewertung des Editors und eine Anzahl Möglichkeiten, wie man die begonnene Arbeit weiterführen könnte.

7. Anhang: Bilder

Dieser abschliessende Teil beinhaltet graphische Beispiele und Resultate, auf welche (vor allem in Kapitel 6) verwiesen wird.

1. Einleitung

Das Paper von Denis Zorin, Peter Schröder und Wim Sweldens beschreibt ein Tool zur Darstellung von willkürlichen Netzen in mehreren Auflösungen, basierend auf der Subdivision. Dabei soll das Editieren des Netzes auf einem beliebigen Level interaktiv, also in Realzeit möglich sein. Dies bedeutet natürlich, dass die Basisalgorithmen sehr einfach und doch effizient sein müssen.

Erreicht werden soll das durch adaptive und lokale Algorithmen, sodass auch auf weniger leistungsfähigen Computern mehrere Frames pro Sekunde bearbeitet und dargestellt werden können, indem zur Darstellung des Netzes automatisch tiefere Auflösungen verwendet werden.

Die beiden grundlegenden Algorithmen, welche für dieses Tool entworfen wurden, sind Analysis und Synthesis mit den folgenden Aufgaben:

- Analysis berechnet Punkte einer tiefer aufgelösten Stufe, indem es das Netz glättet. Danach wird das Netz wieder subdividiert und die Differenzen zur originalen Darstellung werden gespeichert.
- Synthesis rekonstruiert ein höheres Level, indem es zuerst subdividiert und dann die Differenzenvektoren zu den Punkten dazu addiert.

Diese beiden Algorithmen werden später genauer betrachtet.

Manipulation und Animation von feinen Netzen kann sehr rechenaufwendig sein, daher ist es notwendig, sich einen sehr schnellen Computer zu kaufen, oder die Grösse solcher Netze zu reduzieren, indem man sie an den glatteren Stellen (mit weniger Details) weitmaschiger macht. Hier wird selbstverständlich vom zweiten Ansatz ausgegangen respektive einige Stellen werden auf Kosten von anderen bevorzugt.

Das entstandene Tool basiert auf früheren Arbeiten in den Gebieten Editieren von Netzen auf mehreren Auflösungsstufen, Subdivision von willkürlichen Netzen, Wavelets und Vereinfachung von Netzen [1], und soll die folgenden Eigenschaften haben:

- Kontrolle von mehreren Auflösungsstufen:
  Sowohl das Editieren des Netzes in tiefen Auflösungen als auch das von Details in höheren Auflösungen ist möglich.
- Kompromiss zwischen Geschwindigkeit und Auflösung:
  Die Algorithmen passen sich automatisch den gegebenen Ressourcen an, damit ein interaktives Editieren des Netzes möglich ist.
- Einfachheit:
  Die einzigen Primitiven zur Darstellung der Flächen in allen Auflösungen sind Dreiecke.

2. Bisherige Ansätze

H-Splines: H-Splines erhält man, wenn man höher aufgelöste B-Splines zu einem existierenden B-Spline Patch hinzufügt. Damit können rekursiv sehr komplexe Formen generiert werden. Allerdings kommt mit jeder Rekursionsstufe wieder eine Menge von Offsets hinzu, welche verwaltet werden muss.

Schlussendlich wird das gesamte Netz eher einem aus Primitiven (Dreiecken) aufgebauten Netz als einer Splinefunktion gleichen.

Dieses Schema kann nicht so einfach adaptiv gemacht werden, das heisst die Kontrolle der Details und der verschiedenen Auflösungen hat der Benutzer von Hand zu steuern. Abgesehen davon ist dieser Ansatz zu rechenaufwendig, um ein interaktives Tool zu programmieren.

Wavelets: Wavelets liefern gute Approximationen und schnelle Algorithmen zur Analyse eines Netzes. Wie bei den H-Splines wird auch hier mit Offsets auf der Approximation mit tieferer Auflösung gearbeitet. Diese Konstruktion für ein willkürliches topologisches Oberflächenframework umzusetzen ist nicht einfach.

Den Hauptbeitrag, welchen die Theorie der Wavelets zum Editieren in mehreren Auflösungen eingebracht hat, liegt im Einführen einer Analysis Funktion und deren Verbindung mit der Subdivision.

Basierend auf dieser Kombination entwickelten Zorin, Schröder und Sweldens ihr Tool. Die beiden Funktionen Analysis und Synthesis bilden zusammen den zentralen Beitrag ihres Papers und damit auch den Kern der Implementierung des Editors.

3. Datenstruktur

Die Darstellung, beziehungsweise Approximation einer Oberfläche kann prinzipiell auf mehrere Arten geschehen. Zum ersten einmal mittels **Patches**. Patches sind ein mächtiges Mittel für die Konstruktion von Modellen. Kombiniert mit einer geeigneten Hardware Unterstützung sind sehr schnelle Implementationen möglich. In komplexeren Modellen kann aber der fliessende Übergang auf der Grenze zweier Patches teuer werden. Um keine hochfrequentigen Details zu verlieren, kann man auch verschiedene Patches mittels Displacement Maps kombinieren. Damit umzugehen ist aber in willkürlichen Topologien ziemlich schwierig (vergleiche dazu auch Kapitel 2). Ein Weg dazu wäre der Gebrauch von Spline Oberflächen, wie ihn Charles Loop in [2] ausführlich beschreibt.

Wenn allerdings höhere Auflösungen einer Topologie benötigt werden, kann die zahlreiche Vermehrung von Kontrollpunkten und Patches schnell einmal jede noch so starke Hardware überlasten. In diesen Fällen ist dann die Darstellung mittels Netzen aus Polygonen angebracht. Solche **polygonalen Netze** haben diverse Vorteile: So können beispielsweise willkürliche Topologien sehr intuitiv übernommen und dargestellt werden oder Daten aus einem Laser Scan müssen nicht umgewandelt werden. Allerdings muss man auch sagen, dass solche Netze schnell einmal dermassen umfangreich werden, dass es schwierig ist, sie interaktiv zu manipulieren. Daher wird ein Algorithmus zur Vereinfachung benötigt, welcher einerseits die Anzahl der Polygone in glatten Regionen vermindert, trotzdem aber anderseits die Details sehr genau belässt. Dieser Algorithmus ist die Subdivision.

