EUROGRAPHICS 2002

Tutorial TH3: View-Dependent Rendering for Polygonal Datasets

J. El-Sana, Ben Gurion University, Israel
L. De Floriani, University of Genova
E. Puppo, University of Genova
A. Shamir, The Interdisciplinary Center, Herzelia

Published by
The Eurographics Association
ISSN 1017-4565

The European Association for Computer Graphics
23rd Annual Conference

EUROGRAPHICS 2002

Saarbrücken, Germany
September 2–6, 2002

EUROGRAPHICS
THE EUROPEAN ASSOCIATION
FOR COMPUTER GRAPHICS

Organized by

Max-Planck-Institut
für Informatik
Saarbrücken, Germany

Universität des Saarlandes
Germany

International Programme Committee Chairs
George Drettakis (France)
Hans-Peter Seidel (Germany)

Conference Co-Chairs
Frits Post (The Netherlands)
Dietmar Saupe (Germany)

Honorary Conference Co-Chairs
Jose Encarnação (Germany)
Wolfgang Straßer (Germany)

Tutorial Chairs
Sabine Coquillart (France)
Heinrich Müller (Germany)

STAR Report Chairs
Dieter Fellner (Germany)
Roberto Scopignio (Italy)

Lab Presentation Chairs
Günther Greiner (Germany)
Werner Purgathofer (Austria)

Industrial Seminar Chairs
Thomas Ertl (Germany)
Bernd Kehrer (Germany)

Günter Enderle Award Committee Chair
François Sillion (France)

Conference Game Chair
Nigel W. John (UK)

John Lansdown Award Chair
Huw Jones (UK)

Conference Director
Christoph Storb (Germany)

Short/Poster Presentation Chairs
Isabel Navazo (Spain)
Philipp Slusallek (Germany)

Local Organization
Annette Scheel (Germany)
Hartmut Schirmacher (Germany)

Eurographics 2002

View-Dependent Rendering for Polygonal Datasets

Jihad El- Sana (Ben- Gurion University of the Negev),
Leila De Floriani (University of Genova),
Enrico Puppo (University of Genova), and
Arik Shamir(The Interdisciplinary Center)

View-Dependent Rendering for Polygonal Datasets

Jihad El-Sana (Ben-Gurion University of the Negev), Leila De Floriani (University of Genova), Enrico Puppo (University of Genova), Arik Shamir(The Interdisciplinary Center)

Proposed Length: Half-day.

Synopsis:

Recent advances in three-dimensional shape acquisition, simulation, and design technologies often lead to the generation of datasets that are beyond the interactive rendering capabilities of the current graphics hardware. In order to bridge the increasing gap between hardware capabilities and graphics dataset size, methods are employed to reduce the complexity of the geometric datasets while keeping their visual appearance. Multi-resolution hierarchies and view-dependent rendering have been introduced to enable various levels of detail to seamlessly co-exist over different regions of the same surface. These levels of detail depend on parameters such as view location, illumination, and speed of motion and are determined per-frame.

This tutorial will focus on describing techniques to construct multi-resolution hierarchies for geometric objects, and utilizing these hierarchies to accelerate rendering and transmission of large polygonal datasets. It is intended for those who have an understanding of the basics of 3D graphics and analysis of algorithms. The goal of the tutorial course is to expose both students and professionals to various advanced techniques and algorithms to accelerate the rendering of large polygonal datasets through the use of multiresolution hierarchies.

The proposed tutorial shall start by presenting level of details and multi-resolution hierarchies in the general framework, then it covers the recently developed view-dependent rendering approaches which include multi-triangulation, generalized view-dependent rendering, and multi-resolution hierarchies for dynamic objects and scenes. Lastly, it will introduce advance techniques such as view-dependent rendering over network and integrating view-dependent rendering with occlusion culling.

Since the tutorial presents several view-dependent rendering algorithms, it will cover in depth first the construction of the data-structures which are usually created off-line, and second, the real-time navigation which utilizes these structures online to extract appropriate level of details for rendering based on view-parameters and illumination.

Prerequisites: Understanding of the basics of 3D graphics and scientific visualization techniques.

Tutorial Syllabus

- Introduction to the course (El-Sana) *5 minutes*
- The Multi-Triangulation: A Framework for Multiresolution Triangle Meshes (De Floriani)
- Generalized View-Dependent Rendering (El-Sana)
- Dynamic Multi-Resolution Hierarchies (Shamir)
- Client/Server View-Dependent Rendering (Puppo)
- Integrating View-Dependent Rendering with Occlusion Culling (El-Sana)

Presenters' Biographies
Jihad El-Sana

Department of Computer Science
Ben Gurion University of The Negeve
Beer-Sheva, 84105
Israel
E-mail: el-sana@cs.bgu.ac.il

Jihad El-Sana is a Lecturer (equivalent to assistant professor in the U.S.) of Computer Science at Ben-Gurion University of the Negev, Israel. El-Sana's research interests include 3D interactive graphics, multi-resolution hierarchies, geometric modeling, computational geometry, virtual environments, and distributed and scientific visualization. His research focus is on polygonal simplification, occlusion culling, accelerating rendering, remote/distributed visualization, and exploring the applications of virtual reality in engineering, science, and medicine. El-Sana received a B.Sc. and M.Sc. in Computer Science from Ben-Gurion University of the Negev, Israel in 1991 and 1993. He received a Ph.D. in Computer Science from the State University of New York at Stony Brook in 1999. Before his Ph.D. studies, El-Sana worked as a scientific programmer at Scitex Corporation to develop commercial image processing and pre-press systems. While a Ph.D. student, he worked at IBM T. J. Watson Research Center on occlusion culling algorithms. El-Sana has published over 20 papers in international conferences and journals. He is a member of the ACM, Eurographics, and IEEE.

Leila De Floriani

Dipartimento di Informatica e Scienze dell'Informazione
Universita' di Genova
Via Dodecaneso, 35
16146 Genova - ITALY
E-mail: deflo@disi.unige.it

Leila De Floriani is Professor of Computer Science at the University of Genova, Genova, Italy, since 1990. She received the Laurea degree in Mathematics from the University of Genova in 1977. From 1982 to 1990 she has been a Senior Scientist at the Italian National Research Council. She has been a visiting professor at several Universities and research institutions in the United States. She is currently a visiting professor at the Department of Computer Science of the University of Maryland (USA). Leila De Floriani has written over 140 technical publications on the subjects of geometric modeling, computer graphics, algorithms and data structures for spatial data handling. Her current research is concerned with geometric algorithms and data structures for scientific data visualization, solid modeling, Geographic Information Systems and computer vision. She has received several research grants form the National Research Council of Italy (CNR), from the Italian Ministry of Science and Scientific and Technological Research (MURST), and from the European Community. She has been a member of the Program Committee of several international conferences and on the editorial board of international journals in the field. She is a member of ACM, IEEE Computer Society, and member and fellow of the International Association for Pattern Recognition (IAPR).

Enrico Puppo
Dipartimento di Informatica e Scienze dell'Informazione
Universita' di Genova
Via Dodecaneso, 35
16146 Genova - ITALY
E-mail: puppo@disi.unige.it

Enrico Puppo is Professor of Computer Science at the Department of Computer and Information Sciences (DISI) of the University of Genova. He received a Laurea in Mathematics from the University of Genova, Italy, in March 1986. From April 1986 to October 1998 he has been research assistant (until November 1988), and research scientist (from December 1988) at the Institute for Applied Mathematics of the National Research Council of Italy. From November 1998 to October 2000 he has been Associate Professor at DISI. In different periods, between 1989 and 1992, he has been visiting researcher at the Center for Automation Research of the University of Maryland. Enrico Puppo has written over 70 scientific publications on the subjects of algorithms and data structures for spatial data handling, geometric modeling, computational geometry, parallel algorithms, and image processing. His current research interests are in multiresolution modeling, geometric algorithms and data structures, and object reconstruction, with applications to computer graphics, Geographical Information Systems, scientific visualization, and computer vision. Enrico Puppo is a member of ACM, IEEE Computer Society, and International Association for Pattern Recognition (IAPR).

Arik Shamir
School of Computer Science
The Interdisciplinary Center
P.O.B. 167 Herzelia, 46150
Israel
E-mail: arik@idc.ac.il

Ariel Shamir is a Lecturer (assistant professor) at the school of Computer Science at the Interdisciplinary Center in Israel. His research interests include visualization, geometric modeling, computer graphics, virtual reality and computer aided design. Shamir's recent research concentrates on rendering, compression and transmission of dynamic geometric models employing mutli-resolution techniques, he has several publications in journals and international refereed conferences. Shamir Received his B.Sc. and M.Sc. degrees in math and computer science Cum Laude from the Hebrew University in Jerusalem in 1991 and 1996. His Ph.D. in computer science in 1999 also from the Hebrew University in Jerusalem. Dr. Shamir spent two years as a post-doctoral fellow at the center for computational visualization at the University of Austin in Texas. Dr. Shamir has a broad commercial experience as a programmer, project leader and algorithm designer. In the last five years and until 2001 Dr. Shamir also held the position of manager of research and development in Heshev L.T.D., a commercial software company for typeface design and manufacturing. He is a member of the IEEE.


View-Dependent Rendering
Introduction

Jihad El-Sana

Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva Israel 84105

Motivation

In real time visualization we often require

- Realism
- Interactivity

However

- Realism requires detailed representation which demands large number of polygons
- Rendering time is proportional to size of the rendered dataset.

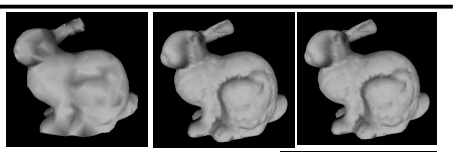
Solutions

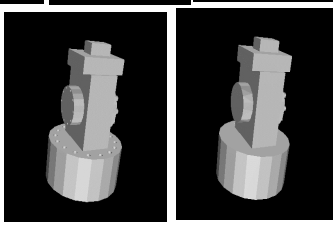
Software and algorithmic solutions try to reduce the complexity of polygonal datasets in a smart manner.

- Level-of-detail rendering
  - Discrete levels of detail
  - Continuous levels of detail
- Occlusion culling
- Image-based rendering

Discrete Level of Detail

Resolution in term of vertices and polygons





- Fixed level of detail
- Far objects are represented in low resolution
- Close objects are represented in high resolution

Continuous Level of Detail

- Multi-Resolution Hierarchies that encode the various level of detail
- Continuous means: The difference between two levels of detail could be only one simplification operation.
- Edge Collapse based approaches
  - Progressive Meshes (Hoppe 96)
- Non Edge Collapse based approaches
  - Multi-Triangulation (De Floriani et al)
  - Vertex Tree (Luebke 97)

However ..

Large datasets consist of few objects

- Low resolution is not appropriate for close-to-viewer regions.
- Unutilized high resolution for far regions may affect interactivity.

Also what about?

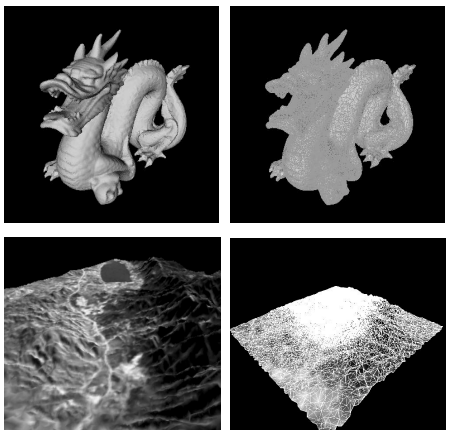
- Light position, visual acuity, silhouettes, and view direction.

View-Dependent Rendering

- View-dependent rendering algorithms construct multi-resolution hierarchies off-line
- At run time the constructed hierarchies are used to extract an appropriate level of detail for rendering base on
  - View position and direction
  - Illumination
- It can also
  - Emphasize Silhouette
  - Take into account back-facing triangles.

