View-Dependent Simplification

in

Computer Graphics

Metovic Sasa - Mustafa Fettahoglu

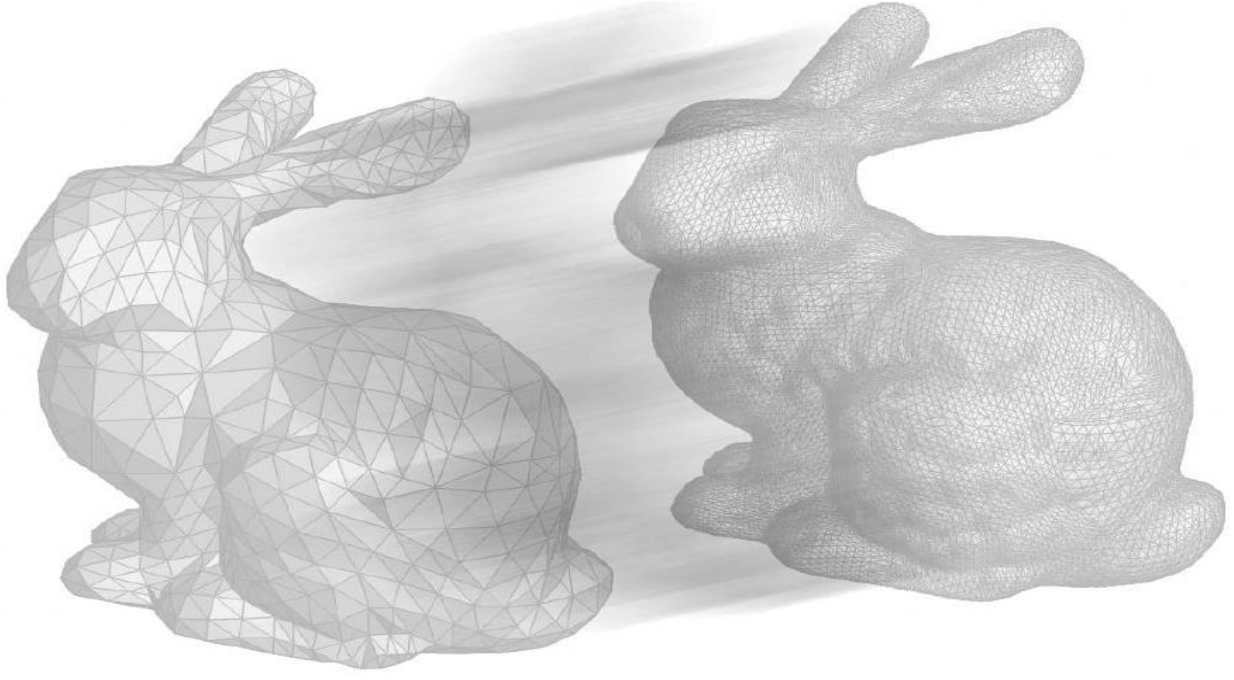
Salzburg, am 30.01.2003

INHALTSVERZEICHNIS

EINFÜHRUNG

Die Erstellung von hochdetaillierten 3D Modellen ist heutzutage keine große Herausforderung mehr, weshalb sie beginnen einen festen Platz im Feld der angewandten Computergrafik einzunehmen, da sie eine realistischere Darstellung ermöglichen. Eine sehr handhabbare, portable und weit verbreitete Darstellungsform für solche 3D Modelle ist eine gewaltige Liste von Dreiecken, welche auch als Dreiecks-Mesh bezeichnet wird. Obwohl der technologische Fortschritt im Bereich der Prozessoren und Grafik-Hardware in den letzten Jahren enorm war, stellt ein hoch detailliertes 3D Modell immer noch eine Herausforderung in den unterschiedlichsten Bereichen dar. Insbesondere soll hier auf die Problematik in den folgenden Bereichen hingewiesen werden:

- Interaktive Darstellung
- Übertragung von 3D Modellen bei begrenzter Bandbreite (z.B. Streaming im Internet)
- Effiziente Speicherung von 3D Modellen



ANSICHT ABHÄNGIGE VEREINFACHUNG

View-Dependent Simplification (VDS): Diese Art der Simplification versucht die Position des Betrachters bei der Darstellung zu berücksichtigen. Somit kann das Modell selektiv verfeinert werden, z.B. an der Silhouette oder auf der zugewandten Seite des Modells.

