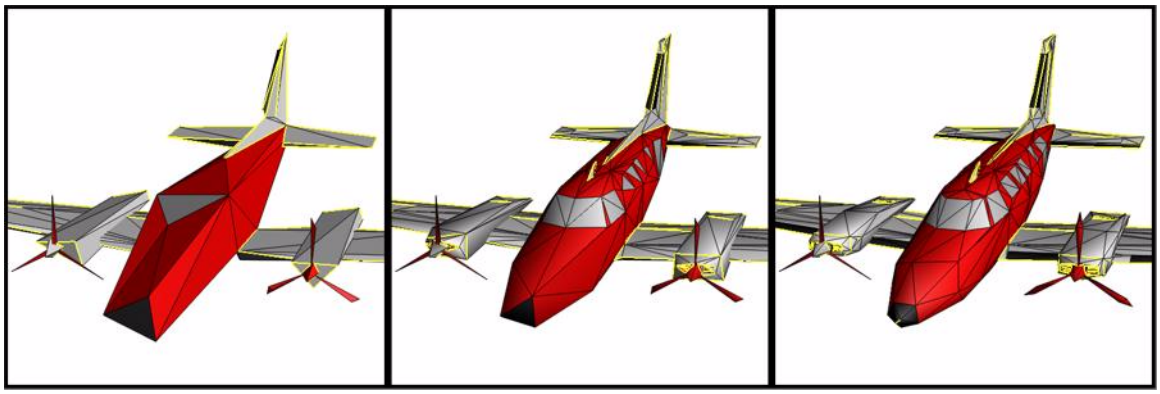
Fachseminar
Graphische Datenverarbeitung SS97
Prof.M.Gross

Progressive Meshes

Hugues Hoppe
Microsoft Research



Seminararbeit von
Martin Knecht

Paper: Progressive Meshes von Hugues Hoppe
SIGGRAPH'96, pp. 99-108

Inhaltsverzeichnis

Motivation

Zur Darstellung der Oberfläche eines geometrischen Modells werden oft Dreiecksvermaschungen verwendet. Diese Darstellung ist leider nicht ganz problemlos. Die Übertragungszeiten für genügend genaue Modelle werden zu lange oder der Speicherbedarf ist so gross, dass ein effizientes Arbeiten mit den Daten nicht mehr möglich ist.
Diese Probleme gaben Anlass, Forschung in diesem Gebiet zu betreiben und ein Resultat davon ist dieses Paper.

Meshes in der Computer Graphik

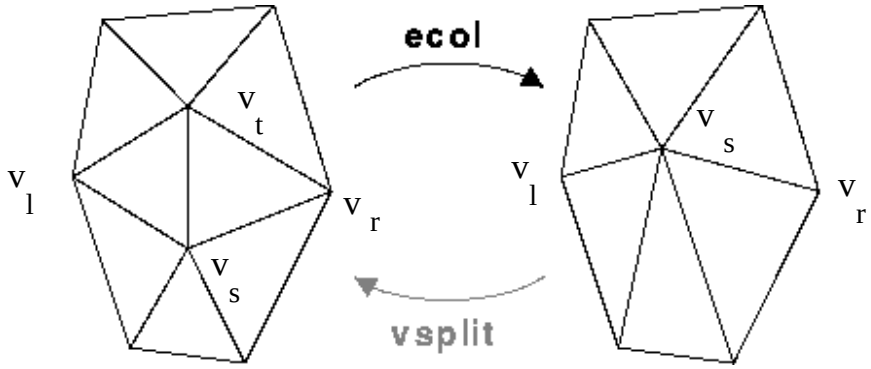
In der Computer Graphik wird bei Meshes im Allgemeinen klar unterschieden zwischen Konnektivität von Knoten und deren geometrischen Position. Das ermöglicht eine rasche Änderung von Koordinaten ohne die Primitive zu denen der Knoten gehört anpassen zu müssen.
Die Geometrie eines Meshes ***M*** wird formal als Tupel (K,V) repräsentiert. Dabei bedeutet **K** die Konnektivität der Mesh Simplizes, sprich den Knoten (vertices) [0-simplices $\{i\} \in K$] , Kanten (edges) [1-simplices $\{i,j\} \in K$] und Dreiecksflächen (faces) [2-simplices $\{i,j,k\} \in K$].
K besteht aus einer Menge von Knoten {1,…,m} mitsamt den Verbindungsinformationen der vertices, edges und faces.
$\mathbf{V}=\{v_1,\ldots,v_m\}$, $v_i \in \boldsymbol{R^3}$ ist die Menge, welche die Geometrie des Meshes in $\boldsymbol{R^3}$ definiert, indem sie zu jedem Knoten i dessen Koordinaten v_i speichert.
Zu einem Mesh gehört oft nicht nur die Geometrie, sondern auch andere Attribute, die sich auf Knoten und Dreiecksflächen oder auf *corners* (vertex-face-Tupel) beziehen. Solche Attribute können Farben, Normalen oder Texturkoordinaten sein.