View-Dependent Rendering

- Accelerate rendering
- Different level of detail over the same surface
- Smooth transition of resolution
- Coherence between consecutive frames
- Transparent to users



View-Dependent Rendering

- View-Dependent Rendering For Height Fields
  - A Hierarchical triangle-based model for terrain description, *De Floriani at al 93*
  - Fast multiresolution surface meshing, *Gross et al 95*
  - Real-time, continuous level of detail rendering of height fields, *Lindstrom et al 96*
  - A fast algorithm for selective refinement of terrain meshes, *Brown 96*
  - ROAMing terrain: Real-time optimally adapting meshes. *Duchaineau,at al 97*

View-Dependent Rendering

- Geometric (Topology Preserving) Simplification
  - Hierarchical Triangulation for Multiresolution Surface Description, *De Floriani at al 95*
  - Adaptive Real-Time Level-of-detail-based Rendering for polygonal models, *Xia et al 97*
  - View-dependent refinement of progressive meshes, *Hoppe 97*
  - Interactive multiresolution mesh editing.*Zorin et al 97*
  - Generation of multiresolution models from CAD data for real time rendering, *Klein at al 97*
  - Simplicial maps for progressive transmission of polygonal surfaces, *Gu´ eziec et al 98*

View-Dependent Rendering

- Topology Simplification
  - View-dependent simplification of arbitrary polygonal environments, *Luebke et el 97*
  - Generalized view-dependent rendering, *El-Sana et al 99*
- Over Networks
  - Dynamic view-dependent multiresolution on a client-server architecture, *De-Floriani et al 2000*
  - Multi-user view-dependent rendering, *El-Sana 2000*
  - Remote view-dependent rendering, *El-Sana 20001*
  - A stateless client for progressive view-dependent transmission, *Southern at al 2001*

View-Dependent Rendering

- External Memory
  - Smooth view-dependent level-of-detail control and its application to terrain rendering, *Hoppe 98*
  - External memory view-dependent simplification, *El-Sana et al 2000*
- Texture Mapping
  - Rendering of multiresolution models with textures, Schilling et al 98
  - Texture mapping progressive meshes, *Sander et al 2001*
  - AGSphere:Multiresolution structure for complex geometry with texture, *Kim at al 2001*

View-Dependent Rendering

- Dynamic View-Dependent Rendering
  - Multiresolution dynamic meshes with arbitrary deformation. *Shamir et al 2000*
- Other Work
  - Efficiently computing and updating triangle strips for real-time rendering, *El-Sana et al 99*
  - Truly selective refinement of progressive meshes. *Kim et al 2001*
  - Integrating occlusion culling with view-dependent rendering, *El-Sana et al 2001*
  - FastMesh: Efficient view-dependent meshing, *Pajarola 2001*

The Multi-Tessellation:

a Framework for Multiresolution Mesh Representation

Leila De Floriani

Department of Computer and Information Sciences
University of Genova, Genova (Italy)

http://www.disi.unige.it/research/Geometric_modeling/

The Multi-Tessellation (MT)

- *A collection of mesh modifications* describing small portions of a spatial object at different LODs
- Suitable *dependency relation* that allows selecting subsets of modifications (according to application-dependent criteria)
- *Properties:*
  - independent of the properties of the modifications
  - based on a "natural" notion of dependency
  - dimension-independent

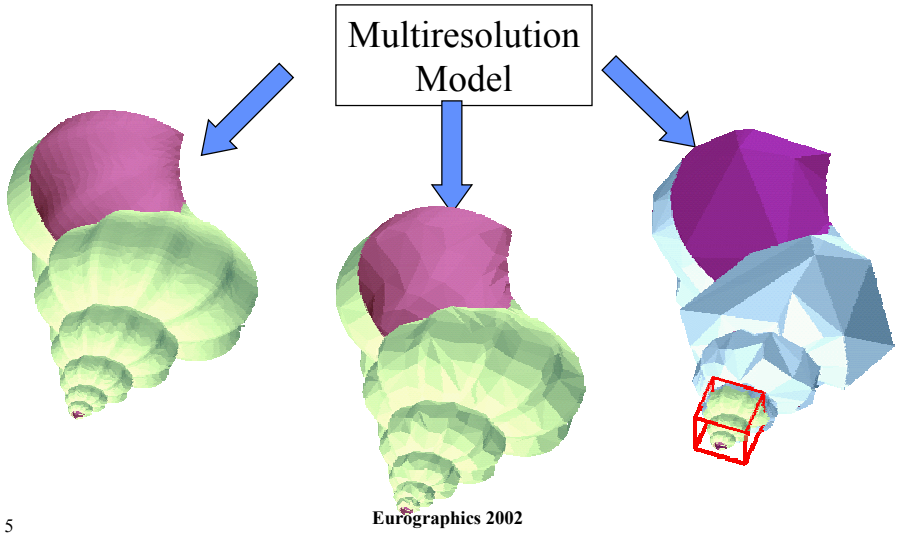
Introduction

- *Multiresolution geometric models* support representation and processing of spatial objects at different Levels-of-Detail (LODs)
- *Applications*: terrain modeling, virtual reality, scientific data visualization, etc.
- Multiresolution geometric models are based on a decomposition of a spatial object into cells forming a *mesh*
- *Accuracy* of a mesh in representing a spatial object related to the number and to the density of the cells in the mesh: *resolution* of the mesh
- It is often necessary to *locally adapt* the accuracy of a mesh in different parts of the object: *variable-resolution meshes*

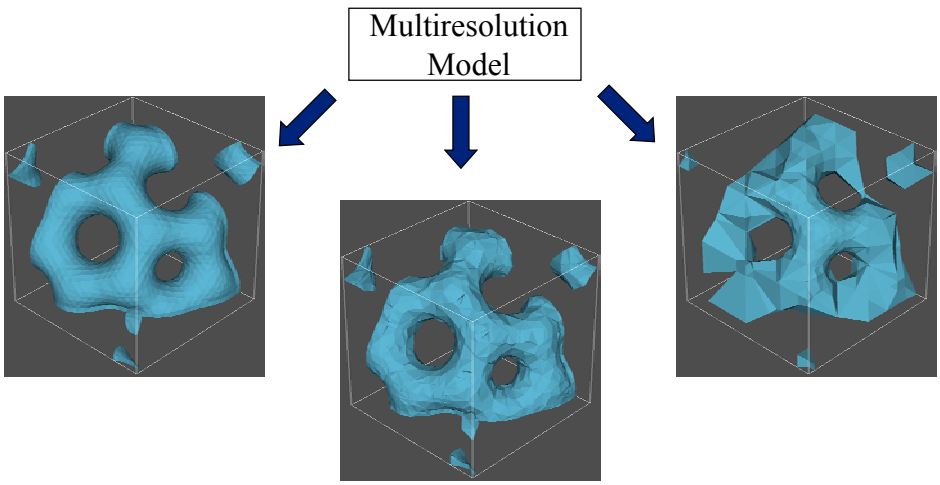
Introduction

- Two ways of computing a mesh at a certain (variable) LOD:
  1. *on-the-fly construction* through simplification techniques
  2. *use of a multiresolution model*
- Multiresolution model= data structure built *off-line* which
  - *preprocesses* and *organizes* a collection of alternative mesh representations of a spatial object
  - can be efficiently *queried* according to parameters specified by an application task to *extract* adaptively refined meshes *on-line*

A multiresolution model for a triangulated surface



A multiresolution model for a 3D scalar field



Multiresolution models

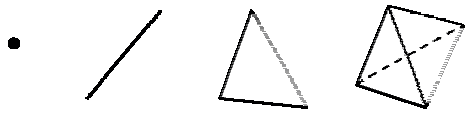
- *Existing multiresolution models:*
  - Designed for *specific classes of applications* (e.g., visualization, terrain modeling, etc.)
  - Designed for performing *only* some operations efficiently
  - Usually rely *on specific construction strategies*
- *Requirements for a multiresolution model:*
  - low overhead with respect to storing the mesh at full resolution
  - support powerful and efficient algorithms for *selective refinement*, i.e., extraction of variable-resolution meshes

Overview

- Background notions:
  - Simplicial meshes
  - Modification in a simplicial mesh
- The Multi-Tessellation:
  - Definition
  - Instances of the Multi-Tessellation
- Selective refinement queries
- Techniques for selective refinement

Simplicial meshes

- A *k-dimensional simplex* (or *k-simplex*) σ in E^d: locus of the points in E^d that can be expressed as the convex combination of $k+1$ linearly independent points.
- Such points are called *vertices* of σ



- Given a *k*-simplex σ, any *s*-simplex τ generated by a subset of $s+1 <= k$ vertices of σ is called an *s-face* of σ.

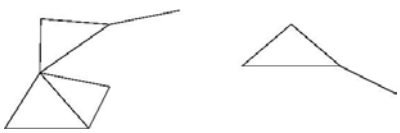
Simplicial Meshes

- A finite collection Σ of simplexes is a (*conforming) simplicial mesh* when the following conditions hold:

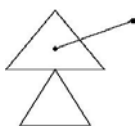
  1.for each simplex $\sigma \in \Sigma$, all faces of σ belong to Σ.

  2. for each pair of simplexes σ and τ ,either $\sigma \cap \tau = \varnothing$ or $\sigma \cap \tau$ is is a face of both σ and τ.

  If *d* is the maximum of the orders of the simplexes of Σ,then *d* is the order of Σ, and Σ is a *d-simplicial mesh*.



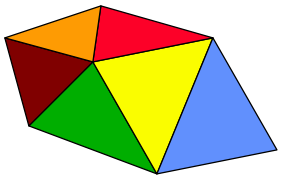
2D simplicial meshes



A set of simplexes not forming a mesh

Simplicial meshes with a manifold domain

- *Carrier* (or *domain*) of a mesh Σ, denoted $D(\Sigma)$: union as a point-set of the cells of Σ
- We consider simplicial meshes with a manifold domain

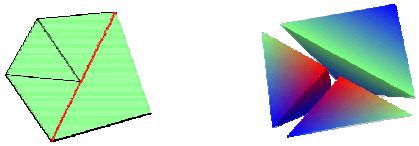


a triangle mesh in E^2

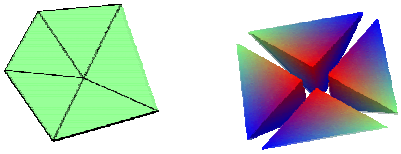
- Every k-simplex, with $k<d$, is generated by a subset of vertices of some d-simplex

Non-conforming simplicial meshes

- *Non-conforming simplicial meshes* are collections of simplexes such that the intersection of two simplexes σ and τ does not necessarily consist of simplexes belonging to both σ and τ
- Examples of non-conforming meshes:

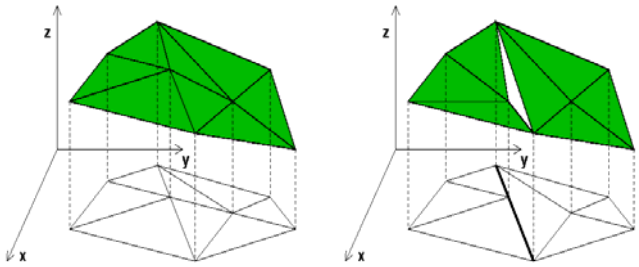


- Examples of conforming simplicial meshes:



Simplicial meshes as the basis for surface approximation

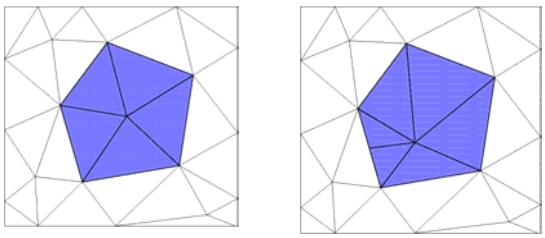
- *(Conforming) simplicial meshes* are a way of ensuring a crack-free surface approximation



Modifications in a mesh

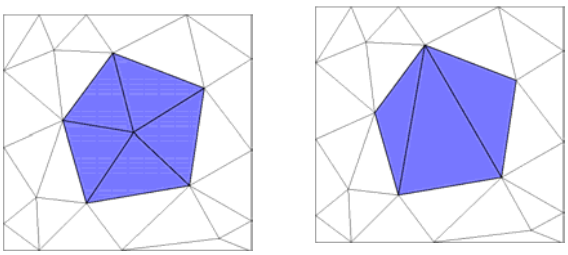
A *modification* of a mesh Σ: pair of meshes $M=(\Sigma_1, \Sigma_2)$

- Σ_1 is a sub-mesh of Σ
- Σ_2 replaces Σ_1 in Σ by filling the hole left by Σ_1.



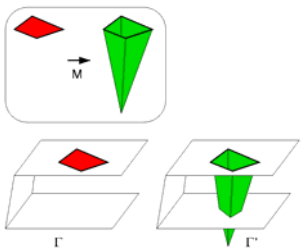
Conforming modifications in a mesh

- A modification $M=(\Sigma_1, \Sigma_2)$ is *conforming* when
  - Σ_1 and Σ_2 are conforming meshes
  - the combinatorial boundary of Σ_1 consists of the same set of cells as that of Σ_2.



Conforming modifications and simplicial meshes

- A (conforming) modification applied to a simplicial mesh may generate intersection of simplexes



- The result of applying a *conforming modification* to a simplicial mesh, if it is a mesh, is a (*conforming) simplicial mesh.*

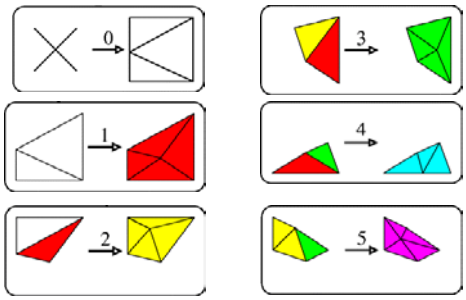
Properties of modifications

- A modification $M=(\Gamma_1,\Gamma_2)$ is a *refinement* modification if Γ_2 has more cells than Γ_1; it is a *coarsening* modification, otherwise.
- *Notation*
  - for a refinement: $M=(M^-; M^+)$
  - for a coarsening: $M=(M^+; M^-)$

  Note: in what follows we will usually use *refinement modifications*

The Multi-Tessellation

- Γ_0: a *d*-dimensional mesh (called the *base mesh*)
- $\{M_1, M_2,\ldots,M_h\}$: a set of *d*-dimensional *conforming refinement modifications* such that for any cell σ in M_i^-, σ belongs either to Γ_0 or to exactly one M_j^+ (with $j \neq i$)

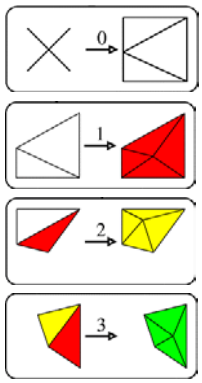


The Multi-Tessellation: dependency relation

- A modification M_j *directly depends* on a modification M_i, with $i \neq j$, if M_j removes some d-simplex introduced by M_i

Example:

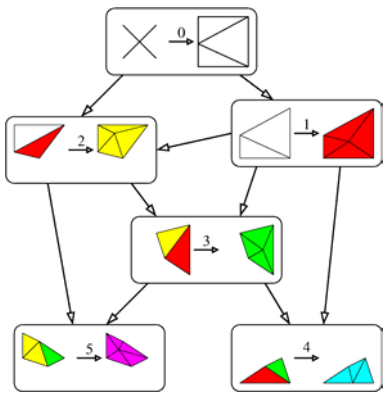
- M_1 depend on M_0
- M_3 directly depends on both M_1 and M_2



Multi-Tessellation as a DAG

- If the transitive closure $\prec$ of the dependency relation is a partial order, then
  $\boldsymbol{M}=(\Gamma_0, \{M_1, M_2,\ldots,M_h\}, \prec)$ is a *Multi-Tessellation (MT).*