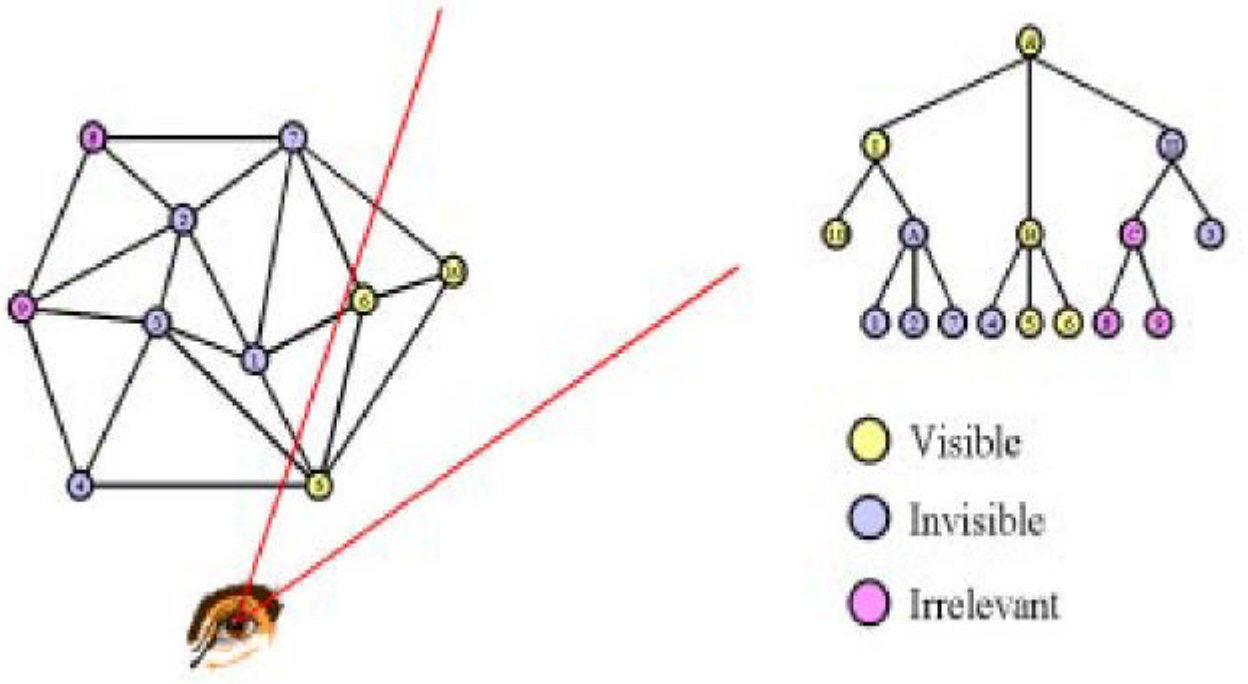


Bild1: Sichtbare, unsichtbare und irrelevante Knoten.

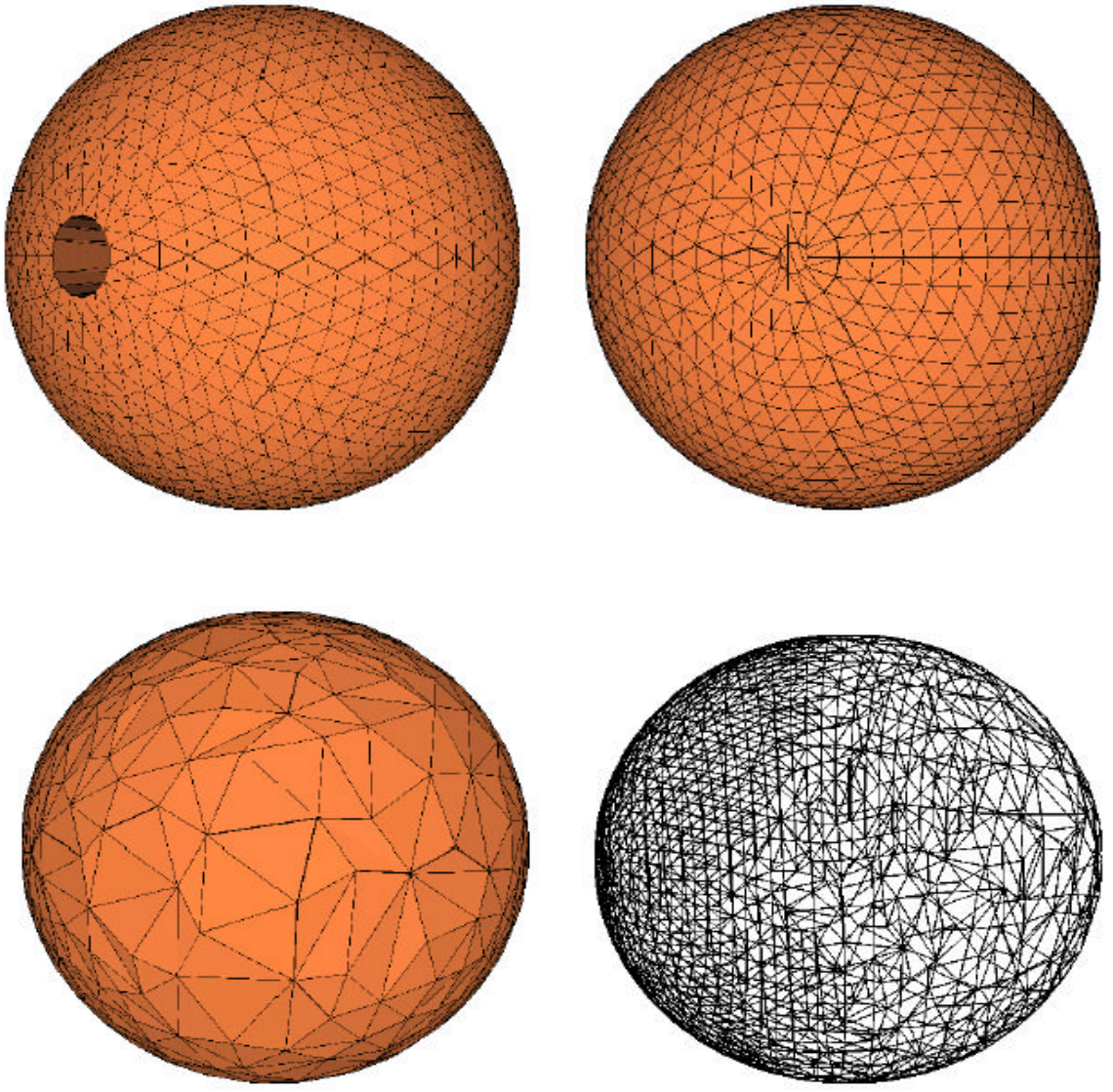


Bild2: Orginal – Vereinfachte Ansicht von vorne, hinten und von der Seite.