Patches und polygonale Netze stellen zwei Endpunkte des Darstellungsspektrums dar. Patches beschreiben effizient weite, glatte Flächen, während polygonale Netze zur genauen Darstellung von Details dienen. Die Subdivision dient als eine Art Brücke, welche diese beiden Endpunkte der Skala verbindet.

Denis Zorin, Peter Schröder und Wim Sweldens haben sich für die Darstellung mittels polygonalen Netzen entschieden und sich dabei auf ein einziges Polygon, das Dreieck, beschränkt. Damit sind Anwendungen wie die Vereinfachung eines Netzes, Level of Detail Approximation und selektive Verfeinerung intuitiv lösbar. Für ihr Tool haben die drei nun folgende Datenstruktur entworfen:

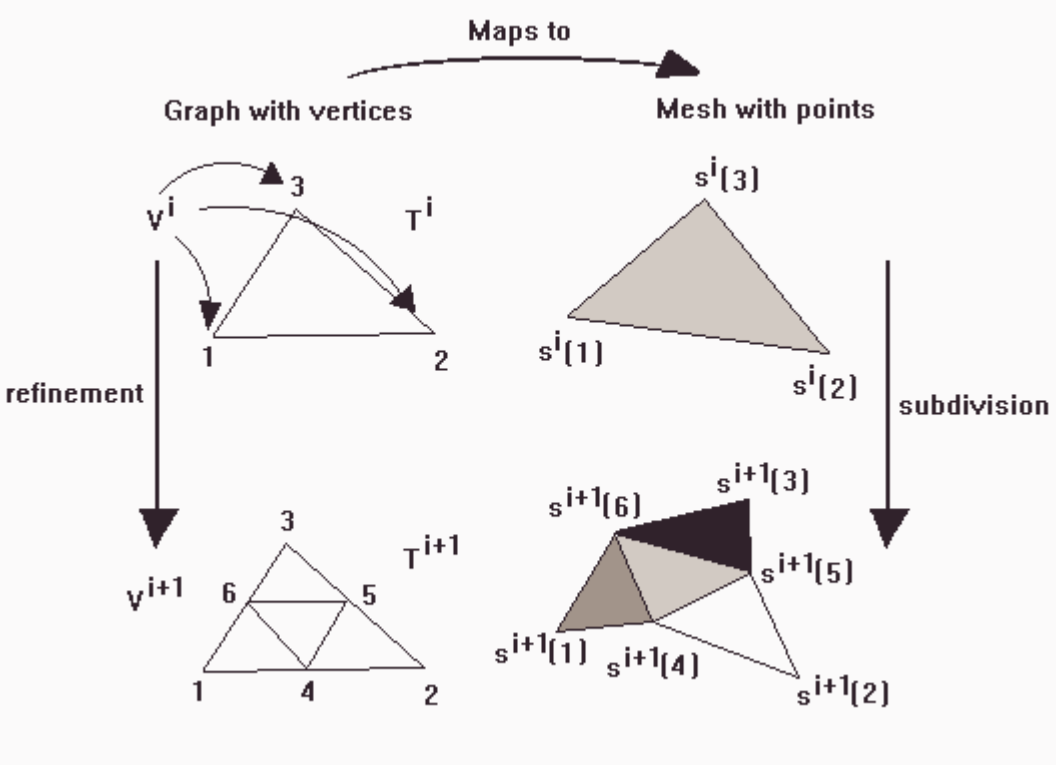
Einerseits haben wir einen abstrakten Graphen, an welchem wir topologische Änderungen durchführen. Anderseits haben wir das Netz als geometrisches Objekt im dreidimensionalen Raum. Dieses Netz verbindet einen Punkt im Raum mit jeder Ecke des Graphen.

Zu Beginn haben wir also einen initialen Dreiecksgraphen T^o mit der Eckmenge V^o und der dazu gehörigen Punktmenge s^0. Rekursiv kann jetzt jedes Dreieck des Graphen in vier kleinere Dreiecke aufgeteilt werden. Damit ergibt sich eine Sequenz von T^i mit den Eckenmengen V^i. Das hochgestellte i zeigt die Stufe der Triangulierung an. Zu bemerken ist vielleicht noch, dass V^{i+1} alle Ecken aus V^i beinhaltet. Jedes Dreieck $t \in T^i$ ist ein Tripel von Indizes: $t=\{v_a, v_b, v_c\} \subset V^i$.

V^{i+1} besteht aus zwei disjunkten Mengen, den geraden Knoten V^i und den ungeraden (d.h. auf dem Level i+1 neu eingeführten) Knoten M^i. M^i ist demzufolge definiert als $V^{i+1} \setminus V^i$.

Die Tiefe eines Knotens ist definiert als das kleinste i, für welches $v \in V^i$ ist. Da jeder Knoten nur einmal hinzugefügt wird, also nur in einem M^i ist, heisst das, dass genau dann wenn der Knoten $v \in M^i$ ist, die Tiefe von v gleich i+1 ist.

Der nächste Schritt ist jetzt die Verbindung des Graphen mit den Punkten im Raum. Für jeden Knoten v und jedes Level i existiert ein Punkt $s^i(v)$ im Raum. Die Menge s^i enthält alle Punkte eines Levels i. Das ist schon alles.



[Figur 1] Die Figur zeigt, wie der Graph und das Netz verbunden sind und wie der Graph verfeinert, respektive das Netz subdividiert wird.