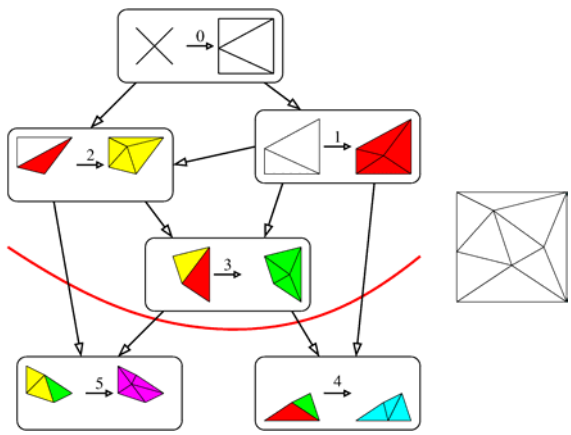
- MT described by Directed Acyclic Graph (DAG) in which *nodes* are modifications and *arcs* direct dependency links



Closed subsets of modifications and extracted meshes

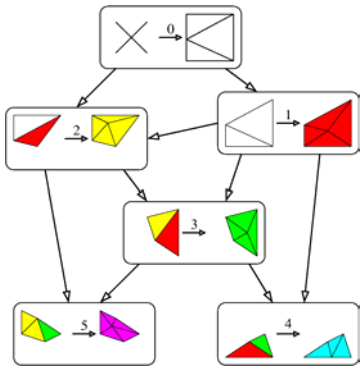
- A subset S of modifications in an MT is *closed (with respect to the partial order)* iff
  for each modification M_j in S, all modifications on which M_j depends, are in S
- A *closed subset S* plus the *base mesh* Γ_0 define a collection of cells Γ_S: result from applying the modifications in S to Γ_0.
- If Γ_S is a mesh, then Γ_S is called the *extracted mesh* associated with S

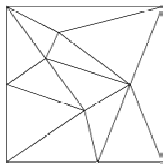
A Closed Subset and the Corresponding Extracted Mesh



Reference Mesh

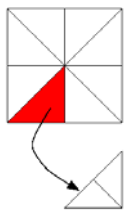
Any sequence of modifications in $\boldsymbol{M}$ corresponding to a total order extending $\prec$ produces the mesh at full resolution, called the *reference mesh*



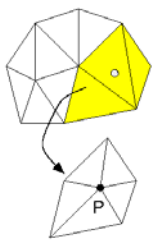


The Multi-Tessellation as a general framework

- Multiresolution meshes proposed in the literature differ in
  - the *type of modifications* involved
  - the properties of the *dependency relation*



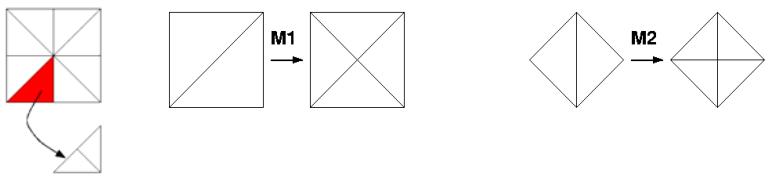
- Multiresolution meshes can be *classified* into:
  - *Nested models*: modifications expand just a single cell and are not conforming
  - *Non-nested models*: modifications affect several cells and are conforming



Hierarchy of right triangles

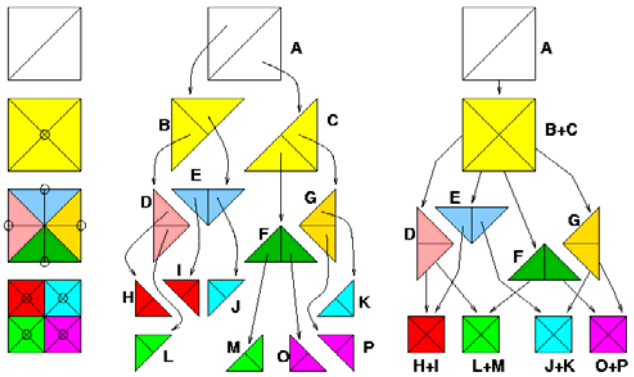
[Lindstrom et al., 1996; Evans et al., 1997; Duchaineau et al., 1997; Pajarola, 1998]

- *Subdivision rule*: replace a triangle σ with two triangles obtained by splitting σ through the midpoint of its longest edge.
- Corresponding *conforming modification* obtained by *clustering* the two triangles sharing the edge to be split



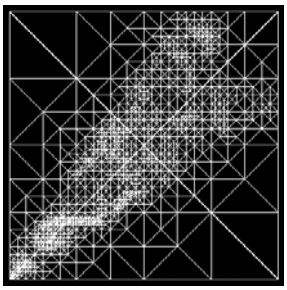
A hierarchy of right triangles as an MT

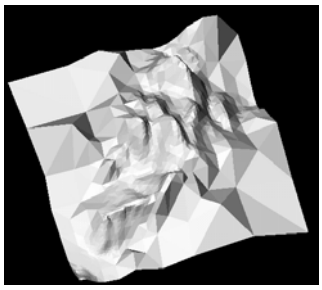
Each node (with the exception of boundary nodes) has *two* parents and *four* children



A hierarchy of right triangles as an MT

A variable-resolution mesh extracted from a nested model based on triangle bisection

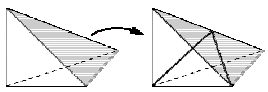




Hierarchy of tetrahedra

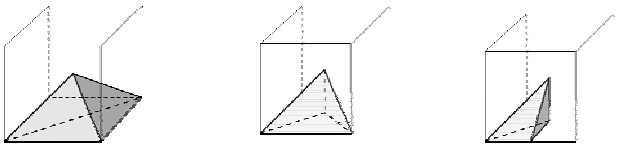
[Rivara and Levin, 1992; Maubach, 1994; Zhou et al., 1997; Ohlberger and Rumpf, 1997; Gerstner et al., 1999-2000; Lee, et al., 2000]

- It consists of bisecting a tetrahedron along its longest edge



- It generates *three* classes of congruent tetrahedral shapes from an initial subdivision of the cubic domain into six tetrahedra

1/2 pyramid *1/4 pyramid* *1/8 pyramid*



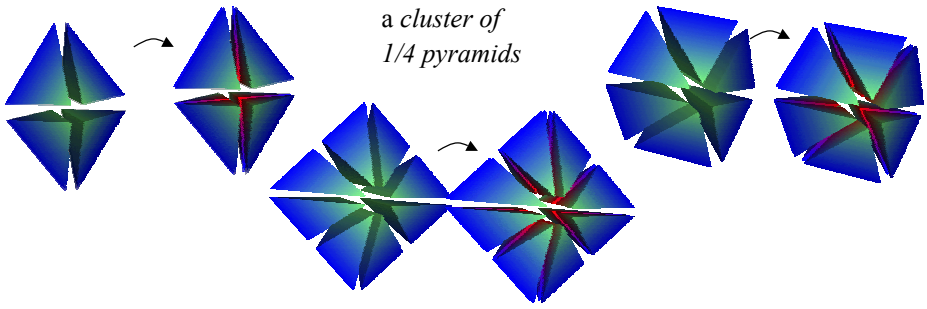
Conforming modifications defined by tetrahedron bisection

Three types of *clusters* of tetrahedra around a bisected edge

a *cluster of 1/2 pyramids*

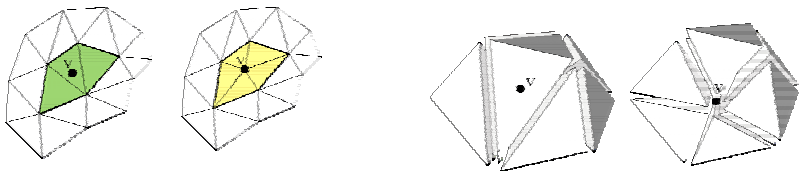
a *cluster of 1/4 pyramids*

a *cluster of 1/8 pyramids*



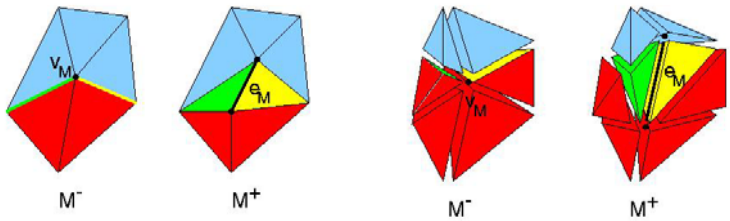
Irregular multiresolution models

- Models based on *vertex insertion/removal* in *2D*
  [De Floriani, 1989; de Berg and Dobrindt, 1995; Cignoni et al., 1995; Brown, 1996/97; Klein and Strasser, 1996; Klein and Gumhold, 1998; Puppo, 1996; De Floriani et al., 97/98/2000]
- Models based on *vertex insertion/removal* in *3D*
  [Cignoni et al., 1994; De Floriani et al., 1995]



Irregular multiresolution models

- Models based on *vertex split/edge collapse* in *2D*
  [Hoppe, 1996/97/98; Xia et al., 1996/97; Gueziec et al., 1998; Kobbelt et al., 1998; El-Sana and Varshney, 1999]
- Models based on *vertex split/edge collapse* in *3D*
  [Gross and Stadt, 1998; Popovic and Hoppe, 1997; Cignoni et al., 2002; Danovaro and De Floriani, 2002]



Models based vertex insertion/removal

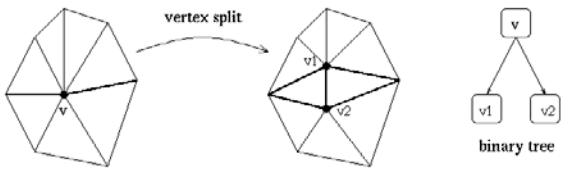
- *Progressive models:* keep just the sequence of modifications performed in refinement or decimation
  [Klein and Strasser, 1996]
- *Layered models:* keep a collection of meshes plus interference links among consecutive levels
  [De Floriani, 1989; de Berg and Dobrindt, 1995; Cignoni et al., 1994; De Floriani et al., 1995]
- *DAG-based models:* the DAG is explicitly encoded
  [Brown, 1996/97; Klein and Strasser, 1996; Klein and Gumhold, 1998; Puppo, 1996; De Floriani et al., 97/98/2000]

Models based vertex split/edge collapse

- *Progressive models:* keep just the reverse sequence of modifications performed in mesh decimation through edge collapse

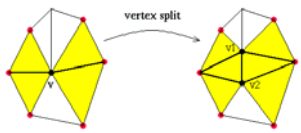
  [Hoppe, 1996; Gross and Stadt, 1998; Popovic and Hoppe, 1997]

- *Vertex hierarchies:* a vertex depends on the vertex that has been split to create it plus a few neighboring vertices
  [Hoppe, 1997; Xia et al., 1997, Gueziec et al., 1998; El Sana and Varshney, 1999; Cignoni et al, 2002]

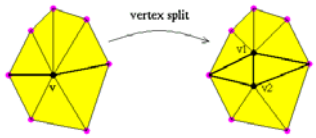


Models based on vertex split/edge collapse

- Two types of dependencies (in both case DAG of vertex dependencies, in some proposals implicitly encoded):
  - Hoppe, 1997

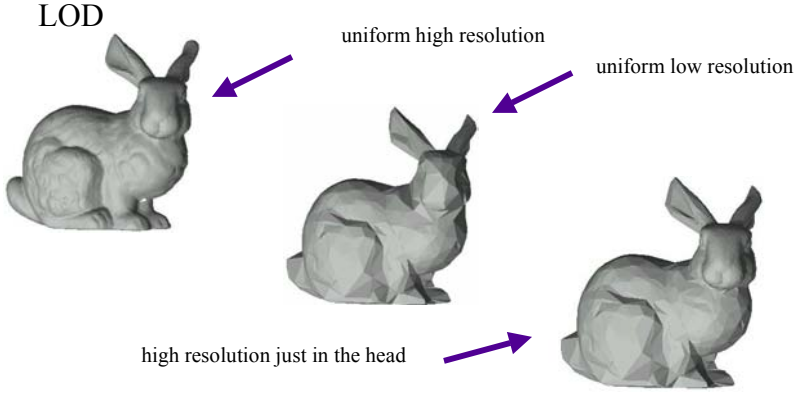


  - Xia et al., 1997, Gueziec et al., 1998; El Sana and Varshney, 1999; Cignoni et al, 2002



Selective refinement queries on a Multi-Tessellation

- Extract from a multiresolution model a mesh satisfying some application-dependent requirements based on LOD



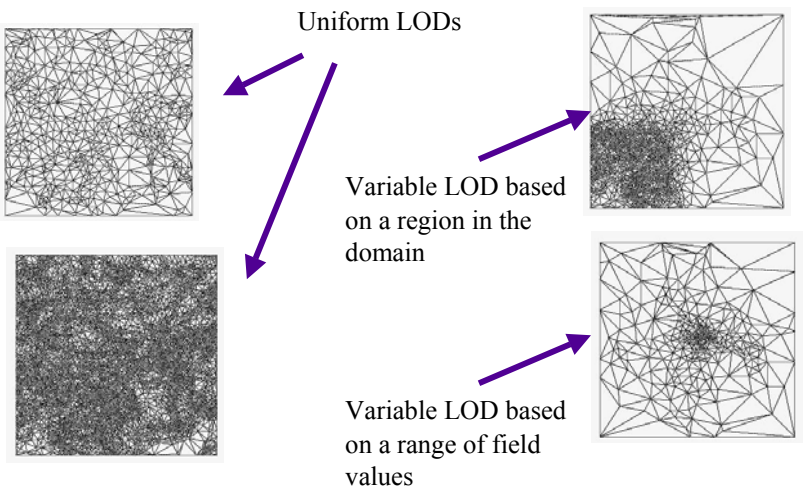
Level-Of-Detail (LOD)