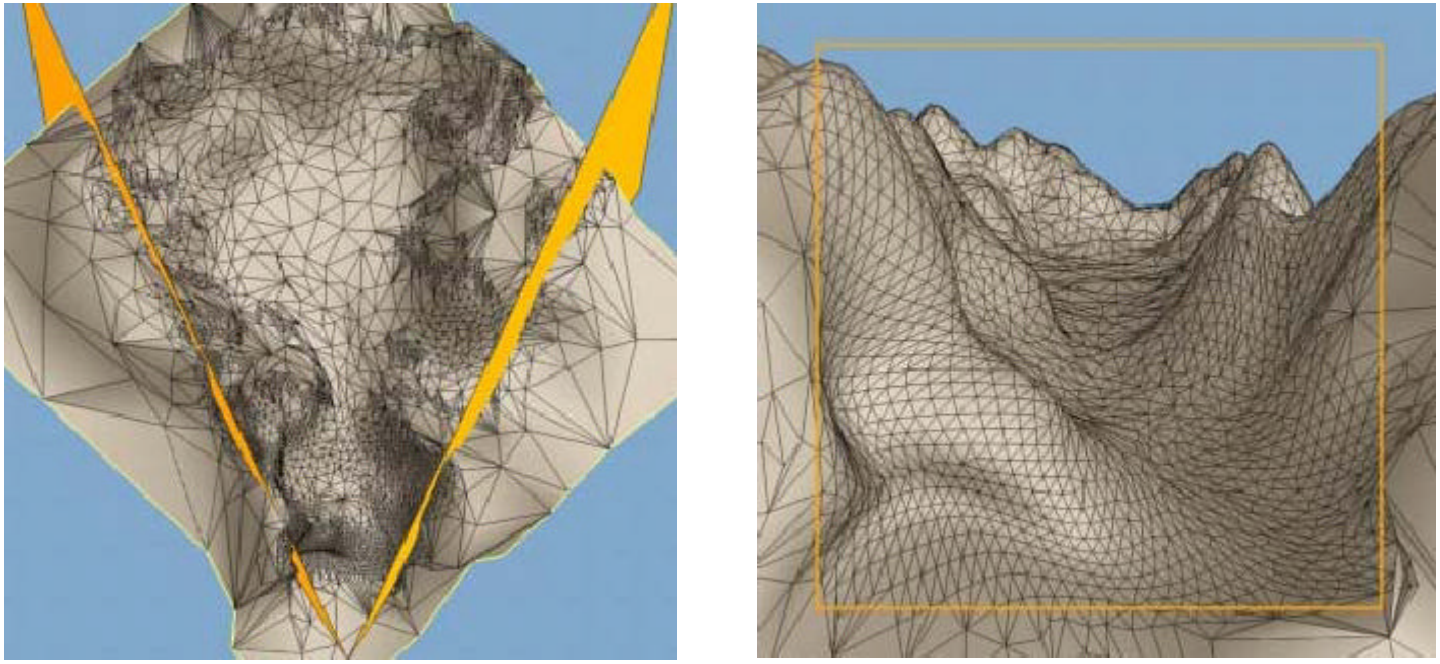


Bild3: Beispiel von VDS in der Topologie.