Progressive Mesh Konstruktion

Das Problem wird hierbei in zwei Teilprobleme zerlegt.
Im ersten Teil (*Preprocessing*) wird das hochaufgelöste Orginalmesh $\hat{M}$ systematisch vereinfacht und in einer speziellen Datenstruktur, dem

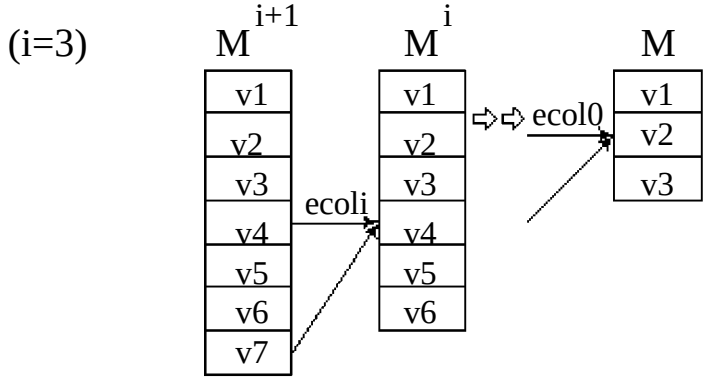
Progressive Mesh (PM), gespeichert. Die hierfür nötigen Berechnungen können sehr aufwendig sein.
Im zweiten Teil, der Rekonstruktion des vorberechneten PM, kann ohne grossen Rechenaufwand verlustfrei wieder die Orginalauflösung oder eine geringere Genauigkeitsstufe des Meshes konstruiert werden.

Um ein Mesh zu vereinfachen, genügt die sogenannte ***edge collapse*** Transformation *ecol(s,t)*. Sie vereinigt zwei benachbarte Knoten s und t mit den Koordinaten v_s und v_t in den neuen Knoten s mit den neuen Koordinaten v_s.
Der Knoten t und die beiden Dreiecksflächen {l,s,t} und {r,s,t} werden in diesem Schritt aus dem Mesh entfernt.



Ein initiales Mesh $\hat{M} = M^n$ mit m_0+n Knoten kann durch Anwendung einer Sequenz von n aufeinanderfolgenden *edge collapse* Transformationen in ein vereinfachtes Mesh M^0 mit m_0 Knoten umgewandelt werden.

$$(\hat{M} = M^n) \xrightarrow{ecol_{n-1}} \ldots \xrightarrow{ecol_1} M^1 \xrightarrow{ecol_0} M^0$$



Die Sequenz der edge collapse Transformationen muss sehr sorgfältig gewählt werden, da sie die Qualität der Näherungen M^i, i<n bestimmt. Dies geschieht einerseits durch statische Legalitätsbedingungen, die angeben, ob eine Kante entfernt werden darf oder nicht. Andererseits wird unter den Kanten mittels geeigneten Kostenfunktionen in jedem Schritt jene gewählt, welche beim collapse die kleinste Veränderungen im Mesh und in den Attributen bewirkt.
Für eine gute Approximation kommt noch ein weiterer Faktor hinzu, nämlich die Anpassung der Attribute in der Umgebung des collapses. Dieses bewirkt, dass die optische Erscheinung nicht wesentlich ändert.

Wichtig ist nun die Beobachtung, dass die edge collapse Transformation invertierbar ist. Die inverse Transformation wird als vertex split bezeichnet.
Der Ausdruck vsplit(s,l,r,t,A) fügt neben dem neuen Knoten vs, den Knoten v_t und zwei neue Dreiecke $\{v_s,v_t,v_l\}$ und $\{v_t,v_s,v_r\}$ ein. Die Attribute der Nachbarschaft der neuen Kante $\{v_s,v_t\}$ werden durch den Parameter A aktualisiert.

Weil die edge collapse Transformationen invertierbar sind, kann ein Beliebiges Mesh $\hat{M}$ als ein Mesh M^0 zusammen mit einer Sequenz von n vsplit-Records dargestellt werden

$$M^0 \xrightarrow{vsplit_0} M^1 \xrightarrow{vsplit_1} \ldots \xrightarrow{vsplit_{n-1}} (\hat{M} = M^n)$$

$(M^0,\{vsplit_0,\ldots,vsplit_{n-1}\})$ wird als Progressive Mesh Repräsentation , oder kurz als Progressive Mesh (PM), von $\hat{M}$ bezeichnet.