- *LOD criterion* τ: Boolean function defined over the *d*-simplexes σ of an MT
  $\tau(\sigma)$ =*true* if σ satisfies the LOD requirements
- A mesh *satisfies a given LOD criterion* τ if and only if
  $\tau(\sigma)$= *true* for all d-simplexes σ of Γ
- An *approximation error* is usually associated with the *d*-simplexes of an MT: distance, according to some metric, with respect to a reference surface (or hypersurface)

Examples of LOD criteria

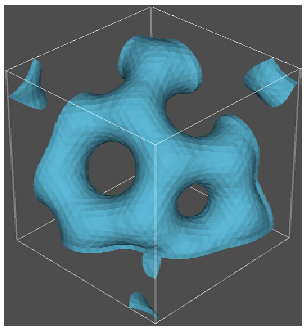
- *Uniform LOD*:
  - $\tau(\sigma)$= *true* if the error associated with σ is less or equal to a constant threshold ε
- *Variable LOD*:
  - $\tau(\sigma)$= *true* if the error associated with σ is less or equal to the maximum over σ of a threshold function ϕ defined at each point of the domain (e.g., a view-dependent function)
- Special case of *variable LOD for scalar fields*:
  - the threshold function ϕ depends on the value of the field at each point of the domain (e.g, $\phi(p)$ small at a set of interesting field values, large otherwise)

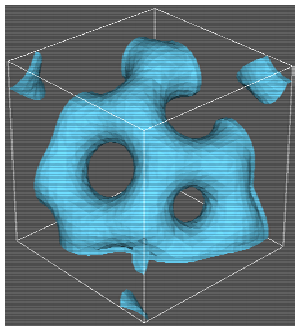
Examples of LOD criteria on a terrain



Example:uniform LOD meshes from a 3D scalar field

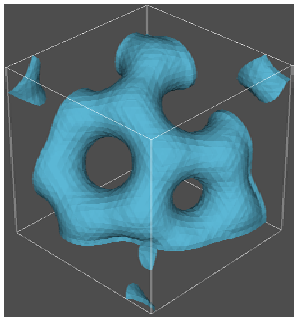
- *Error threshold*: *0.5%* of the absolute range of the field value
- *Size of the extracted mesh:* 44.7% of size of the full resolution mesh

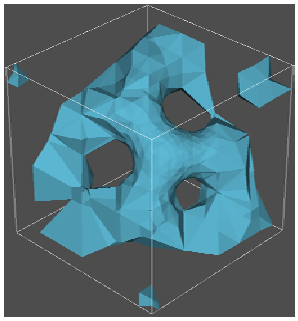




Example: variable LOD based on a region of interest

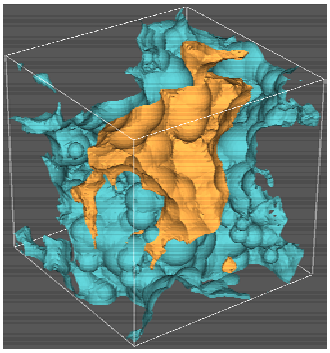
- A mesh with a *small approximation error* inside a given *region of interest* (0.1% of the field value within the box)
- *Size of the extracted mesh:* 6 % of the size of the mesh at uniform LOD with error equal to 0.1%

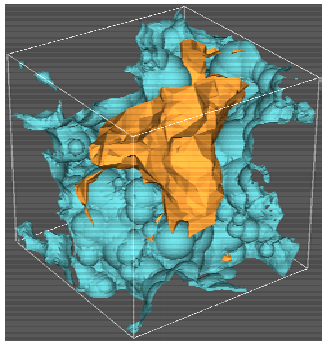




Example: variable LOD based on the field value

- *Error threshold* on the tetrahedra intersected by isosurface of value 1.27: *0.1%* of the range of the field values
- *Size of the extracted mesh:* 26% of the size of the mesh at uniform LOD with error equal to 0.1%





Selective Refinement Query

Given a Multi-Tessellation ***M*** and a LOD criterion τ

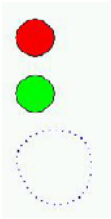
- *extract* from ***M*** the mesh of *minimum size* Γ_S satisfying τ

or, equivalently,

- *extrac*t from ***M*** the mesh Γ_S associated with the closed subset S of minimum cardinality such that Γ_S satisfies τ

Algorithms for selective refinement

- *Top-down* approach: extract the mesh through a top-down DAG traversal
- *Incremental* approach: update a previously extracted mesh through local refinement or local coarsening
- *Pictorial Notation:*



red = modification M such that $\tau(M^-)$=*false*

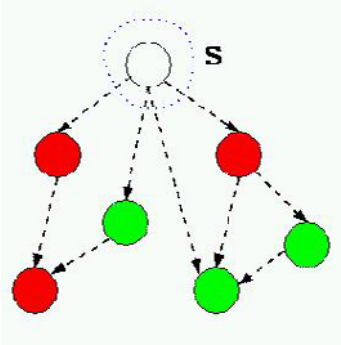
green = modification M such that $\tau(M)$=*true*

dotted blue line = current set S

Remark: We need to apply all *red* modifications

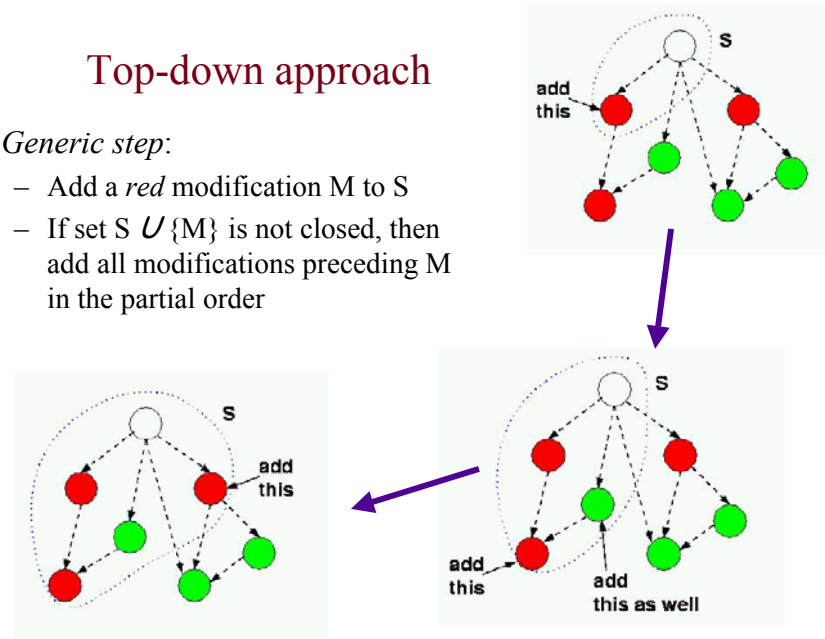
Top-down approach

- *Initialization step*:
  - set S = empty
  - current mesh $\Gamma_S = \Gamma_0$



Top-down approach

- *Generic step*:
  - Add a *red* modification M to S
  - If set S $\cup$ {M} is not closed, then add all modifications preceding M in the partial order

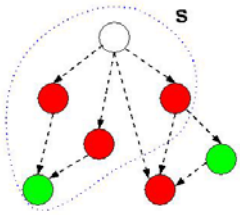


Top-down approach: implementation issues

- It can be efficiently implemented as a *depth-first traversal* (e.g., in the case of hierarchies of triangles or tetrahedra)
- If a *LOD criterion based on approximation error* is used, it is often implemented by using a *priority queue* ordered according to decreasing error values
  - modifications with a largest error value are performed first
  - the resulting algorithm is *interruptible* (i.e., intermediate steps produce approximations of the resulting mesh)

Incremental approach

- *Initialization step:*
  - set S = the set of modifications generated by a *previous query*
  - current mesh Γ_S: mesh corresponding to S

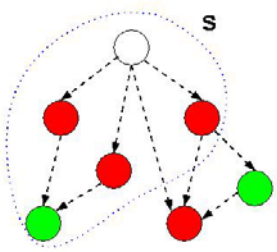


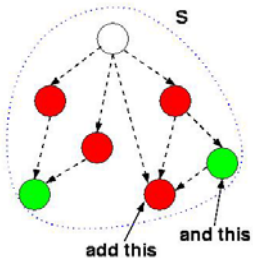
Red modifications must be added, *green* modifications must be removed

Incremental approach

- *Generic step*:

  update set S and mesh Γ_S by adding r*ed* modifications, and *all* those modifications that are necessary to maintain S a closed set

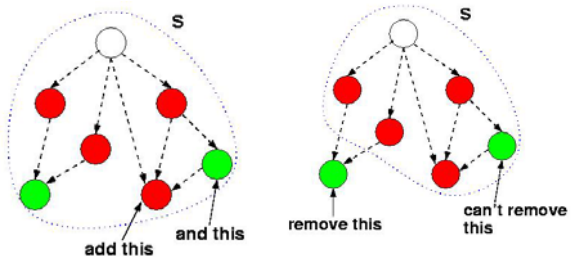




Incremental approach

- *Generic step*:

  update set S and mesh Γ_S by removing *green* modifications M such that S\{M} is a closed set



- Remark: usually implemented by using *two* priority queues

Basic primitives for selective refinement

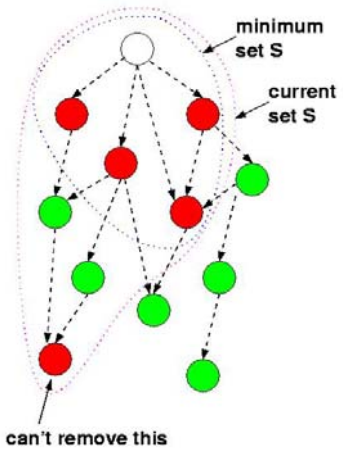
- *Acceptance_Test*:

  given a modification M, evaluate if M- satisfies a given LOD criterion τ
- *Feasibility_Test*: given a closed set S, decide
  - if a modification M *not in* S can be *added* to S
  - if a modification M *in* S can be *removed* from S
- *Dependency_Retrieval:*

  retrieve all modifications which must be added to S in order to make the insertion of a modification M feasible
- *Current_Mesh_Update:*
  - *add* a modification M to S and *refine* mesh Γ_S
  - *Remove* a modification M from S and *coarsen* mesh Γ_S

Optimality Issues

- *Top-down* approach always returns the mesh of *minimum size* satisfying the LOD criterion
- Incremental approach returns a mesh of *minimum size* provided that ***M*** does not contain any modification M such that M^- satisfies τ and M^+ does not satisfy τ
- For *error-based LOD criteria*: provided that the error monotonically decreases at each modification



References: survey papers

- *Triangle meshes*:

  L.De Floriani, P. Magillo, Multiresolution meshes: Models and data structures, *Principles of Multiresolution Geometric Modeling* (M.Floater, A.Iske, and E.Qwak (editors)), Lecture Notes in Mathematics, Springer-Verlag, 2002.
- *Irregular tetrahedral meshes*:

  E. Danovaro, L. De Floriani, P. Magillo, E. Puppo, Data Structures for 3D Multi-Tessellations: an Overview, *Proceedings Dagstuhl Scientific Visualization Seminar* (F.H.Post, G.P.Bonneau, and G.M.Nielson (editors)), Kluwer Academic Publishers, 2002.
- *See* also
  - attached list of references
  - `http://www.disi.unige.it/research/Geometric_modeling/`


Generalized View-Dependent Rendering

Jihad El-Sana

(with A. Varshney)

Department of Computer Science

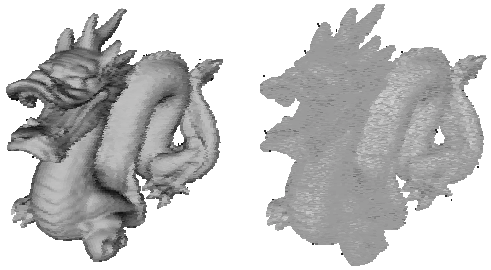
Ben-Gurion University of The Negev

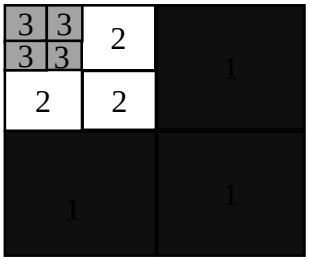
Beer-Sheva, Israel

1

View-Dependent Rendering

- Different levels of detail across different regions of object of the same object
- Seamless merge of the different levels of details
- Vertex hierarchies (forest)





2

Topology Simplification

- Aggressive simplification
- Variable topology simplification
- Connect different objects
- View-Dependent simplification
- Efficient foldover prevention policy
- Real-time

3

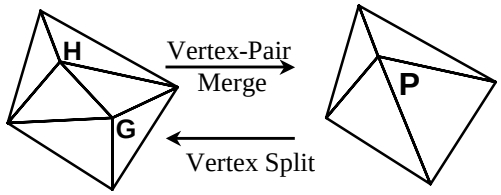
Topology Simplification

- Extend edge-collapse to vertex-pair collapse
- Eliminate dependencies lists
- Accept general datasets
  - Handle Non-Manifold, and multiple objects
- Real-time geometry and topology Simplification
- Treat simplification metric as a parameter

4

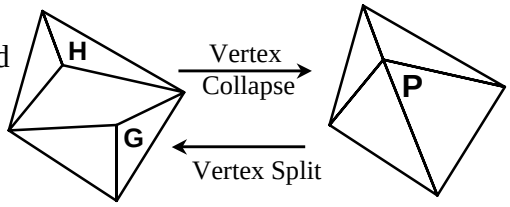
Vertex-Pair Merge

- Connected Via Edge



- No Connection:

A virtual edge is constructed in order to detect the pair to merge.