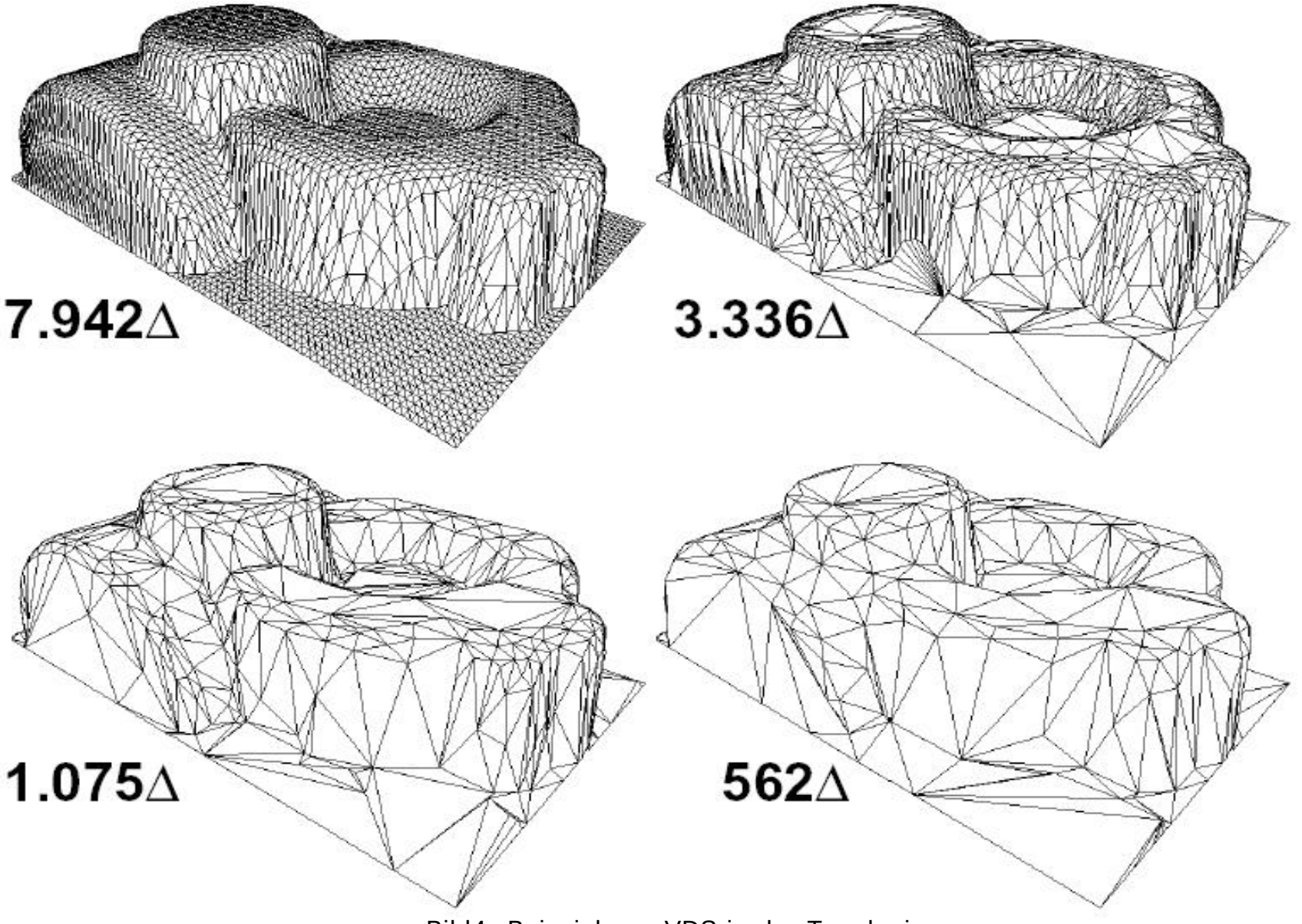


Bild4: Beispiel von VDS in der Topologie.

AUFBAU EINES MESHES

Ein Mesh besteht grob gesagt aus einer Menge von Punkten (engl. Vertices) und einer Beschreibung wie diese Punkte untereinander verbunden sind. Diese Beschreibung kann zum Beispiel eine einfach Liste von Kanten sein, welche Punktpaare untereinander verbindet. Die Darstellung eines solchen Meshes würde ein Drahtgittermodell erzeugen. Die übliche Beschreibung der Konnektivität einzelner Punkte ist jedoch ein Dreieck, welches immer drei Punkte untereinander verbindet. Generell lässt sich also ein einfaches Mesh wie folgt definieren [Hop96]

Definition: (Einfaches Mesh) Gegeben eine Menge von Punkten V und eine Beschreibung der Konnektivität K. Dann sei ein Mesh M definiert durch das Tupel (K, V).

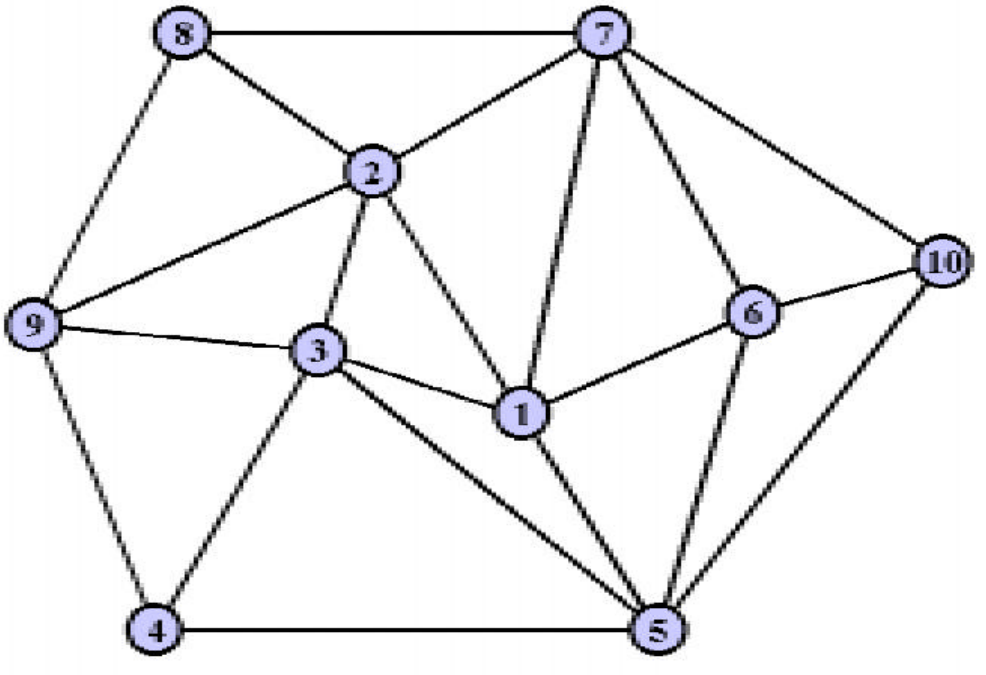


Bild5: Ein einfaches Mesh.

EDGE COLLAPSE UND VERTEX SPLIT

Die Operation Edge Collapse. Hierbei wird, wie der Name schon besagt, eine Kante des initialen Meshes kollabiert. Dies hat zur Folge, dass alle Dreiecke des Meshes, welche diese Kante nutzen auch aus dem Mesh entfernt werden, da sie durch die Entfernung der Kante degenerieren.