Das Subdivisionsschema lässt sich nun sehr einfach darstellen. Es wird definiert als ein linearer Operator S, welcher die Punktmenge eines Levels i in diejenige des Levels i+1 überführt:

$$s^{i+1} = S\, s^i$$

Wenn wir nun annehmen, dass die Subdivision konvergiert, so wird es eine Limit-Oberfläche σ geben, welche folgendermassen dargestellt werden kann:

$$\sigma = \lim_{k \to \infty} S^k\, s^o$$

$S^k\, s^o$ heisst dabei, dass der Operator S k mal auf die initiale Punktmenge s^o angewendet wird. Als σ(v) wird derjenige Punkt der Oberfläche bezeichnet, welchen die Ecke v repräsentiert.

Was wir jetzt noch brauchen, sind zwei Tangentenvektoren und eine Normale auf die Oberfläche. Dazu führen wir noch einmal zwei lineare Operatoren Q und R ein. Q und R erzeugen zu einem Punkt σ(v) zwei linear unabhängige Tangentenvektoren:

$$q^i(v) = Q\, s^i(v)$$
$$r^i(v) = R\, s^i(v)$$

Zusammen mit der Orientierung $n^i(v)$ definieren diese beiden Tangentenvektoren nun einen lokalen orthonormalen Rahmen:

$$F^i(v) = (n^i(v), q^i(v), r^i(v))$$

Die zentrale Datenstruktur in der Implementierung dieses Tools ist ein Wald von Quadtrees. Nachbarschaftsbeziehungen innerhalb eines Quadtrees werden durch einfaches Traversieren aufgelöst, solche zwischen Quadtrees mittels einer Wurzelsammlung eines ganzen Levels. Subnetze beliebiger Auflösungsstufen können eingefügt werden, indem die Wurzel des entsprechenden Baumes in diese Wurzelsammlung eingefügt wird.

Die Implementation des Tools sollte effizienten Support für Mengenoperationen haben. Konkret wurde dafür die C++ Standart Template Library benutzt. Der Editor wurde mit OpenInventor und OpenGL implementiert und läuft sowohl auf einer SGI, als auch auf einem PC.

4. Zentrale Transformationen

a) Subdivision

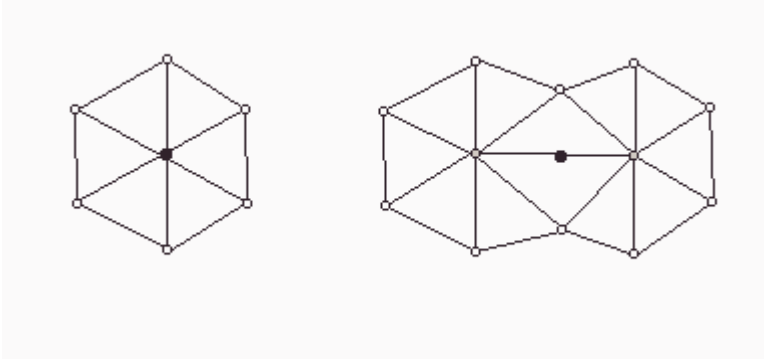
Die Subdivision ist allgemein ein Knoten-Einfüge-Algorithmus. Naive solche Algorithmen haben sowohl was die Zeit, als auch den Speicherplatz anbelangt einen exponentiell grossen Aufwand. Im behandelten Paper wird die Subdivision adaptiv ausgeführt, was dazu führt, dass zum Schluss das Netz an den Stellen mit Details um einiges enger ist, als an glatten Stellen. Dazu wird die Grösse der berechneten Detailvektoren (siehe weiter unten in diesem Kapitel) mit einem Schwellwert verglichen.

Die Features dieses Subdivisions Algorithmus sind:

- Topologische Allgemeinheit:
  Die Knoten benötigen keine Wertigkeit, lokale Berechnungen bringen exakte Grenzen und Normalen hervor.
- mehrere Auflösungen:
  Weil sie das Resultat von Verfeinerungen sind, unterstützen subdividierte Oberflächen direkt Anwendungen wie Level-Of-Detail Rendering, Kompression, Wavelets und so weiter.
- Einfachheit:
  Die Subdivisionsalgorithmen bilden ein feinmaschigeres Netz ganz einfach durch Einfügen von Knoten, gefolgt von lokalen Glättungsdurchgängen.

Als Schema für die Subdivision wurde ein von Loop in [2] vorgestellter Algorithmus verwendet. Das Loop Schema ist ein 1-Ring Schema, das heisst die Punkte auf dem Level i+1 sind nur abhängig von ihren direkten Nachbarn auf dem Level i. Dazu müssen zwei Fälle unterschieden werden:

1. Fall: v auf dem Level i ist gerade $\Rightarrow$ $s^{i+1}(v)$ ist ein Punkt, der nur von den Nachbarn von $s^i(v)$ abhängig ist.
2. Fall: v auf dem Level i ist ungerade $\Rightarrow$ $s^i(v)$ ist auf einer Kante.
   $\Rightarrow$ $s^{i+1}(v)$ ist abhängig von den Kanteneckpunkten und deren direkten Nachbarn. Die Kanteneckpunkte werden dafür stärker gewichtet, als die Nachbarknoten.



[Figur 2] 1-Ring Nachbarn eines geraden (links) und eines ungeraden Knoten (rechts).

Für ein $v \in V^i$ gibt es also die Menge $v_k \in V^i$ der K Nachbarn mit $i \le k \le K$. Es ergeben sich die folgenden Formeln zur Berechnung der auf Level i+1 neu eingeführten Punkte:

Neue Punktmenge:

$$s^{i+1}(v) = \frac{a(K)\, s^i(v) + \sum_k s^i(v_k)}{a(K) + K}$$

$$a(K) = \frac{K(1 - \alpha(K))}{\alpha(K)}$$

$$\alpha(K) = \frac{5}{8} - \frac{(3 + 2\cos(\frac{2\pi}{k}))^2}{64}$$

Limit Punkt:

$$\sigma(v) = \frac{w(k)\, s^i(v) + \sum_k s^i(v_k)}{w(K) + K}$$

$$w(K) = \frac{3K}{8\,\alpha(K)}$$

Tangentenvektoren:

$$t_{p=1,2}(v) = \sum_k \cos(\frac{2\pi(k+p)}{K})\, s^i(v_k)$$

b) Analysis

Während die Subdivision zu höheren Auflösungen führte, wird in diesem Abschnitt Analysis vorgestellt, welches uns von feinmaschigen Netzen zu gröberen bringt. Die Analyse Phase bildet somit eine Reihe von immer grobmaschigeren Approximationen einer Oberfläche, welche immer weniger Kontrollparameter besitzen.