5

Virtual Edges

- Subdivide the dataset into patches
  - Initialize each triangle to a patch
  - Merge two patches that:
    - Share at least one edge
    - Their normals differ in less that threshold
- Construct Delaunay triangulation using only the vertices on the boundary of the patches

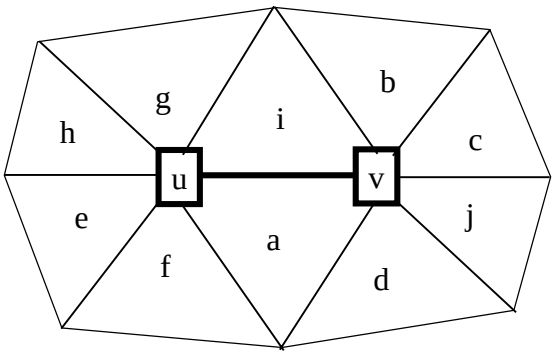
6

Geometry and Topology

- Geometry Simplification First
  - Topology-preserving geometry simplification
  - Topology simplification
- Equal Preference
  - Treat Topology-preserving and Topology-reducing simplification on an equal basis

7

Vertex-Pair Merge: Before

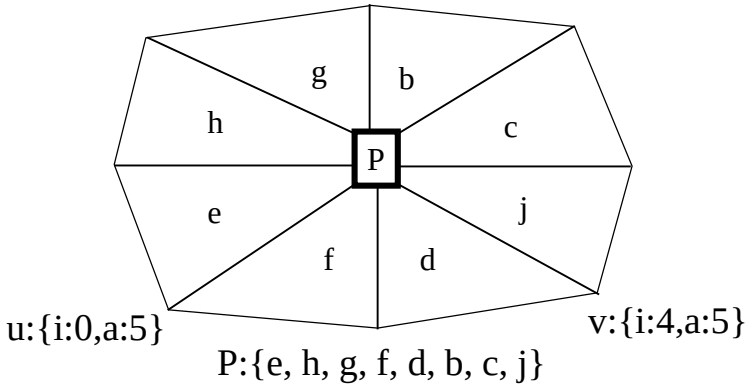


u:{i, e, h, g, f, a}

v:{d, b, c, j, i, a}

8

Vertex-Pair Merge: After



9

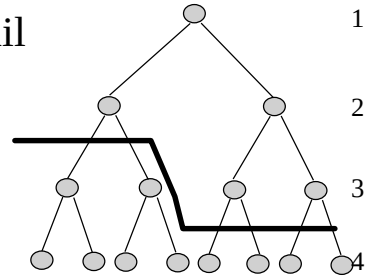
View Dependence Tree

- Supply a “sound” display metric
- Construct the virtual edges set
- Build a heap of all the edges (virtual and real) using the given metric.
- While not empty (heap)
  - Extract minimum edge
  - Collapse its two vertices

10

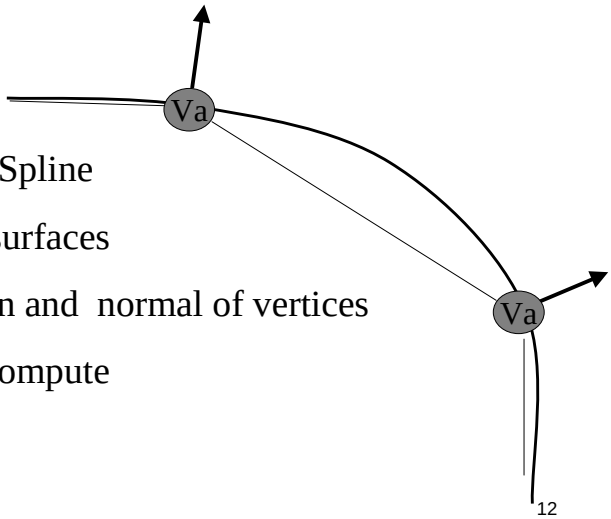
Levels of Detail

- Levels in view-dependence tree represent levels of detail
- Off-line construction of View-dependence trees
- Real-time re-triangulation



11

Spline Metric

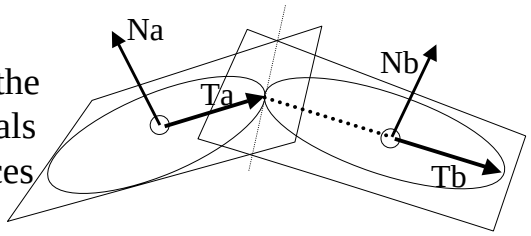


- Based on a cubic Spline
- Simulate curved surfaces
- Combines position and normal of vertices
- Easy and fast to compute

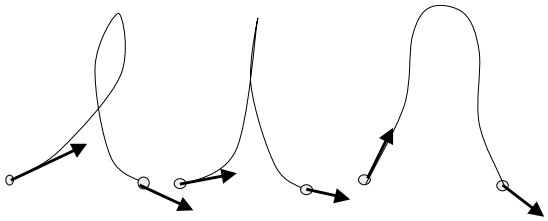
12

Computing Spline Metric

Tangent vectors are estimated based on the approximated normals of the merged vertices



The tangent magnitude is carefully selected to avoid undesired cases



13

Selected Results

13.5K triangle

8.2K triangle

6.9K triangle

2.0K triangle

0.2K triangle

14

Simplification Factors

- Prevent foldovers
  - Implicit Dependencies
- Screen-space projection
- Local illumination
- Visibility culling
- Silhouette boundaries
- Level of detail transfer function

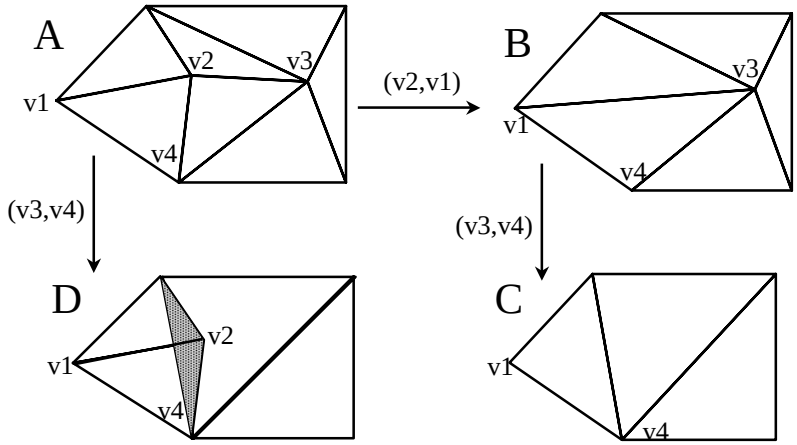
15

Adapt Traversal

```
for each active node n do
 switch(NextStat(n)){
   case SPLIT  : if ( CanSplit(n))
                    Split(n);
   case MERGE: if ( CanMerge(n) &&
                     CanMerge(Sibling(n))
                    Merge(n);
   case STAY   : // No Change on the active-nodes list
 }
```

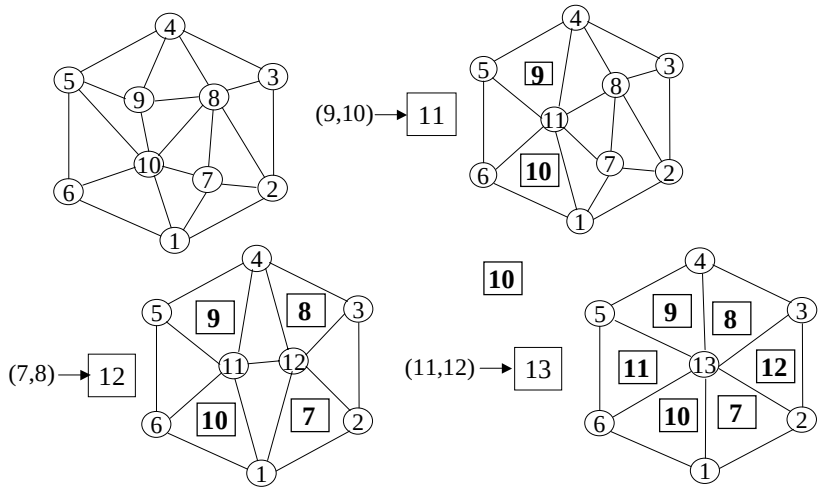
16

Need for Dependencies



17

Implicit Dependencies



18

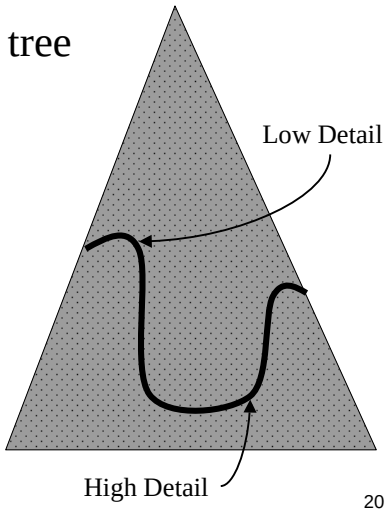
Implicit Dependencies

- Parent *id* is larger than its children *id*-s
- Node *id* is unique integer
- Vertex Split
  - Its *id* is greater than the *id* of all its neighbors
- Vertex-Pair Merge (*u*,*v*)
  - Its parent *id* less than the *id* of neighbors of the two vertices (*u*,*v*)

19

Active Nodes List

- List of view-dependence tree nodes
- Initialized to the root list
- Update
  - Vertex Split
  - Vertex-Pair Merge



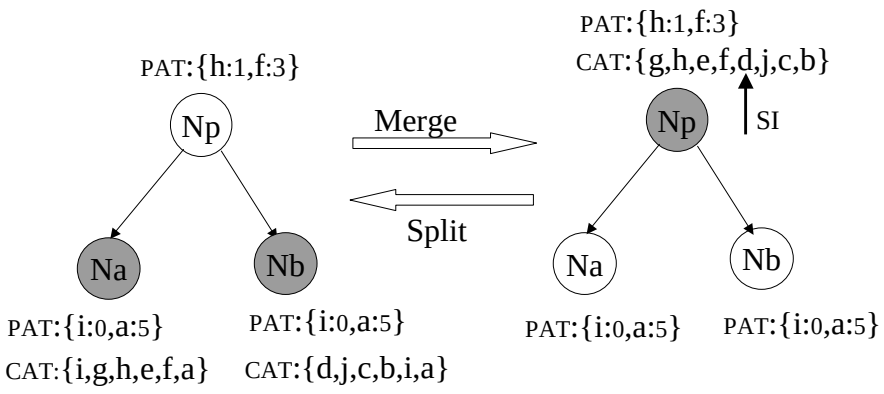
20

Active Triangles List

- Initialized to triangles adjacent to roots
- Add triangles
  - Vertex Split
- Remove triangles
  - Vertex-Pair Merge
- Update triangle
  - Vertex Split
  - Vertex-Pair Merge

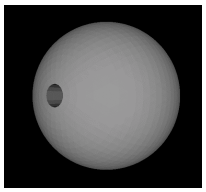
21

Real-Time Merge/Split

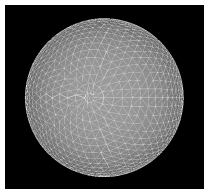


22

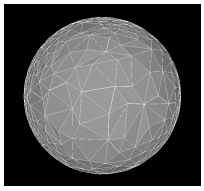
Results



Full Resolution

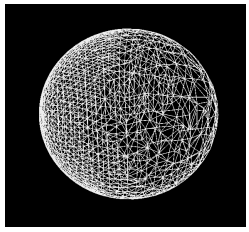


Front



Back

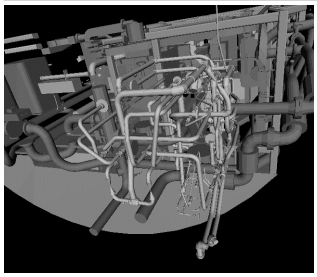
The hole on the back disappeared (simplified)

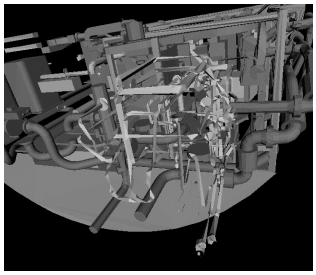


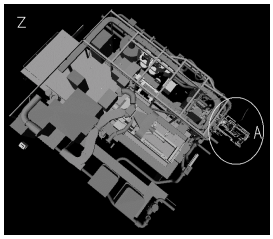
Side View

23

Results







24

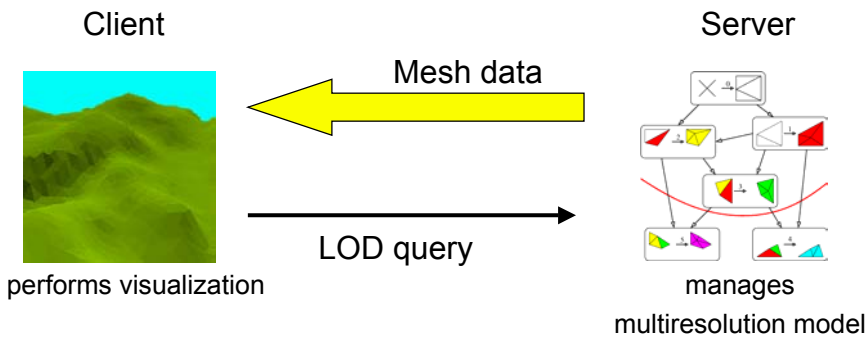
Client/Server View-dependent Rendering

Enrico Puppo

Department of Computer and Information Sciences
University of Genova, Genova (Italy)

http://www.disi.unige.it/research/Geometric_modeling/

The Client/Server Scenario



Constraints

Client has limited on-board memory and processing power

➔ the whole multiresolution model cannot be transferred from server to client

Selectively refined mesh may change from frame to frame

➔ client needs to send LOD parameters to server

➔ server needs to extract and transmit a new mesh to client *in real time*

Communication channel has low bandwidth

➔ a whole selectively refined mesh *cannot be transmitted in real time*

Reducing communication time is the issue!