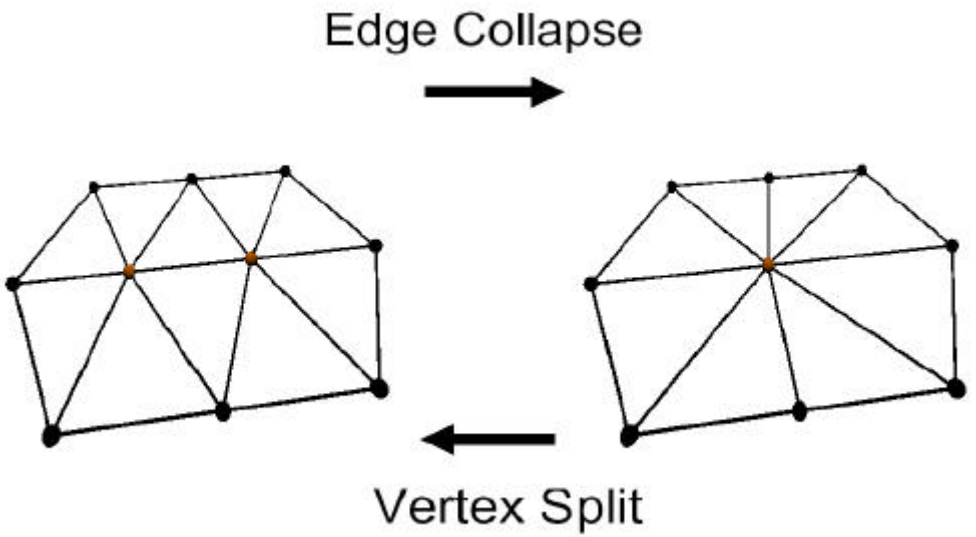


Abb. 1: Die Edge Collapse und Vertex Split Operationen.

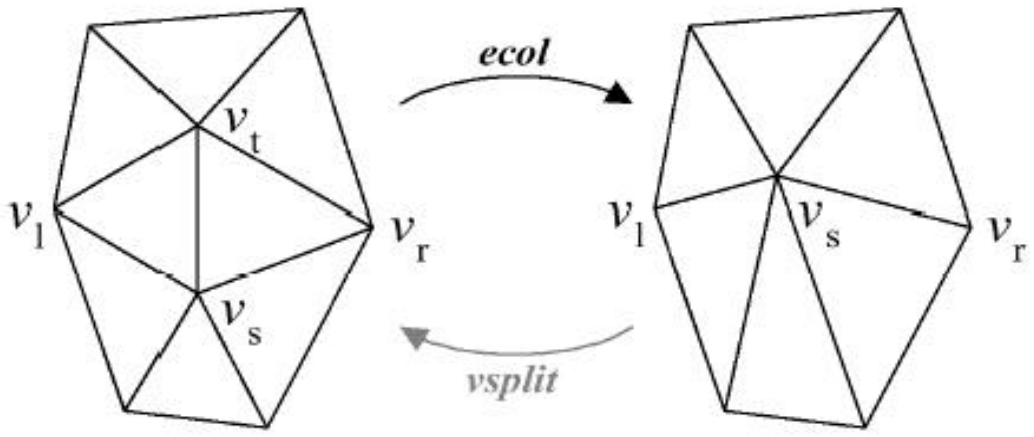


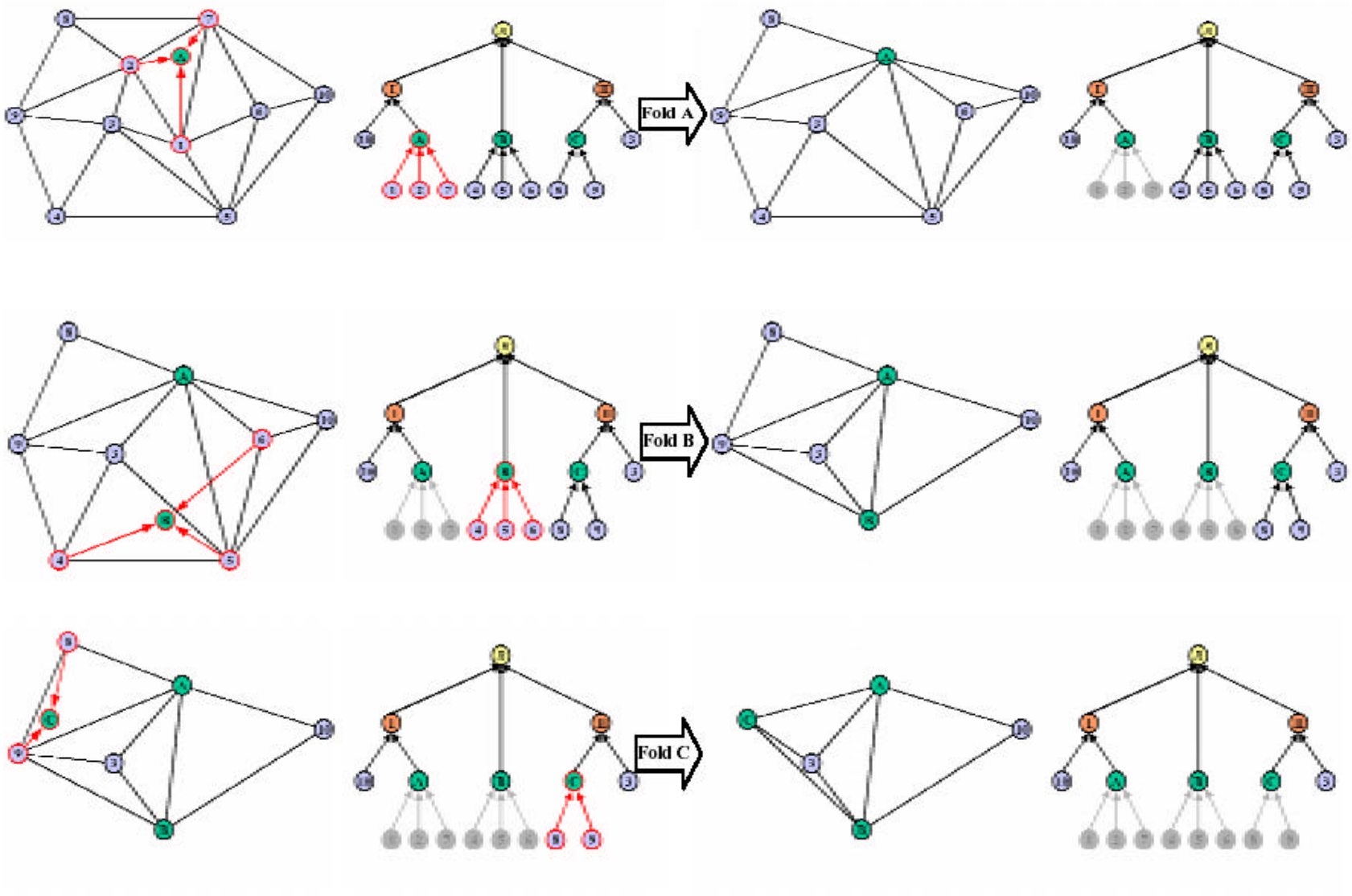
Abb. 2: Die Edge Collapse und Vertex Split Operationen.