Zuerst muss dazu ein Smoothing durchgeführt werden. Dies kann als lineare Operation H aufgefasst werden, welche ein glattes, grobkörniges Netz auf dem Level i bildet:

$$s^{i-1} = H\, s^i$$

Vergleicht man dies mit der Formel für das Subdivisions Schema in Kapitel 3, so sieht man, dass dies eigentlich die genaue Umkehrung ist.

Für diese Operation können prinzipiell mehrere Vorgehensweisen ausgewählt werden. Die erste Variante wäre ein LeastSquares-Ansatz, welcher die folgende Norm minimieren würde:

$$\min_{H\, s^i} \| s^i - S(H\, s^i) \|^2 = \min_{s^{i-1}} \| s^i - S\, s^{i-1} \|^2$$

Eine zweite Variante ist diejenige, welche für die Implementation des Editors gewählt wurde. Es handelt sich um ein von Taubin in [3] präsentiertes Verfahren, welches lokal glättet, ohne dass dabei die Oberfläche schrumpft.

Um eine geschlossene Kurve zu glätten, reicht es, sich die Koordinaten ihrer Kontrollpunkte als Signale anzusehen und dann das Rauschen aus diesen Signalen herauszufiltern. Das heisst, dass man die Koordinatensignale auf einen Unterraum von tieferen Frequenzen abbildet. Dazu kann die Fourier

Zerlegung der Signale berechnet werden. Dies ist aber eine Methode, welche für die meisten Computersysteme in einer interaktiven, graphischen Anwendungen zu aufwendig ist.

Die von Taubin vorgeschlagene Alternative approximiert eine solche Projektion lediglich. Sie verwendet dazu einen low-pass Filter. Dabei benötigen wir eine Art Durchschnitt aller Nachbarknoten um einen Operator ähnlich dem Laplace-Operator zu konstruieren:

$$\overline{s^i(v)} = \frac{\sum_k s^i(v_k)}{K}$$

$$\mathcal{L}(v) = \overline{s^i(v)} - s^i(v)$$

Auf dieser Basis entsteht nun ein nicht schrumpfender Glätter:

$$H := (I + \mu\mathcal{L})(I + \lambda\mathcal{L})$$

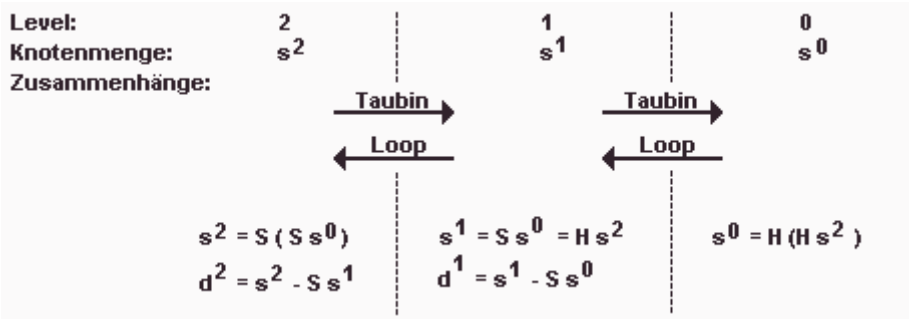
Dieser Glättungsalgorithmus von Taubin ist nicht exakt das Gegenstück zum Subdivisionsfilter von Loop. Dies führt, wie bald genauer ausgeführt wird, zu einer Überrepräsentation der Oberfläche. Für genauere Informationen betreffend Taubin Smoothing verweise ich hier auf [3].

c) Die Vereinigung

Mit der Subdivision und dem Glätten an Ort können wir nun die benötigten Transformationen beschreiben. Die beiden Algorithmen können als eine Art Auf- und Abfilterung angesehen werden.

Was wir benötigen, ist die Differenz zwischen aufeinanderfolgenden Levels in Bezug auf den lokalen orthonormalen Rahmen (vergleiche Kapitel 3) des tieferen Levels. Für jeden Knoten v auf jedem Level i>0 ergibt sich damit ein Detailvektor $d^i(v)$ im Raum. Die Menge d^i enthält alle Detailvektoren eines Levels i. Ein solcher Detailvektor gibt an, um wieviel sich die Punkte auf dem Level i von denjenigen unterscheiden, welche durch Subdivision des Levels i-1 entstehen. Hier ergibt sich die bereits angesprochene Überrepräsentation. Theoretisch wäre es nämlich hinreichend, nur für ungerade Knoten einen Detailvektor zu speichern (was in den Wavelet Transformationen gemacht wird). Dies würde aber die Familie der benutzbaren Glättungsoperatoren verkleinern. Dass dieselbe Oberfläche nun durch verschiedene Mengen von Detailvektoren repräsentiert werden kann, ist kein Problem.

Ich möchte zum Abschluss dieses Kapitels ein Beispiel für die Levels null bis zwei anfügen, damit eventuelle Unklarheiten über das Zusammenwirken der verschiedenen Komponenten bereinigt werden können.



[Figur 3] Hier werden beide Richtungen der Transformation gezeigt. Angenommen s^2 habe N Knoten, dann hat die Menge d^1 noch $^N/_4$ Elemente.