Solution

- Client loads and maintains only a *current mesh*
- Server performs selective refinement and sends to client *only updates needed to modify the current mesh*

What we need:

➔ *Dynamic selective refinement* algorithm

➔ *Synchronized data structures* at Client and Server sides

➔ *Efficient encoding* to minimize the size of transmitted updates

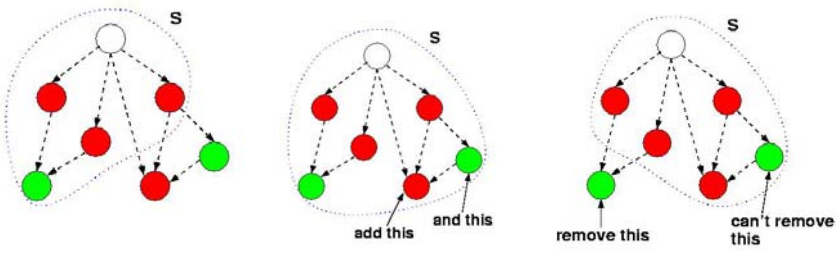
➔ *Caching policy* to minimize the number of transmitted updates

Outline

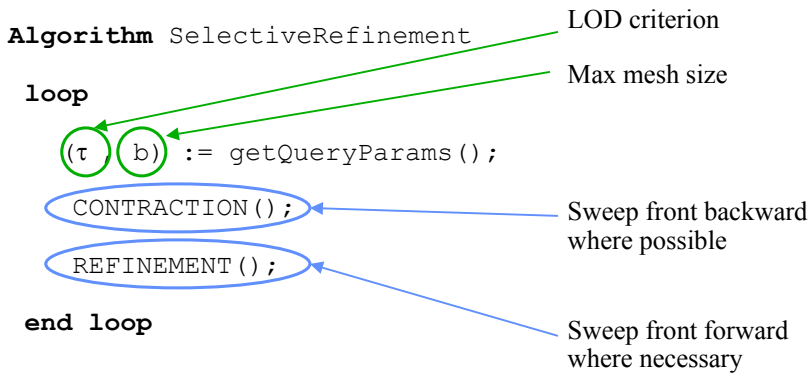
- Dynamic selective refinement algorithm (stand-alone)
- Dynamic selective refinement algorithm (client/server):
  - Transmitted Updates
  - Caching
  - The Client process
- Data structures:
  - Updates
  - Multi-Triangulation

Dynamic selective refinement algorithm

- S is a closed subset of nodes (in the MT)
- Γ_S is the corresponding mesh *(current mesh)*
- S (and Γ_S) must be updated according to new LOD criterion τ

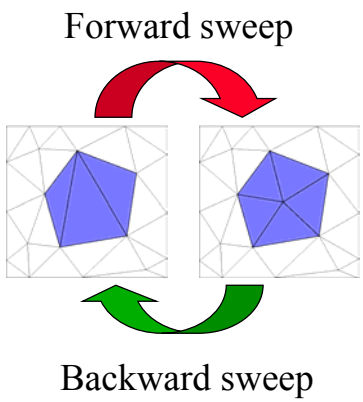


Dynamic selective refinement algorithm



Effects of front sweep

- A forward sweep of the front in S across a node produces a refinement update in Γ_S
- A backward sweep of the front in S across a node produces a coarsening update in Γ_S



Contraction phase

- Initialize a priority queue Q with the leaves of S
- Higher resolution $\Rightarrow$ higher priority
- Extract nodes from Q and eliminte them if either their resolution is too refined or the size of the mesh is too large
- For each node eliminated, update Γ_S accordingly
- If a node become a leaf then add it to Q
- Stop when Q becomes empty

Contraction phase

```
Procedure CONTRACTION()
  EMPTY(Q);
  for all M in LEAVES(S) do INSERT( Q, M, τ(M⁻) );
  while (not IS-EMPTY(Q)) and
        (|Γ_S| > b or PRIORITY(MIN(Q))≤ 0) do
    M : DELETEMIN(Q);
    DELETE( S, M );
    COARSEN( M );
    for all M_1 in PARENTS(M) do
        if (IS-LEAF(S, M_1)) and (M_1 ≠ ROOT()) then
           INSERT( Q, M_1, τ(M_1⁻) );
```

Expansion phase

- Initialize a priority queue Q with all nodes just below the front that do not satisfy LOD criterion
- Lower resolution $\Rightarrow$ higher priority
- Extract nodes from Q and add them to S if the size of the mesh is below maximum size
- If a node to be inserted has parents that do not belong ot S then add them recursively
- For each node inserted, update Γ_S accordingly
- Stop when Q becomes empty

Expansion phase

```
Procedure EXPANSION()
  EMPTY(Q);
  for all M ∉ S that have a parent in S do
     if τ(M⁻)>0 then INSERT( Q, M, -τ(M⁻) );
  while (not IS-EMPTY(Q)) and  |Γ_S| < b
                    and PRIORITY(MIN(Q)) < 0) do
    M : DELETEMIN(Q);
    if not IS-IN(S,M) then RECURSIVEADD(M);
```

Expansion phase

```
Procedure RECURSIVEADD(M)
  for all M' in PARENTS(M) do
     if M' ∉ S then RECURSIVEADD(M');
  if |Γ_S| < b then
     ADD( S, M );
     REFINE( M );
     for all M"∈ CHILDREN(M) do
        if not IS-IN(S,M") and τ(M"⁻)>0 then
           INSERT( Q, M", τ(M"⁻) );
```


Stand-alone implementation

A data structure for the *MT* that supports:

- CHILDREN, PARENT, ROOT

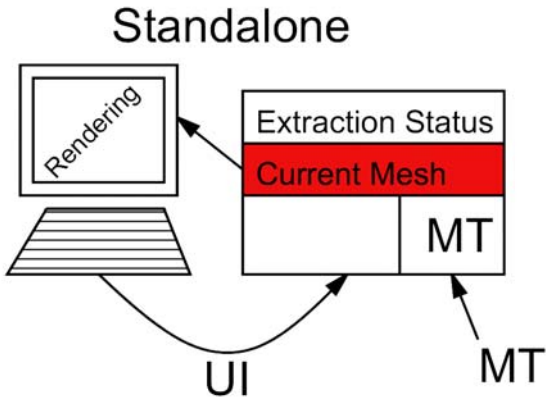
An *extraction status* that supports primitives on current set S:

- ADD, DELETE, IS-IN, LEAVES, IS-LEAF

A data structure for the *current mesh* Γ_S that supports:

- COARSEN, REFINE

The Stand-alone architecture



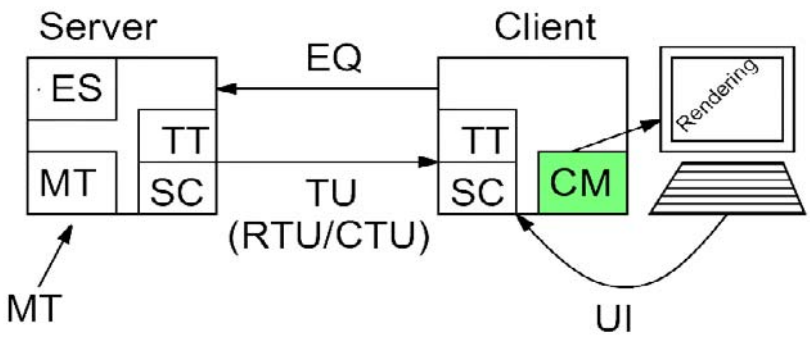
Client/Server implementation

- Two concurrent processes running on different machines
- A bi-directional transmission channel
- The Server:
  - maintains the *MT* and the *Extraction status* S
  - executes the same selective reifnement procedure, but *does not store and modify the Current mesh*
  - receives *query parameters* and modifies the *Extraction status*
  - sends instructions on how to perform COARSEN and REFINE operations on the *current mesh*:
    - *Refining Transmitted Updates (RTUs)*
    - *Coarsening Transmitted Updates (CTUs)*

Client/Server implementation

- The Client:
  - maintains the *Current mesh* Γ_S
  - sends *query parameters*
  - receives *Transmitted Updates*
  - updates the *Current mesh*
  - renders the result

The Client/Server architecture



The Server process

```
Algorithm SelectiveRefinement

 loop

   (τ , b) := getQueryParams();

   CONTRACTION();

   REFINEMENT();

 end loop
```

Puts the process in a stand-by status until parameters are received from the Client

Produces and sends CTUs

Produces and sends RTUs

Contraction phase

```
Procedure CONTRACTION()
  EMPTY(Q);
  for all M in LEAVES(S) do INSERT( Q, M, τ(M⁻) );
  while (not IS-EMPTY(Q)) and
        (|Γ_S| > b or PRIORITY(MIN(Q))≤ 0) do
    M : DELETEMIN(Q);
    DELETE( S, M );
    COARSEN( M );
    for all M_1 in PARENTS(M) do
        if (IS-LEAF(S, M_1)) and (M_1 ≠ ROOT()) then
            INSERT( Q, M_1, τ(M_1⁻) );
```

A CTU is produced and sent here

Expansion phase

```
Procedure EXPANSION()
  EMPTY(Q);
  for all M ∉ S that have a parent in S do
     if τ(M⁻)>0 then INSERT( Q, M, -τ(M⁻) );
  while (not IS-EMPTY(Q)) and  |Γ_S| < b
                    and PRIORITY(MIN(Q)) < 0) do
    M : DELETEMIN(Q);
    if not IS-IN(S,M) then RECURSIVEADD(M);
```


Expansion phase

```
Procedure RECURSIVEADD(M)
  for all M' in PARENTS(M) do
     if M' ∉ S then RECURSIVEADD(M');
  if |Γ_S| < b then
     ADD( S, M );
     REFINE( M );
     for all M"∈ CHILDREN(M) do
        if not IS-IN(S,M") and τ(M"⁻)>0 then
           INSERT( Q, M", τ(M"⁻) );
```

A RTU is produced and sent here

The Transmitted Updates

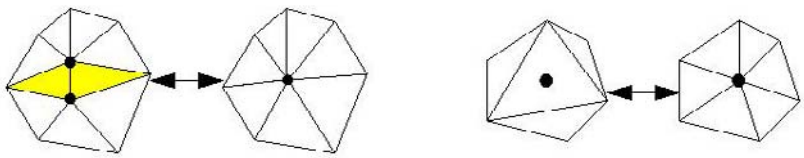
- In principle, a Transmitted Update must specify:
  - a set of cells of the *Current mesh* to be deleted
  - a new set of cells to replace them in the *Current mesh*
- Explicit encoding of sets of cells is too expensive
- If updates in the MT follow a *specific kind of operation*, a more compact *procedural description* can be used:
  - TUs contain perameters that specify *where* and *how* to perform a specific update in the *Current mesh*
- Operations for COARSEN and REFINE are conceptually different

The Transmitted Updates

- Procedures at the Server:
  - RTU-ENCODE(M) takes a MT node M and produces a RTU corresponding to update M^+
  - CTU-ENCODE(M) takes a MT node M and produces a CTU corresponding to update M^-
- Procedures at the Client:
  - RTU-DECODE(u) takes a RTU u and applies it to the *Current mesh*
  - CTU-DECODE(u) takes a CTU u and applies it to the *Current mesh*
- RTU and CTU encoding mechanisms depend on the specific types of updates adopted in a given MT

Types of updates

- Edge collapse / vertex split: *edge-based MT*
- Vertex insertion / removal: *vertex-based MT*



Details later

Caching

- During interaction, certain parts of the mesh may be refined and coarsened multiple times
- Problem: avoid sending the same update many times
- *Thin Client policy:* a Client with small memory
  - the Client maintains some TUs it receives in a *local cache* with limited capacity
  - the Server mirrors the cache and sends either complete TUs, or just a reference to the local cache
- *Fat Client policy:* a Client with large memory
  - the Client builds a (partial) copy of the MT as updates are received
  - the Client mirrors the selective refinement algorithm

Operations on Cache

- Server cache:
  - MEMBER(M) tells whether or ot a node M is in cache
  - ADD-NODE(M) inserts a node into cache (overflow management)
  - GET-CODE(M) gets a key code for a node in cache
- Client cache:
  - ADD-UPDATE(u) inserts the decription of an update into cache (overflow management)
  - GET-UPDATE(key) gets the description of an update from cache

Server procedures

```
Procedure REFINE(M)
  if MEMBER(M) then
     transmit CACHED-TU(GET-CODE(M))
  else
     ADD-NODE(M);
     transmit RTU-ENCODE(M);

Procedure COARSEN(M)
  if MEMBER(M) then
     transmit CACHED-TU(GET-CODE(M))
  else
     ADD-NODE(M);
     transmit CTU-ENCODE(M);
```

Client process

- Initialization plus main loop
- At each cycle:
  - Query parameters are sent to the Server
  - A sequence of CTUs is received
  - Each CTU is either obtained through deconding or retrieved from client cache
  - Corresponding updates are performed on the *current mesh*
  - A sequence of RTUs is received
  - Each RTU is either obtained through deconding or retrieved from client cache
  - Corresponding updates are performed on the *current mesh*

Client process

```
Initialize;
loop
  sendQueryParams (τ , b);
  receive msg;
  while msg ≠ STOP do
    if CACHED-TU(msg,ck) then u := GET-UPD(ck)
    else
      u := CTU-DECODE(msg);
      ADD-UPDATE(u);
    COARSEN(u);
    receive msg;
  ..........  // analogous code for RTUs
```

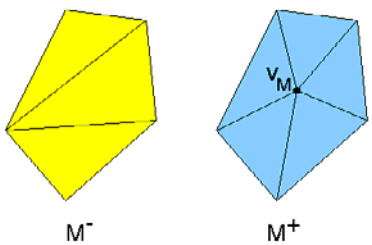
Data structures

- MT (Server)
- Extraction status (Server)
- Current mesh (Client)
- Cache (Client and Server)
- Translation tables (Client and Server): used during encoding/decoding of Tus to relate data structures on the two processes