Die Funktionsweise ist am leichtesten anhand von Abbildung 2 zu erklären. Hier wird auf die Kante {vs, vt} die ecol-Operation angewandt. Dies hat zur Folge, dass die Punkte *vs* und *vt* zu einem neuen Punkt vs verschmolzen werden, und dass hierbei auch die Dreiecke {vt, vs, vl} und {vs, vt, vr} aus dem Mesh entfernt werden. Man kann diesen Prozess auch so verstehen, dass der Punkt *vt* entfernt wird und die inzidenten Kanten entsprechend an *vs* gehangen werden (doppelte Kanten müssen natürlich eliminiert werden). Die neue Position von *vs* ist beliebig, sollte jedoch so gewählt werden, dass das neue Mesh optimal die Form das initialen Meshes wiedergibt.

Durch die Edge Collapse Operation ist es also möglich Dreiecke aus einem Mesh zu entfernen. Durch Mehrfachanwendung dieser Operation wird somit eine Sequenz von Meshes mit immer weniger Dreiecken erzeugt. Dies drückt Hoppe durch folgende Notation aus. Sei M das initiale Mesh und n die Anzahl der Kanten in diesem Mesh. Somit ergibt sich für eine Sequenz von Edge Collapses:

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

Leider kann man durch diese Operation nur Mesh Vereinfachung erreichen, was aber unseren Anforderungen nicht genügt. Zum Glück hat der Edge Collapse auch eine Umkehrfunktion, nämlich den Vertex Split.


Fold A
Fold B
Fold C

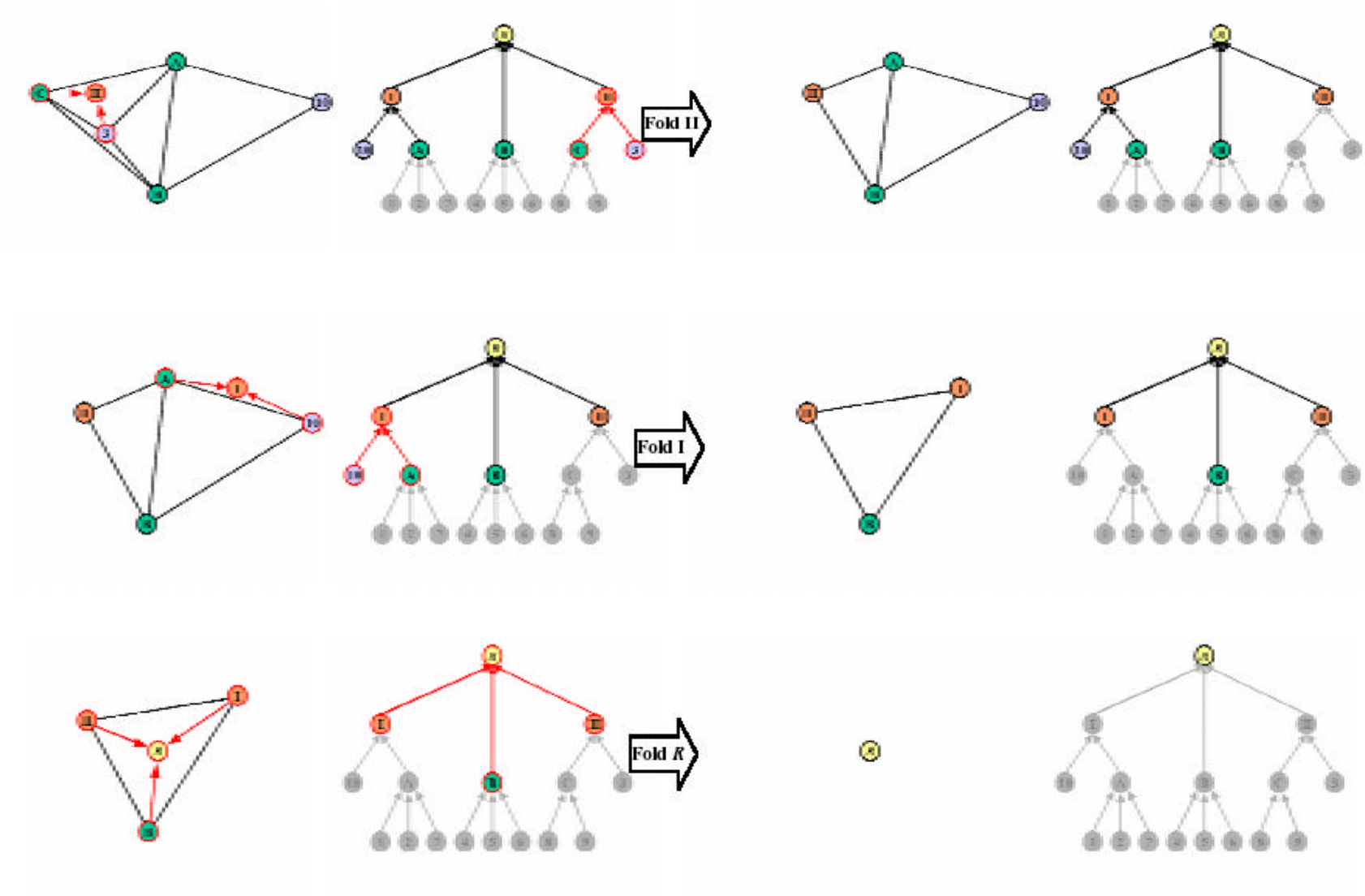
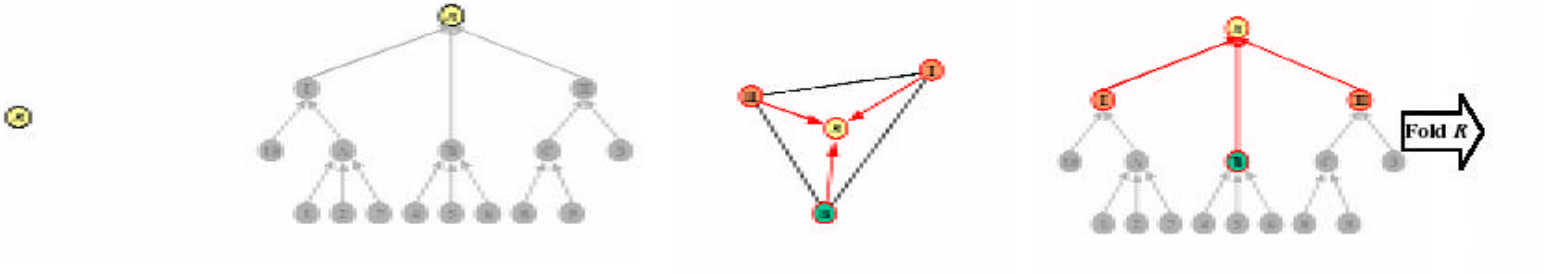