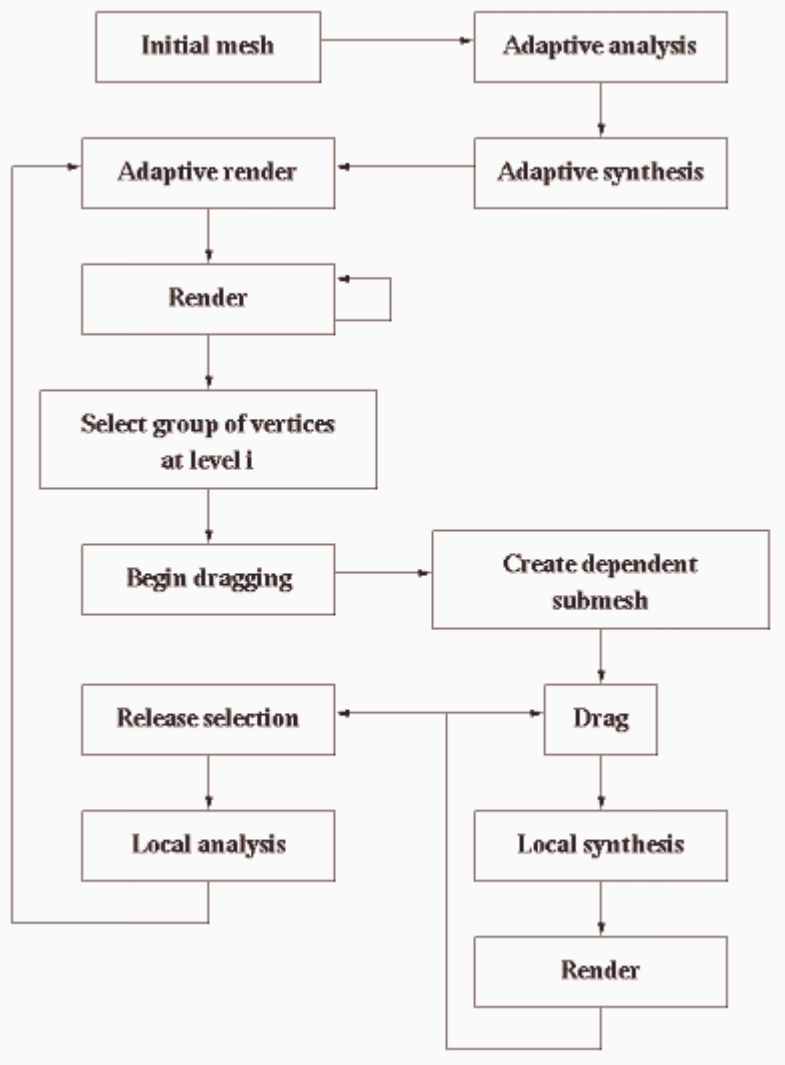
d) Zusammenfassung des Kapitels

Analysis berechnet die Punkte des tiefer aufgelösten Levels i-1 durch Glätten, subdividiert s^{i-1} und berechnet daraus die Detailvektoren d^i.

Synthesis rekonstruiert das Level i indem es das Level i-1 subdividiert und die entsprechenden Detailvektoren hinzufügt.

5. Struktur des Editors

Der Benutzer wählt für die Veränderungen ein geeignetes Level. Zu diesem Zeitpunkt wird die Oberfläche intern durch die Kontrollpunkte dieses Levels und allen Detailvektoren der höheren Levels repräsentiert.
Während des Editierens folgen diese Detailvektoren automatisch ihrem orthogonalen Rahmen nach, die aktuelle Sicht wird von Synthesis gerendert und zwischen den Veränderungen wird die Konsistenz der tieferen Levels mittels Analysis gewährleistet.



[Figur 4] Diese Figur zeigt das Ablaufdiagramm des Editors.

Im Ablaufdiagramm werden lokale und adaptive Varianten der Algorithmen verwendet. Das Konzept der adaptiven und dynamischen Strukturen erlaubt jedem Computersystem, soviel Interaktivität wie möglich anzubieten. Dazu wird auf langsameren Computern die Genauigkeit der Darstellung nötigenfalls vermindert.
Wie im Kapitel 4 erklärt wird, basieren die Algorithmen für die Implementation dieses Editors auf 1-Ring Schemen. Deshalb können sie sich auf lokale Berechnungen beschränken, was wiederum die Performanz erhöht.

Es existieren zwei Kontroll-parameter, von welchen bisher nicht gesprochen wurde.

Der Schwellwert ε_A gibt an, wie klein der kleinste von Analysis noch zu speichernde Deailvektor ist, während ε_S angibt, wie lange ein Netz von Synthesis verfeinert werden soll. Anhand dieser Parameter kann das Konvergenzverhalten eingestellt werden (siehe Limit-Oberfläche σ im Kapitel 3). Durch lokales Erhöhen von ε_S können Linseneffekte erzwungen werden. Dazu findet sich im Anhang ein Beispiel.

6. Resultate, Schlussfolgerungen und Ausblick

a) Resultate

Was ich vorausschicken muss, ist, dass ich den entstandenen Editor selber nie gesehen habe, nur die damit erzeugten Netze und die gerenderten Bilder. Ausserdem habe ich drei Videosequenzen, von welchen behauptet wird, dass sie Echtzeit Ausschnitte aus der Arbeit mit dem Editor sind. Ich werde mich demzufolge auf die Angaben aus dem Paper und den persönlichen Mailings von Denis Zorin und Peter Schröder verlassen müssen.

Die adaptiv gerenderten Versionen enthalten jeweils approximativ gleich viele Dreiecke wie die uniform gerenderten. Die Qualität der Bilder ist aber bei den ersten um einiges höher. Man vergleiche dazu die Wireframes und die gerenderten Bilder im Anhang dieser Arbeit. Zudem sind sogenannte Linseneffekte durch einfaches Anpassen eines Parameters realisierbar. Damit können Regionen von besonderem Interesse genauer modelliert werden als der Rest der Oberfläche. Auch dies wird im Anhang anhand eines Netzes und der dazugehörigen Farbgraphik demonstriert.