Key issue: representation of updates

Vertex-Based Updates

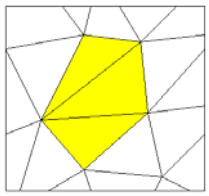
- Refining Update M^+: *vertex insertion*
- Coarsening Update M^- : *vertex removal*

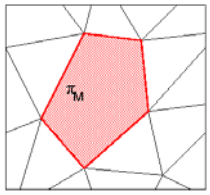


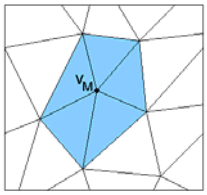
Representing Vertex-Based Updates

Information for performing M^+ (inserting v_M):

- *Patch identification:* find the triangles of M^-
- Deleting such triangles leaves an empty region bounded by a star-shaped polygon π_M
- Polygon π_M is triangulated by a fan of triangles centered at v_M



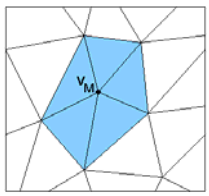


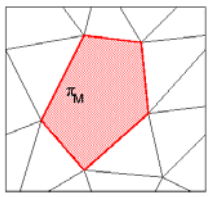


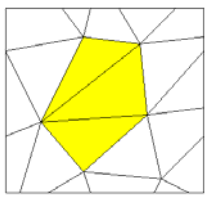
Representing Vertex-Based Updates

Information for performing M^- (removing v_M):

- Deleting triangles incident at v_M leaves an empty region bounded by a star-shaped polygon π_M
- *Boundary re-triangulation:* polygon π_M must be triangulated by triangles of M^-







Representing Vertex-Based Updates

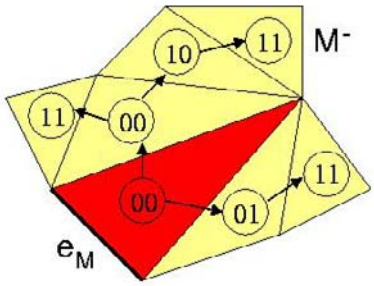
- Patch identification and Boundary re-triangulation are the crucial steps
- A unique mechanism to represent them both
- Store:
  - one of the edges of π_M (reference edge)
  - a bitstream that describes the triangulation of π_M
- The reference edge is used to initilize traversal of the current mesh
- The bitstream drives traversal through triangle adjacencies

Representing Vertex-Based Updates

- Klein & Gumhold [1998]: store all triangulation patterns for a polygon in a table, identify one triangulation by an index to the table
- Andujar & De Floriani [2001]: as above, but don't store the table, provide an algorithm to decode the triangulation from its index
- De Floriani et al. [2000]: encode a sequence of ear cuts necessary to triangulate the polygon
- Taubin et al. [1998]: polygon encoding in the context of progressive transmission of large triangle meshes

An example: Method by Taubin et al.

- Let γ be the triangle of M^- adjacent to the starting edge e_M
- Generate a code of two bits for γ
  - first bit = 0 if the part of M^- lying to the left of γ is empty, 1 otherwise
  - second bit = 0 or 1 applying the same rule to the right part

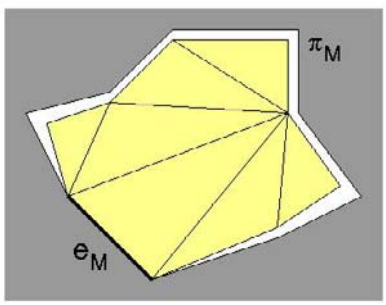


- Recursively activate the process on each non-empty part

An example: Method by Taubin et al.

Use the bitstream to support *Patch identification*

- *Recognize M^- on the current mesh Γ_S*
  - traverse Γ_S from the starting edge, moving from one triangle to another through edge adjacencies
- *Reconstruct M^- inside polygon π_M*
  - create first a triangle at the starting edge, then create other triangles edge-adjacent to already created triangles
  - sew the triangulation to its boundary π_M



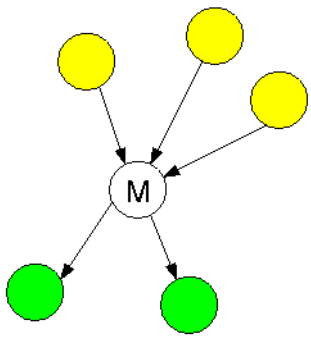
Representing Vertex-Based Updates

Length of the bit stream:

- Klein & Gumhold [1998]: 13 bits + space for the table
- Andujar & De Floriani [2001]: 7 bits
- De Floriani et al. [2000]: 10 bits
- Taubin et al. [1998]: 8 bits

Representing the MT

- Encode **dependencies** by representing the DAG
- Link each modification M (node) to:
  - its direct ancestors
  - its direct descendants



- *Space complexity*
  - each arc is represented twice: 2a node references, where a = number of arcs in the DAG


Integrating Occlusion Culling with View-Dependent Rendering

Jihad El-Sana

(*with N. Sokolovsky, C. Silva*)

Department of Computer Science

Ben-Gurion University of the Negev

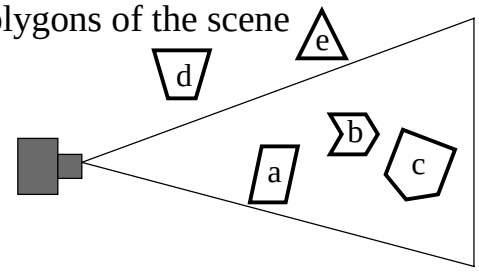
Beer-Sheva Israel 84105

Occlusion Culling

Often large fraction of the scene's polygons do not contribute to the final image because they are not visible from the current viewpoint.

A polygon may be invisible

- Out of the view frustum such as *d* and *e*
- Hidden by other polygons of the scene such as *c*



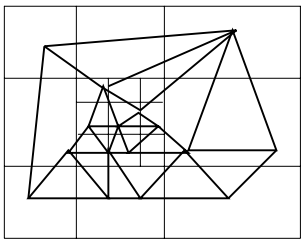
Occlusion Culling Approaches

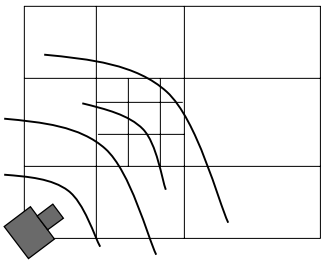
- Determine invisible polygons at each frame is a challenging problem because
  - The computation has to be complete within a limited time (1/30 sec).
  - The worst case time complexity of the problem if $O(n^2)$.
- Occlusion Culling Methods
  - Real-time determination at the rendering time
  - Estimating the set of visible triangles.
    - Usually involve an off-line processing
    - Either Conservative or non-Conservative

Prioritized-Layered Projection

Preprocessing
- Tessellation of space
- Finding list of primitives for each cell
- Computing cell's solidity

Rendering
- Projecting front cells
- Computing solidities for neighbors





Orthogonal Reduction

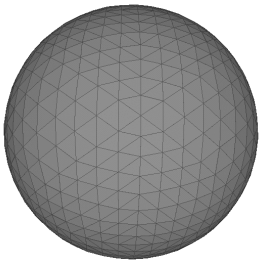
- View-Dependent Rendering reduces resolution based on view-parameters
- Occlusion Culling removes hidden polygons

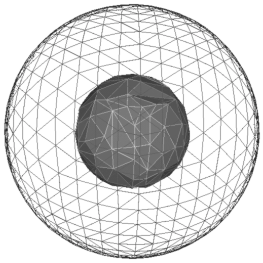
Still

- View-Dependent Rendering renders occluded triangles
- Occlusion Culling renders small triangles

View-Dependent + Visibility

Combining of view-dependent rendering with occlusion culling



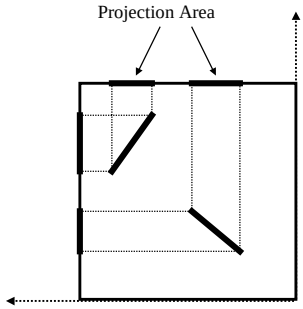


Preprocessing

- Impose 3D grid over a dataset
- Determine the set of polygons in each cell
- Assign a solidity value for each cell [0.0 – 1.0] based on two approaches
  - Face projection
  - Ray shooting

Face Projection

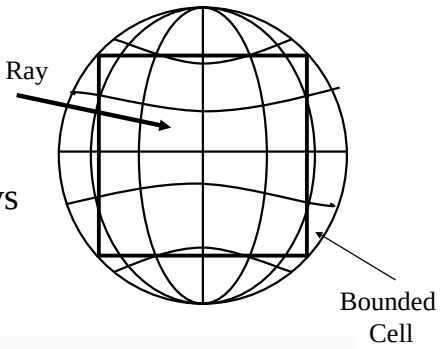
- Projection polygons on faces of the cell
- Ratio between projection areas and total area of faces



$$solidity(c_i) = \frac{\sum_{t \in poly(c_i)} ProjectionArea(t)}{\sum_{f \in face(c_i)} Area(f)}$$

Ray Shooting

- Shoot rays into a cell
- Determine which ray manage to leave the cell
- Ratio between passed rays and total number of rays



$$solidity\ \ (c_i) = 1.0 - \frac{Passed\ \ Rays}{Shot\ \ Rays}$$

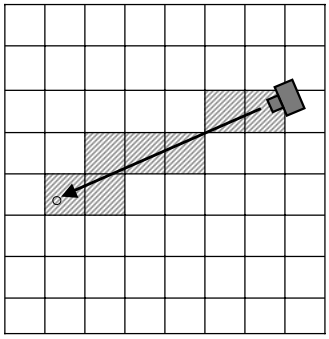
Run-Time Processing

- View-dependent rendering reduces the resolution of far-from-viewer regions.
- Hidden and close polygons are still in high resolution.
- Occlusion probability helps to recognize invisible regions.
- Reduce the selected resolution for regions with respect to their occlusion probability.

Occlusion Probability

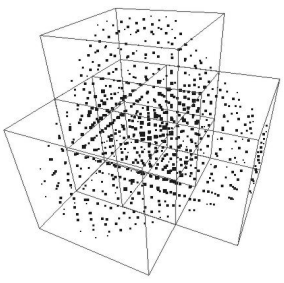
Occlusion probability (OP) depends on solidity of cells.

- From viewpoint to vertex
- Initialized OP to 0.0
- Accumulate the visited-cells' solidity
- Stop at vertex or when OP reaches 1.0

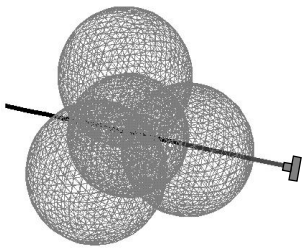


Occlusion Probability for Spheres

Four spheres dataset



Solidity values by point size



Occlusion probability by color

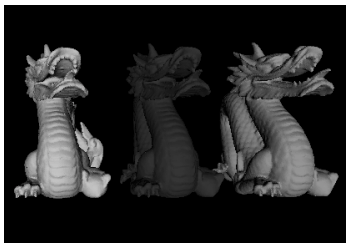
Adaptive Level-of-Detail

- View-parameters of the node
- Occlusion probability
- Adding the visibility as parameter in selecting level-of-detail

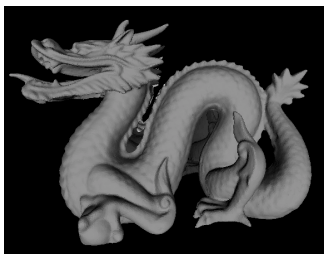
$$LOD_{final} = (1 - OP) * LOD_{view} + OP * LOD_{lowest}$$

LOD – level of detail
OP – occlusion probability

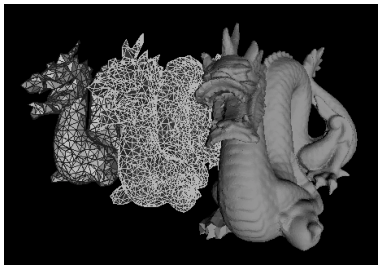
Results



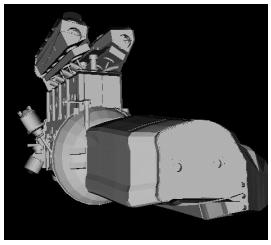
650K triangles



93K triangles

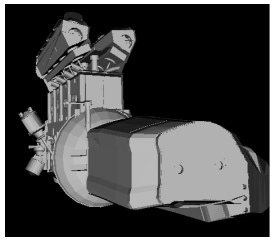


Results



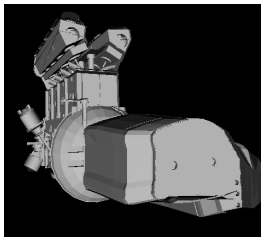
140K triangles
100%

Full resolution model



86K triangles
61%

View-dependent rendering



24K triangles
17 %

View-dependent + occlusion culling

Results

Dataset	Average Triangles/frame		Average Time(ms)/frame	
	Original	VDR + OC	VDR	VDR + OC
Bunny	75K	30K	55	39
Car engine	140K	52K	93	51
Terrain	522K	71K	110	61
Dragon team	650K	74K	115	64
Section01A	748K	81K	120	75

THE INTERDISCIPLINARY CENTER, HERZLIYA
Technical Report CS-TR-2001-02

Temporal and spatial level of details for dynamic meshes

Ariel Shamir
School of Computer Science
The Interdisciplinary Center
46150 Herzliya, Israel
E-mail: arik@idc.ac.il

Valerio Pascucci
Center for Applied Scientific Computing
Laurence Livermore National Laboratory
E-mail: pascucci@llnl.gov

April 7, 2001

Abstract

Multi-resolution techniques enhance the abilities of visualization systems to overcome limitations in time, space and transmission costs. Numerous techniques have been presented which concentrate on creating level of detail models for static meshes. Time-dependent deformable meshes impose even greater difficulties on such systems. In this paper we describe a solution for using level of details for time dependent meshes. Our solution allows for both temporal and spatial level of details to be combined in an efficient manner. By separating low and high frequency temporal information, we gain the ability to create very fast coarse updates in the temporal dimension, which can be adaptively refined for greater details.