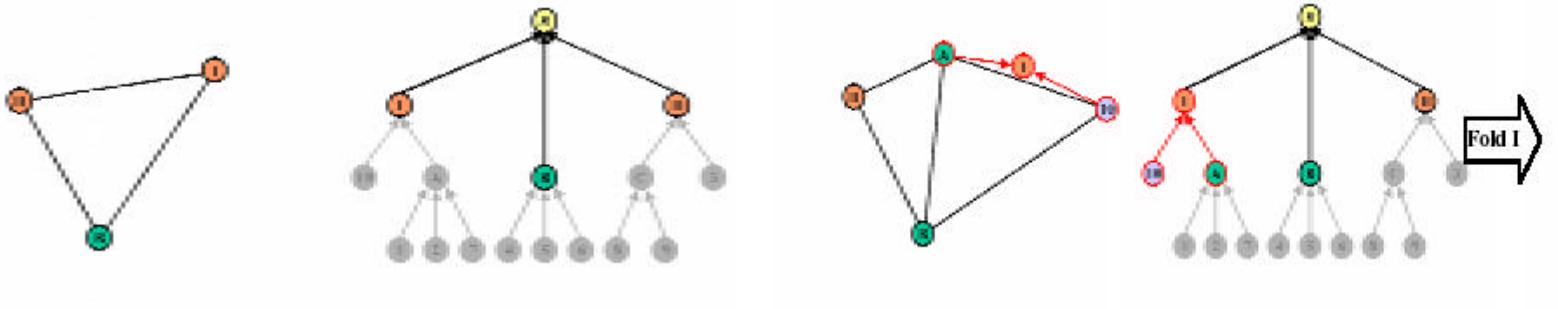
Bild6: Edge Collapse.

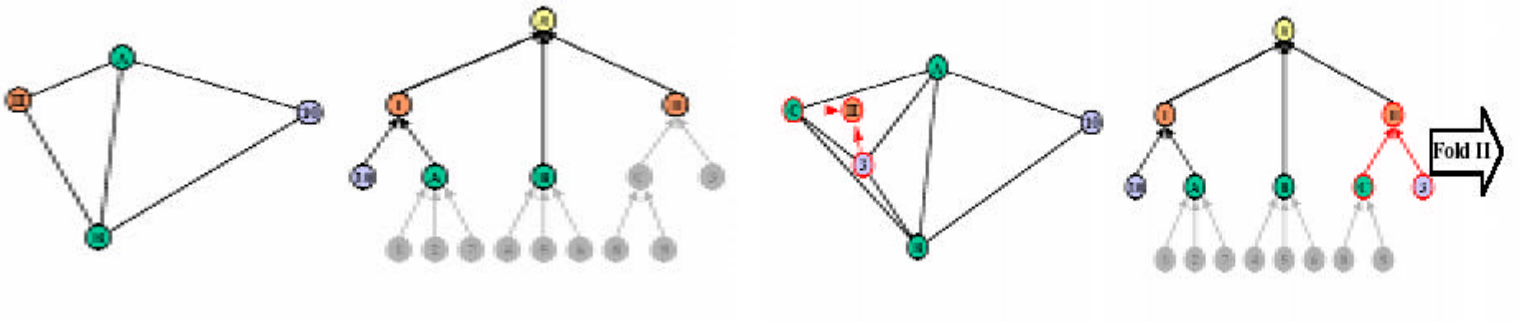
Der Vertex Split Operator vsplit kriegt als Eingabe einen Punkt *vs* des Meshes und kehrt die vorgenommene Transformation des Edge Collapses um. Es werden also ein Punkt *vt* erzeugt, die Position von *vs* aktualisiert, die Kanten umgeordnet sodass die Dreiecke {vt, vs, vl} und {vs, vt, vr} wieder im Mesh vorhanden sind und alle Attribute aktualisiert Mit Hilfe der Vertex Split Operation ist es möglich ein sehr einfaches initiales Mesh schrittweise zu verfeinern. Dadurch wird folgende Mesh-Sequenz ermöglicht:

$$M^0 \rightarrow_{vsplit_0} M^1 \rightarrow_{vsplit_0} \ldots \rightarrow_{vsplit_{n-1}} \left(\hat{M} = M^n\right)$$

Durch die beiden Operationen Edge Collapse und Vertex Split kann man somit die Kette aller Detailstufen des Meshes navigieren. Mit dem Edge Collapse kann man dabei Detail entfernen und mit dem Vertex Split kann man Detail dazugeben.