Am Beispiel des Gürteltierkopfes wurde der Editor auf einer SGI Indigo R10000 mit 175 MHz und einem PentiumPro mit 200 MHz getestet. Die Parameter des Editors wurden so eingestellt, dass fünf Frames pro Sekunde gerendert werden. Der Kopf besteht aus ungefähr 82’000 Dreiecken. Auf der SGI renderte der adaptive Algorithmus dafür ca. 80’000 Polygone, der PC noch ungefähr deren 20’000. Das positive daran ist, dass mit beiden Computern ein Drehen des Objektes in Realzeit möglich ist. Auf dem PC geschieht dies nota bene mit nur einem Viertel der höchstmöglichen Genauigkeit.

Nun möchte man mit einem interaktiven Editor natürlich auch Kontrollpunkte verschieben können. Dazu wurde jeweils ein Subnetz von 20-30 Flächen verändert. Die Zeit, welche zur Neuberechnung und für das notwendige Rendern benötigt wurde, ist aus der folgenden Tabelle ersichtlich:

Level	0	1	2
SGI	200	100	40
PC	1120	250	70

[Tabelle 1] Die Zeit hängt vom Level ab, auf welchem das Subnetz verändert wird. Sie ist in dieser Tabelle in Millisekunden angegeben.

b) Schlussfolgerungen

Das Ziel, einen interaktiven Editor zu erstellen, welcher sich den jeweiligen Gegebenheiten anpasst, wurde meiner Meinung nach erreicht. Sowohl das von Grund auf Neuerstellen, wie auch des Editieren eines gegebenen Netzes bewerkstelligt das Tool. Auch das Ziel, der Einfachheit halber nur Dreiecke zu verwenden, wurde realisiert. Zu diesem Punkt ist aber zu sagen, dass man dies auch als Nachteil formulieren könnte. Es wäre eventuell doch sehr angenehm und nützlich, wenn man verschiedene Primitive zur Verfügung hätte.

Der Editor kann ein Netz übernehmen, welches man dann manipulieren kann. Möchte nun aber jemand ein Modell aus NURBS oder einer ähnlichen Struktur in eine hierarchische Netz Repräsentation konvertieren, so hilft ihm dieses Tool auch nicht weiter.

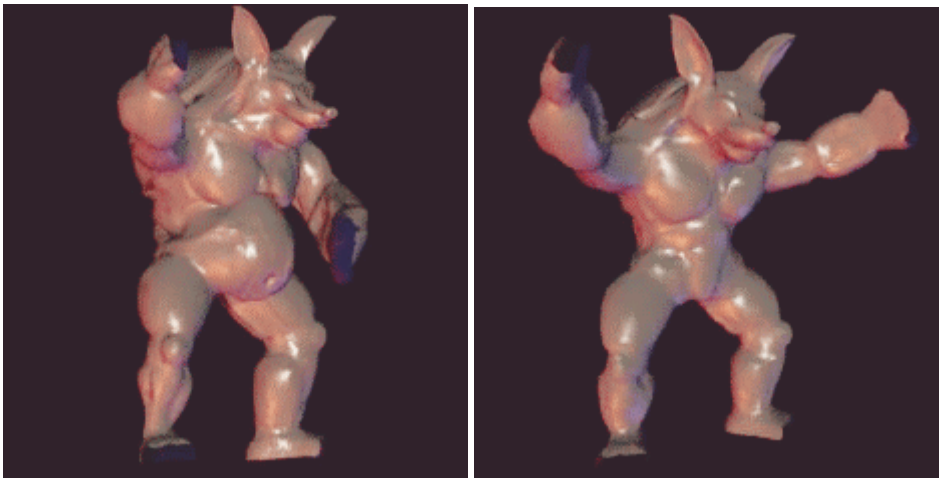
c) Ausblick

Im vorhandenen System sind Veränderungen des tiefsten Levels nur möglich, indem die Knoten dieses Levels direkt editiert werden. Welche Knoten zu welchem Level gehören, also die Tiefe der Knoten, ist aufgrund der Subdivision fix gegeben. Idealerweise sollte der Benutzer diese Tiefe dynamisch verändern können.

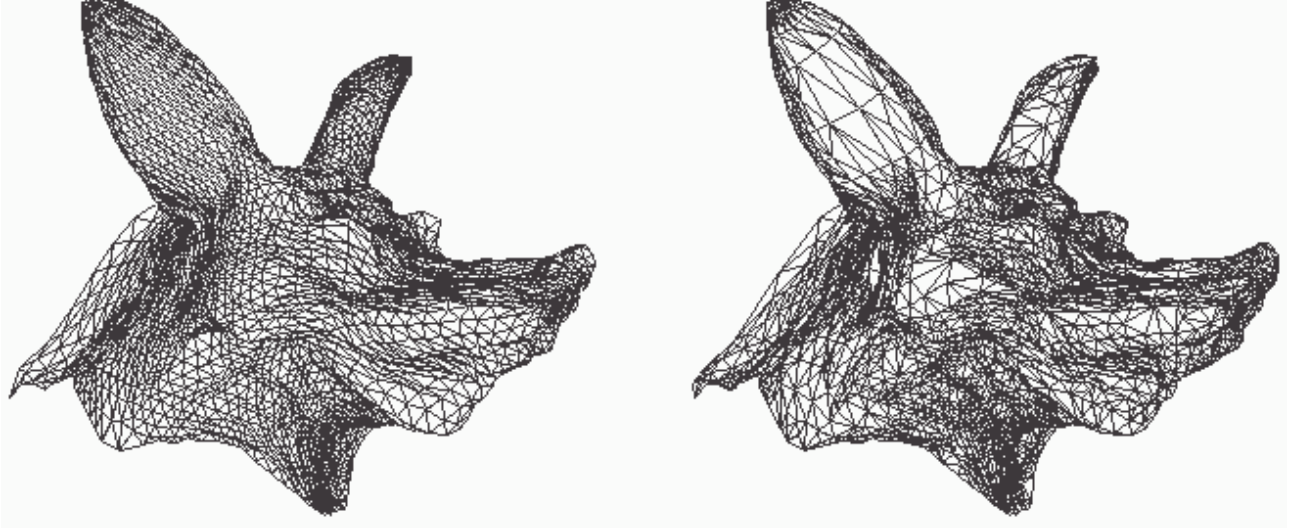
Ein weiterer wichtiger Punkt für weitere Forschung sind topologische Änderungen des Netzes. Gegenwärtig können solche topologischen Veränderungen wie das Ändern der Nachbarschaftsbeziehung, Löschen und Einfügen von Knoten nur auf der tiefsten Auflösungsstufe erfolgen. Benötigt werden nun Algorithmen, welche topologische Änderungen auf allen Levels zulassen.