Ziel der Mesh Optimierung ist ein Mesh M=(K,V) zu generieren, das einerseits möglichst genau dem Orginalmesh $\hat{M}$ entspricht, gleichzeitig aber eine wesentlich kleinere Anzahl an Kanten und Dreiecksflächen enthält.
Dieses Problem wird als Minimierung einer Energie- respektive Kostenfunktion angegangen.

Zuerst wird auf das Orginalmesh $\hat{M}$ eine Menge X von Referenzpunkten $x_i \in \boldsymbol{R}^3$ abgetastet, welche im Allgemeinen die Knoten in $\hat{M}$ sind, aber nach Bedarf auch weitere Punkte von $\hat{M}$ dazugenommen werden können.

Die Energiefunktion eines Meshes M

$$E(M)= E_{dist}(M) + E_{spring}(M) + E_{scalar}(M) + E_{disc}(M)$$

setzt sich zusammen aus vier Termen, einer Distanzenergie E_{dist}, einer Federenergie E_{spring}, einer Skalarenergie E_{scalar} und einer Energie E_{disc}, welche Unstetigkeiten erhalten soll.
Die Distanzenergie

$$E_{dist}(M)= \sum_i d^2(x_i, p_i) \quad \text{mit} \quad d^2(x_i, p_i) = \min_{p_i \in M} \left\| x_i - p_i \right\|^2$$

wobei p_i die Projektionen der x_i auf M mit minimalem Abstand sind.
Die Federenergie

$$E_{spring}(M)= \sum_{\{j,k\} \in K} \kappa \left\| v_j - v_k \right\|^2$$

entspricht der Plazierung einer Feder mit Ruhelänge Null und Federkonstante κ auf jeder Kante des Meshes.
Die Skalarenergie

$$E_{scalar}(M)=(c_{scalar})^2 \sum_i \left\| x_i - p_i \right\|^2$$

misst den Fehler von Skalarwerten (z.B. Texturkoordinaten oder Farben)

auf M im Vergleich zu $\hat{M}$. Hierfür werden die Skalarwerte $\underline{x}_i$ und $\underline{p}_i$ an den Koordinaten x_i und p_i berechnet und verglichen.
Verschiedene Attribute auf einem Mesh stellen wesentliche Eigenschaften für das Auge dar, die erhalten werden müssen, wie z.B. Unstetigkeits-kurven. Um Unstetigkeitskurven zu erhalten, wird

$$E_{disc}(M)= \sum_i d^2(\tilde{x}_i, \tilde{p}_i)$$

ähnlich wie E_{dist} definiert. In $\hat{M}$ wird eine Menge X_{disc} von Punkten $\tilde{x}_i$ auf den Unstetigkeitskurven bestimmt. Diese werden auf das Mesh M auf die Punkte $\tilde{p}_i$ mit kleinstem Abstand projiziert. Eingebettet in das surface fitting Problem muss also ein curve fitting Problem gelöst werden, um die Punkte $\tilde{p}_i$ zu bestimmen.

Die Gesamtkosten eines collapses entsprechen der Differenz der Energien

$$\Delta E=E^{i-1}- E^i$$

des Meshes nach der Transformation minus der Energie des Meshes davor. Das Ziel in jedem Schritt ist eine möglichst kleine Veränderung($\Delta E \cong 0$). Das Optimum würde erreicht, wenn jede einzelne Teilenergie minimal wäre. Da aber die Grössenordnung der Teilenergien sehr verschieden sein kann, muss der Einfluss gewichtet werden. Eine bestimmte Energie darf nicht wegen ihrer Grössenordnung bereits bei, relativ zu ihrer eigenen Grösse, kleinen Veränderungen andere Energieterme vollständig dominieren, und diese somit abwerten.