Fold K


Fold I


Fold II

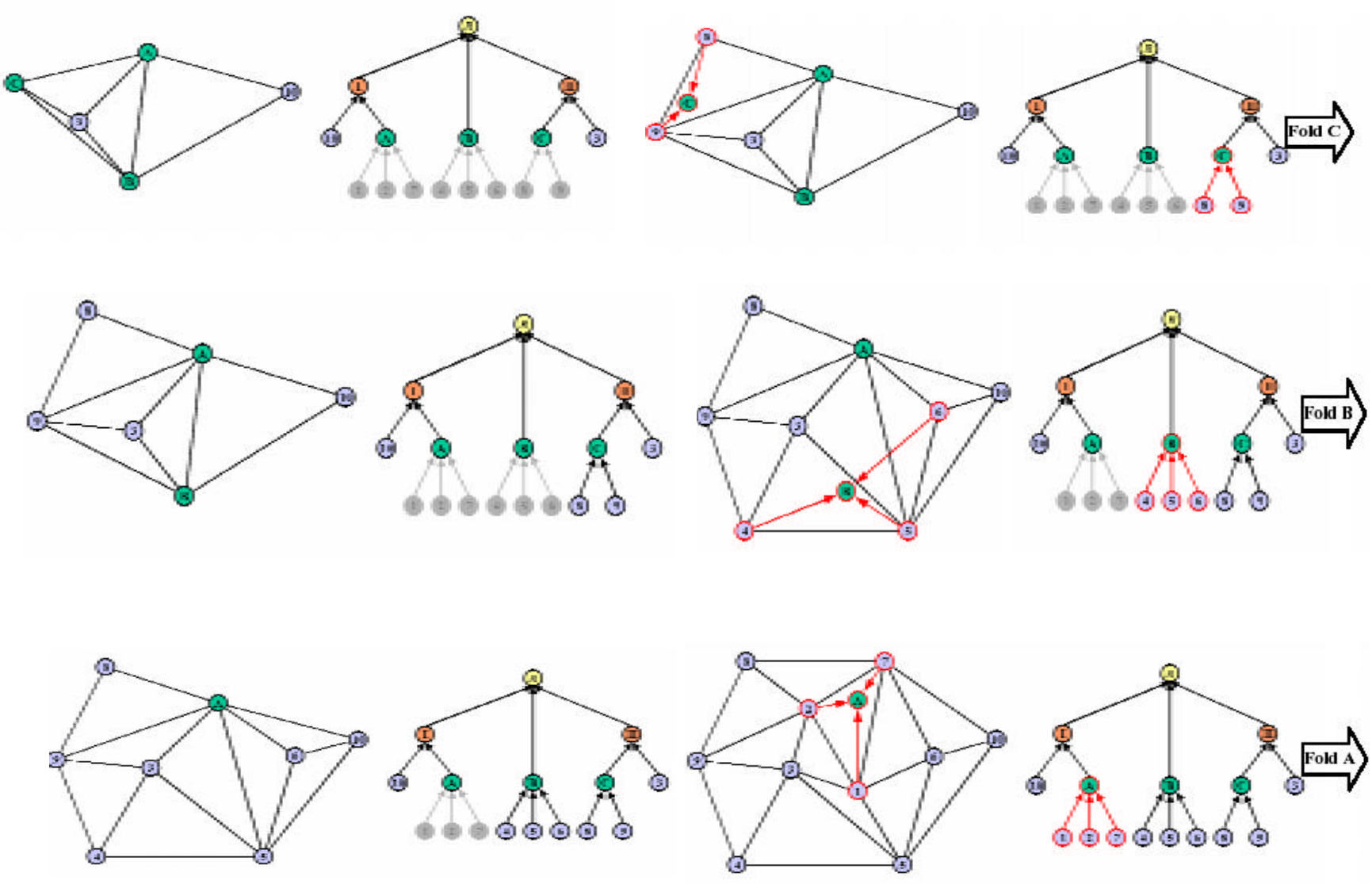


Bild7: Vertex Split.

LITERATUR - QUELLVERZEICHNIS

[Hop96] Hugues Hoppe. Progressive meshes. Computer Graphics, 30(Annual Conference Series):99–108, 1996.

[Hop97] Hugues Hoppe. View-dependent refinement of progressive meshes. Computer Graphics, 31(Annual Conference Series):189–198, 1997.

BILD[1]: David P. Luebke. Robust VDS Very Large-Scale CAD Visualization.
BILD[2]: Graphische Datenverarbeitung. Universität Erlangen – Nürnberg.
BILD[3]: David P. Luebke. Generalizied VDS.
BILD[4]: Jens Heydekorn. Simplification.
BILD[5]: Jihad El-sana / Amitabh Varshey. VDS.
BILD[6-7]: David P. Luebke. VDS.
ABBILDUNG[1-2]: Hugues Hoppe. Computer Graphics.

http://vdslib.virginia.edu