Ebenso wichtig sind Techniken zur Generierung eines Netzes und seiner verschiedenen hierarchischen Levels aus einem gegebenen Datensatz eines Laser Scans oder Konvertierungsalgorithmen welche andere Datenstrukturen in dieselbe hierarchische Struktur überführen.

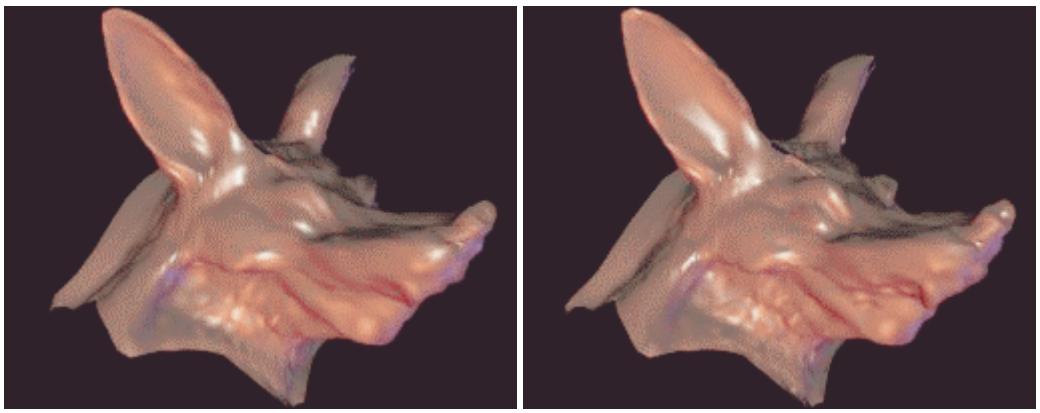
7. Anhang: Bilder



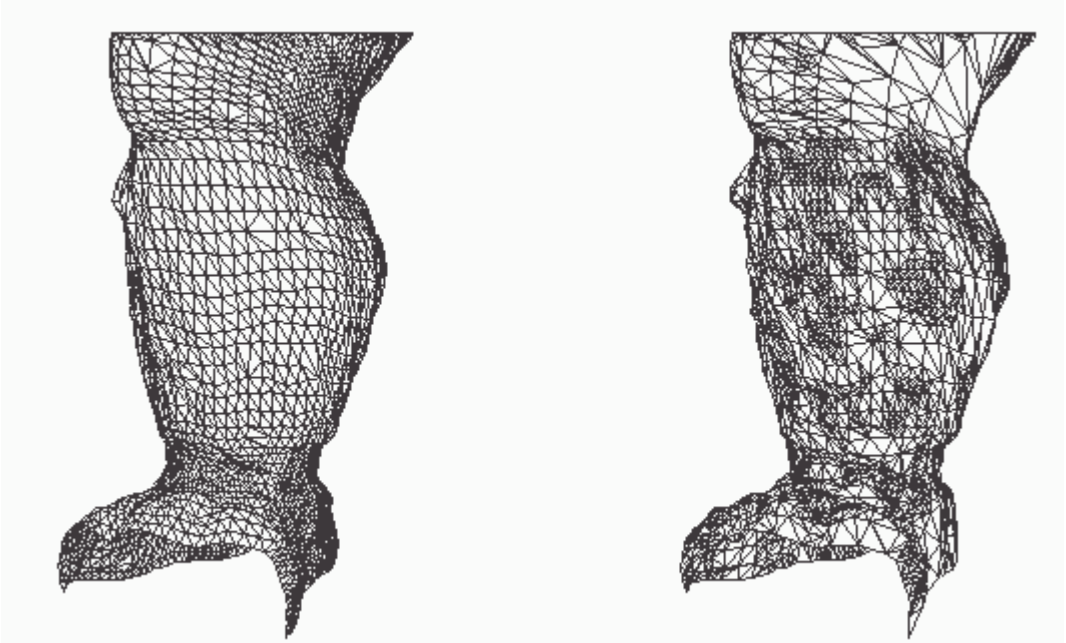
[Bild 1] Mit Hilfe dieses Gürteltieres wurde der Editor getestet. Links ist die ursprüngliche Oberfläche abgebildet, rechts eine editierte Version.



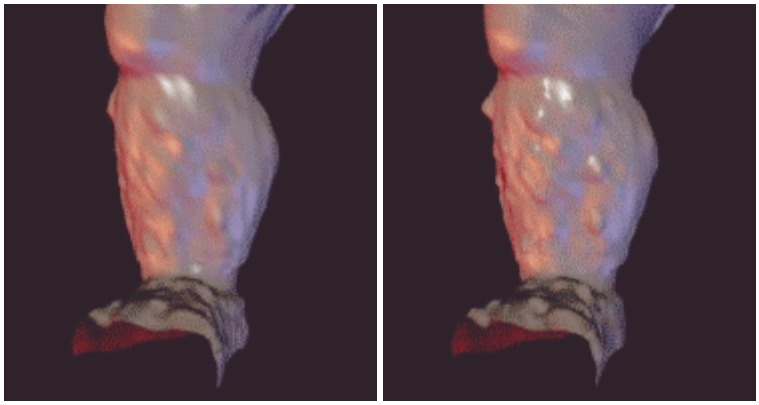
[Bild 2] Hier sieht man zwei verschiedene Wireframes des Kopfes. Die linke Oberfläche wurde uniform unterteilt, die rechte adaptiv. Approximativ enthalten die beiden Darstellungen gleich viele Dreiecke.