Ein möglicher Algorithmus kann nun wie folgt aussehen:
Zuerst wird eine Priority-Queue mit allen legalen Kanten eines Meshes aufgebaut, wobei das ΔE das Ordnungskriterium ist.
Anschliessend wird solange iteriert, wie collapse verlangt werden und die Queue nicht leer ist. Dabei wird in jedem Schritt die vorderste Kante aus der Queue entfernt, ihr collapse durchgeführt und das inverse split-Element erzeugt.
Nach der Durchführung des collapses ist es erforderlich, dass die Legalität aller Kanten in der Nachbarschaft neu überprüft wird. Für jede legale Kante der Nachbarschaft muss ΔE neu berechnet und in die Queue eingefügt werden.
Nach der Iteration muss das resultierende Mesh M^0 zusammen mit den vsplits in ein PM umgewandelt werden. Die Reihenfolge der vsplits muss so umgewandelt werden, dass die zuerst erzeugten vsplits bei der Rekonstruktion als letzte ausgeführt werden. Dies hat zur Folge, dass bei der Rekonstruktion am Anfang die Veränderungen am grössten sind und gegen den Schluss nur noch kleine Änderungen dazukommen.
Weiter müssen die Knoten neu numeriert werden, damit in jedem vsplit der neue Knoten lückenlos als nächster in die Liste der bestehenden Knoten eingefügt werden kann. Die neuen Knoten-Indizes müssen auch in den Dreiecken angepasst werden.

Anwendungen

Das PM-Verfahren unterstützt verschiedene Anwendungsmöglichkeiten. Ist der aufwendige Teil, die Konstruktion des PM erledigt, kann ein Mesh in beliebiger Auflösung M^i, $0 \leq i \leq n$ sehr effizient aufgebaut werden, da keine Berechnungen mehr nötig sind.

- **Mesh simplification:**
  Ein Mesh M^i, i<<n ist wesentlich effizienter zu handhaben als das Original Mesh $\hat{M}$. Darum ist es wünschenswert, solche Verfahren wie das PM einzusetzen.

- **Level-of-detail (LOD) approximation:**
  Das Verfahren unterstützt sehr effizient die Erstellung von verschiedenen LOD-Versionen des Meshes. Damit kann eine hohe Auflösung gewählt werden, wenn ein Objekt nahe dem Betrachter ist und eine tiefere Auflösung wenn es weiter weg ist.

- **Geomorphs:**
  Dadurch, dass für jeden Knoten klar ist, von welchem Knoten er abstammt, kann ein Geomorphing zwischen zwei beliebigen Auflösungsstufen mittels eines α-Blendings realisiert werden. Dafür werden die Knoten des Meshes M^j, die im Mesh M^i mit i<j fehlen für α=0 bei ihren Vorfahren plaziert und wandern dann linear bis α=1 zu ihren Zielkoordinaten in M^j.
  In Animationen können so störende popup-Effekte vermieden werden.

- **Progressive transmission:**
  Bei der Übertragung eines Meshes über einen Kommunikationskanal kann beim Empfänger das Bild progressiv verfeinert werden. Der Empfänger kann somit entscheiden, wann für ihn das Objekt eine genügend hohe Auflösung besitzt und kann somit die Übertragung frühzeitig stoppen.

- **Mesh compression:**
  Das Problem, den Speicher für ein Modell zu minimieren kann auf zwei unabhängigen Wegen angegangen werden. Einerseits kann durch Mesh simplification die Anzahl der Knoten und Dreiecke minimiert werden und andererseits den Speicherplatz für ein bestimmtes Mesh durch Ausnutzung von Abhängigkeiten zu minimieren.

- **Selective refinement:**
  Jede LOD-Repräsentation stellt ein Mesh unabhängig vom Beobachterstandpunkt in einer bestimmten Auflösung dar. Manchmal ist es aber auch wünschenswert, die Auflösung

abhängig von einer Region zu variieren. (z.B. bei einer Flugsimulation: das Landschafts-Mesh muss nur eine hohe Auflösung besitzen, wenn es aus dem Cockpit zu sehen ist und sonst nicht).

Vorteile-Nachteile-Ausblick

- **Vorteile:**
  Kein Verlust von Informationen durch edge collapse Transformation
  ($\hat{M} = M^n$**).**
  Man bekommt früh (d.h. schnell) Informationen, mit denen man schon arbeiten kann.
  Der Nutzen für alle Anwendungen die ich erwähnt habe.
  Ein grosser Vorteil ist auch, dass Unstetigkeitskurven bei den collapsen sehr gut erhalten bleiben.

- **Nachteile:**
  Ein grosser Nachteil ist der grosse Aufwand (Zeit) für das Preprocessing.
  Das PM-Editing ist zur Zeit noch nicht möglich.
  Bei einem Mesh mit geringerer Auflösung als das Orginalmesh kann man keine Prozentzahl angeben, wie wieviel es vom Orginalmesh noch entfernt ist.

- **Ausblick:**
  -Experimentierung mit PM-Editing
  -Repräsentation von gesprochenen oder animierten Modellen
  -Progressive Repräsentationen von allgemeineren Formen
  -Hinzunahme von räumlichen Datenstrukturen, um das selective refinement effizienter zu machen