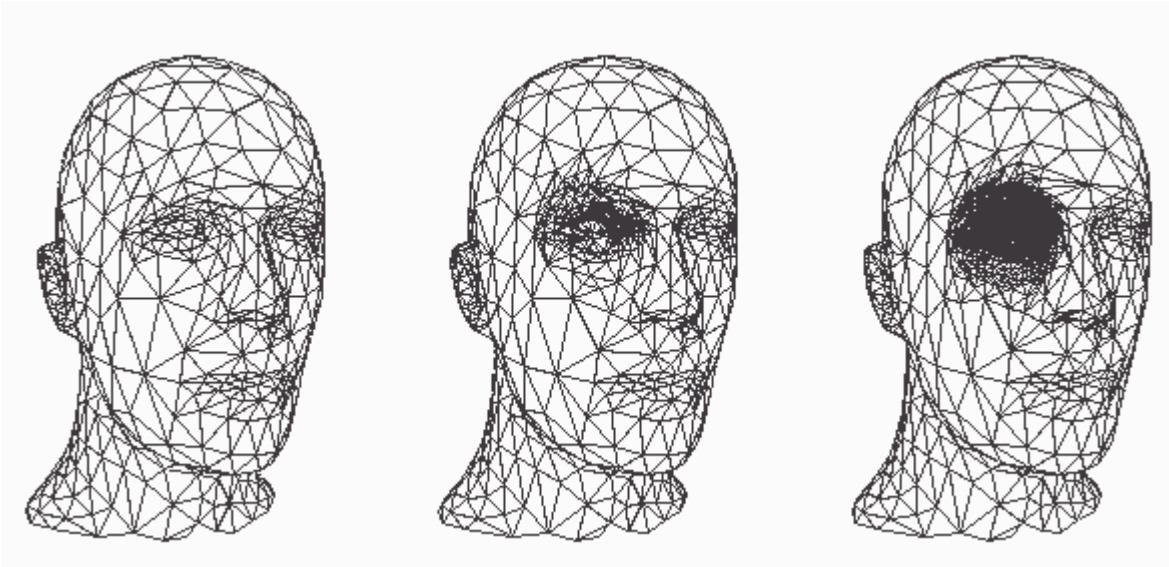
[Bild 3] Entsprechend dem Bild 2 ist hier die linke Abbildung aufgrund der uniformen Unterteilung gerendert worden, die rechte aufgrund der adaptiven.



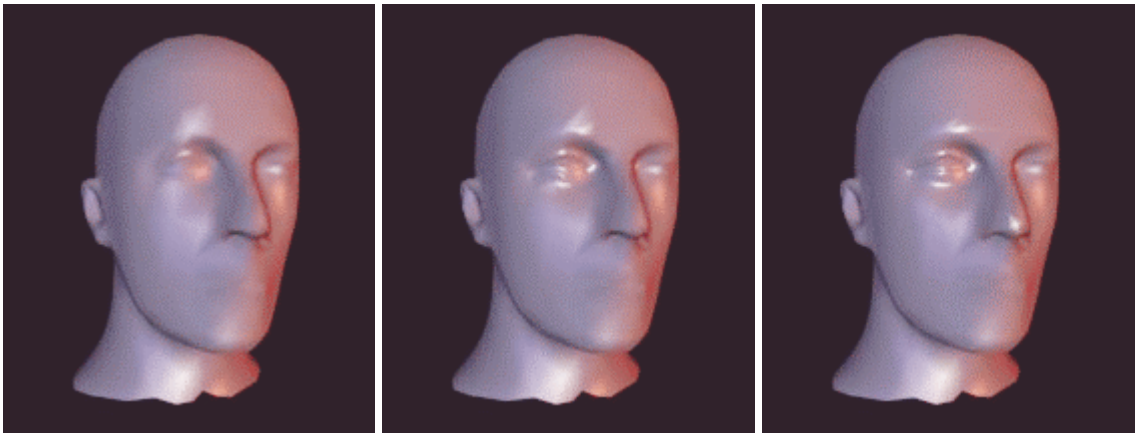
[Bild 4] Hier ist ein Beispiel, welches die Vorteile des Verfahrens noch einmal deutlich hervorhebt. Das adaptiv unterteilte Netz (rechts) zeigt viel mehr Details, als das uniform unterteilte (links).



[Bild 5] Auch hier sieht man die Auswirkungen auf die gerenderten Oberflächen deutlich.



[Bild 6] Dieses Beispiel zeigt nun noch den in Kapitel 5 und 6 angesprochenen Linseneffekt. Auf das rechte Auge des Mannequin Kopfes wurde quasi eine Linse gelegt. Dadurch verdichtet sich das Netz an dieser Stelle.



[Bild 7] In der gerenderten Oberfläche sieht man nun deutlich, was eine solche Linse bewirken kann. Man beachte jedoch, dass sich auch die Nasenspitze verändert. Das ist meiner Meinung nach eher dadurch zu erklären, dass die Netze vom Bild 6 nicht zum Bild 7 gehören.

Literaturverzeichnis

[1] DEROSE, T.D.
„Multiresolution Surfaces for Compression, Display, and Editing“
SIGGRAPH ‘96: Course Number 13: „Wavelets in Computer Graphics“, 1996

[2] LOOP, C.
„Smooth Spline Surfaces over Irregular Meshes“
Computer Graphics Proceedings, Annual Conference Series, 1994

[3] TAUBIN, G. A
„A Signal Processing Approach to Fair Surface Design“
Computer Graphics Proceedings, Annual Conference Series, 1995

[4] ZORIN, D., SCHRÖDER, P. und SWELDENS, W.
„Interactive Multiresolution Mesh Editing“
voraussichtlich SIGGRAPH ‘97, 1997