1 Introduction

Complex time-dependent meshes are becoming more frequent in animation sequences and arise also in many simulation processes. These types of meshes are often viewed as a sequence of static meshes and as such, impose greater demands on visualization systems in terms of rendering time and storage space. Multi-resolution techniques of static meshes have been widely used and studied as a means for overcoming time, storage and transmission restrictions. Many decimation and refinement techniques were developed and many spatial metrics defined for governing the quality and level of detail of multi-resolution static models [7, 15, 13, 10].

In this paper we introduce a multi resolution model for dynamic geometry sequence of meshes, which enables the combination of both *spatial and temporal* level of details to be employed. The key observation is that mesh modifications over time can be separated to ”low frequency” global affine transformations and ”high frequency” local vertex deformations (Figure 1). The low frequency information captures the most visually significant temporal displacements using a very coarse and inexpensive approximation. This approximation can in turn be refined adaptively using the high frequency information as needed to create higher resolutions and greater details in both time and space dimensions. Our model combines three different adaptation strategies: applying the low frequency temporal deformations over time, applying the high frequency temporal deformations, and applying spatial adaptation of level of details. This gives a visualization system the flexibility to comply with a wide range of timing restrictions.

The basis of our model is the TDAG structure (see [17] and Section 2) defined for supporting multi-resolution time dependent meshes. However, instead of encoding the original series of time dependent

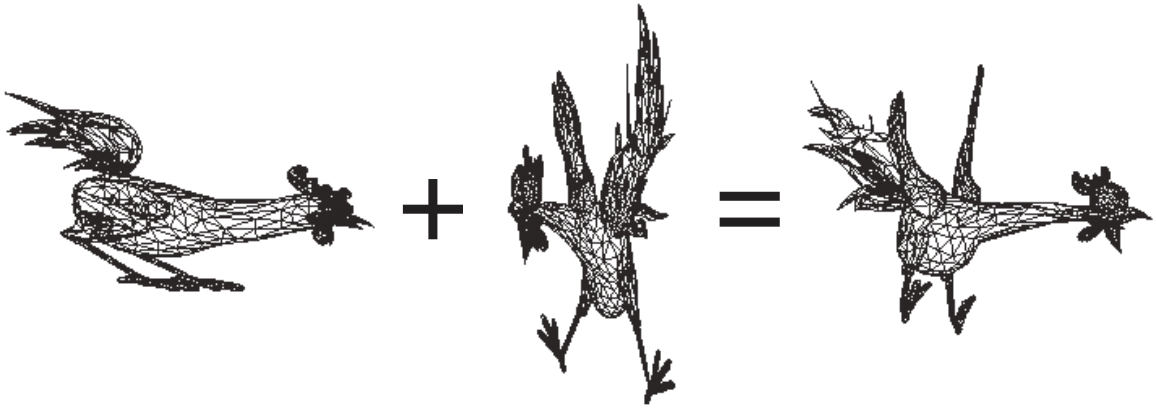


Figure 1: Separating low frequency (left) and high frequency (middle) deformations. This 'equation' is simply an illustration of the concept. The real calculation involves the multiplication of the vertices of the high frequency mesh (middle) by an affine matrix (which effect is symbolized by the left mesh).

meshes in the TDAG, we first extract the low frequency deformations of the meshes, and encode only the resulting residual meshes. The TDAG stores all attributes and positional changes of the residual meshes over time (the high frequency temporal deformations), and enables the extraction of different and adaptive resolution meshes for each time-step (spatial adaptation).

1.1 Previous Work

It is far beyond the scope of this paper to describe the many approaches that have been developed for creating multi-resolution representations of static geometric data for graphics and visualization [7, 15, 13, 10]. In this paper we use edge contraction [8, 9] as the decimation primitive and the Quadric error metrics [5]. Other multi resolution schemes include vertex removal [1], triangle contraction [6], vertex clustering [16], and wavelet analysis [19, 2]. Similar to [4, 13, 1, 6] we organize the levels of detail structure as a DAG (Directed Acyclic Graph). Each node in the DAG represents a decimation operation and the edges represent dependencies between the different operations, which impose a certain partial order for applying them on the mesh. Every **cut** (informally, a cut is a group of edges that include one and only one edge from each path from the roots to the leaves) in this DAG defines a valid adaptive level of detail of the underlying mesh. To date, most of these schemes are based on the assumption that the finest resolution mesh is static. Our scheme aims to define a level of detail in both time and space for multi-resolution dynamic meshes.

Time dependent data structures which include hierarchical decompositions are presented in [20] for the extraction of iso-surfaces from dynamic volumetric data, and in [18] for volume rendering. These structures allows very efficient time-dependent iso-surface extraction and volume rendering respectively, but are tailored for these specific types of visualization primitives and do not deal with general dynamic deformable meshes.

Earlier in [3], a model for multi-resolution video was presented. Temporal as well as spatial level of details were possible by using a binary time tree where each node corresponds to some spatial averaging of all the images of its time span. The node holds a spatial quad-tree built from this average image. This structure supports multi-resolution in the temporal dimension by accessing the average images, and seems very appropriate for video sequences. Our approach covers more general meshes (considering images can be viewed as planar meshes) and treats the temporal dimension a little differently: instead of averaging the meshes over time spans, our temporal resolution defines the length of the intervals between

each time sample.

Recently efforts on behalf of the MPEG4 standard organization [14] defined specific interpolators or behaviors for human figures or faces as well as rigid body transformations to efficiently encode dynamic meshes. Dealing with general dynamic meshes as in this paper was postponed for later dates. Moreover, 3D geometry compression either concentrate on static meshes [21, 22], or assume low geometry bandwidth [12, 23]. In [11] a method is proposed for compression of time dependent geometry. The vertex positions matrix is decomposed into $P \cdot V \cdot G$, where P is the time interpolation, V is the vertex positions at key time-steps and G is the geometry or spatial interpolation. The gross movement of the geometric mesh is extracted from V and encoded with a small set of controls. By separating the low and high frequency temporal information the residual magnitudes are reduced. The gross movement is encoded using affine transformations and the residuals are quantized at low bit rate. Our method uses a similar approach for separating low and high frequency temporal information, but uses them at two different levels of temporal details.

The rest of this paper is structured as follows: in Section 2 we briefly describe the TDAG structure, its construction and supported queries. Separating low and high frequency temporal information is described in Section 3, and the construction of the multi-resolution model in Section 4. Section 5 discusses the possible uses of the model and Sections 6 and 7 show an example and outline future work.

2 The TDAG Structure

This paper focuses on triangular meshes. A triangular mesh $\mathcal{M}$ is a tuple $\mathcal{M} = (P, F, I)$ of vertices $P = \{p_i\}$ in E^3, faces $F \in P \times P \times P$, and some vertex attributes I such as position, normal and color. We consider a time-sequence of meshes:

$$\mathcal{M}_{t_0}, \mathcal{M}_{t_1}, \ldots, \mathcal{M}_{t_k}$$

where $t_0 < t_1 < \ldots < t_k$. All mesh components, i.e. attributes, positions and adjacency, become a function of the time t_i:

$$\mathcal{M}_{t_i} = (P_{t_i}, F_{t_i}, I_{t_i})$$

. The actual time values are irrelevant, and so we normalize the time-steps to unitary intervals $\{t_i\} \rightarrow i$.

We classify the modifications between two consecutive meshes $\mathcal{M}_i$ and $\mathcal{M}_{i+1}$ into four levels: attribute changes (e.g. change in vertex color), position changes (e.g. change in vertex positions), connectivity changes (changes to the set F), and topological changes (e.g. the genus of the mesh changes).

The TDAG (Temporal Directed Acyclic Graph) is a multi-resolution data structure, which uses time-tags for all time dependent information. In particular, it deals with the symbolic information such as mesh connectivity and decimation dependencies in a similar manner as the numeric information including the attributes and positions of vertices (see Figures 2). The TDAG can encode a large class of dynamic models, which include connectivity and topology changes.

The construction of the TDAG is done incrementally using an online algorithm. When each new mesh $\mathcal{M}_{i+1}$ in the sequence is presented, it is merged into an existing TDAG which encodes the meshes $\mathcal{M}_0, \ldots, \mathcal{M}_i$. This is done by decimating $\mathcal{M}_{i+1}$ (for example, by edge contraction) using a metric function which combines current spatial constraints (such as quadric error metric) with global temporal ones. The hierarchy and attributes of the new time-step are stored as time tagged fields of the nodes of the TDAG. The local part of the metric function optimizes the structure of the level of detail DAG for the current mesh, while the global constraints are targeted at preserving the structure of the DAG as much as possible over time. More details can be found in [17].

As a model for the dynamic meshes $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_k$, the TDAG is parametric in two dimensions: resolution and time. Therefore, every valid TDAG query must include these two parameters. There are

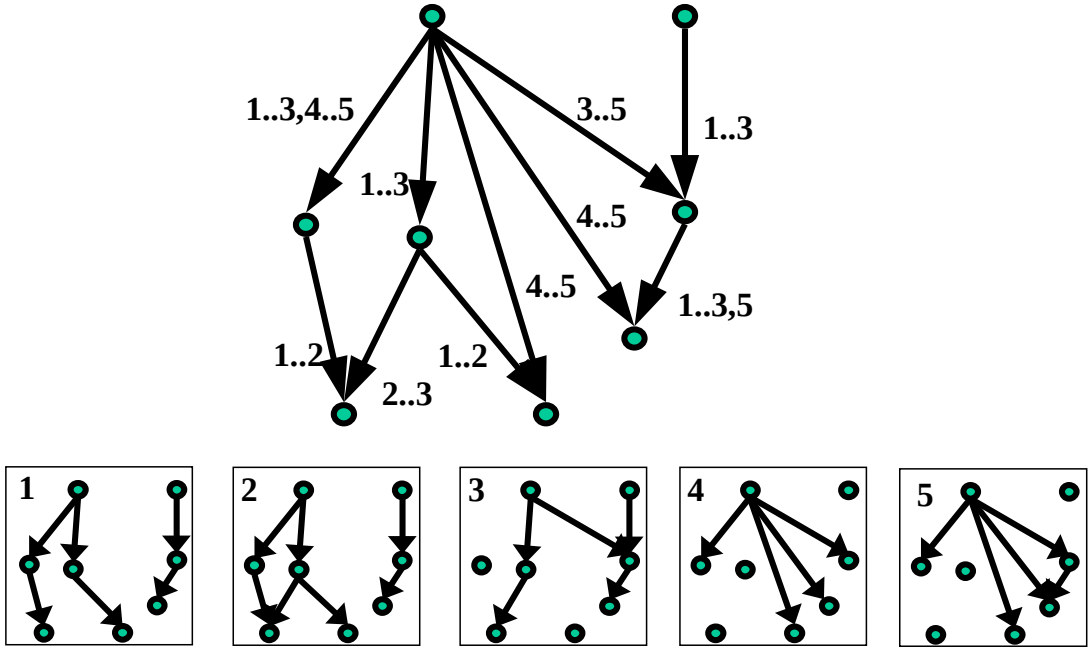


Figure 2: The child links in a T-DAG structure with five time-steps (top), and the five DAGs it represents (bottom). Each child link edge in the top T-DAG carries a tag depicting the range of time-steps in which it is active.

two types of fundamental queries supported by the TDAG: the random type queries and the incremental type queries. Given *(time = t,tol = ϵ_1)*, the random query returns an approximation $\mathcal{M}_t^{\epsilon_1}$ of the mesh $\mathcal{M}_t$ which does not differ from $\mathcal{M}_t$ by more than ϵ_1 under some given error metric. Incremental query in time (resolution) will return $\mathcal{M}_{t+1}^{\epsilon_1}$ ($\mathcal{M}_t^{\epsilon_2}$ when $\epsilon_2 \neq \epsilon_1$).

Although the model described in this paper can support all query types, we will concentrate our discussion on progressive solutions to the incremental time query. In such cases, the level of detail mesh of current time-step is progressively updated to arrive at the next time-step level of detail mesh.

3 Separating Low and High Frequency Temporal Deformations

If we track the trajectories of each vertex in the time sequence meshes $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_k$, it could be difficult to distinguish between global mesh movements (such as rigid body transformations) and local fluctuations or deformations. In order to separate between these low and high frequency deformations we want to map all meshes to a uniform general position in space. We collect all n vectors of homogeneous coordinates of the vertices of mesh $\mathcal{M}_t$ into a global $4 \times n$ matrix V_t. Similar to [11], we select either the first time-step matrix V_0 or an average mesh matrix (created by the average vertex positions of all meshes over time) and search for A_t, a 4×4 affine matrix which is the best least square solution to the equation:

$$A_t V_0 = V_t$$

In practice, we solve the following equation and find A_t for each time-step t:

$$A_t = V_t V_0^T (V_0 V_0^T)^{-1}$$

Gathering, for all t, the affine coefficients of A_t defines the low frequency temporal information of the sequence $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_k$. Applying A_t on V_0 and using the connectivity defined by $\mathcal{M}_0$ gives a

first crude approximation for the mesh sequence (see Figure 5). The residual meshes $R\mathcal{M}_t$ are found by applying A_t^{-1} on each V_t, mapping each mesh $\mathcal{M}_t$ to the uniform position of $\mathcal{M}_0$ (see Figure 6).

When topology or connectivity changes occur over time, we must calculate the affine map by using a sub-set of vertices valid in both V_t and V_0. However, as will be described in Section 4, dynamic cut update is more difficult to implement in these situations, and often the mesh will need to be reconstructed from the roots of the DAG when a cut update is requested.

4 Hierarchy Construction

The multi-resolution TDAG imposes no restrictions on mesh deformations over time in terms of connectivity and topology changes. However, when the sequence of meshes do not change their connectivity over time steps. In of an eas updating nodes of

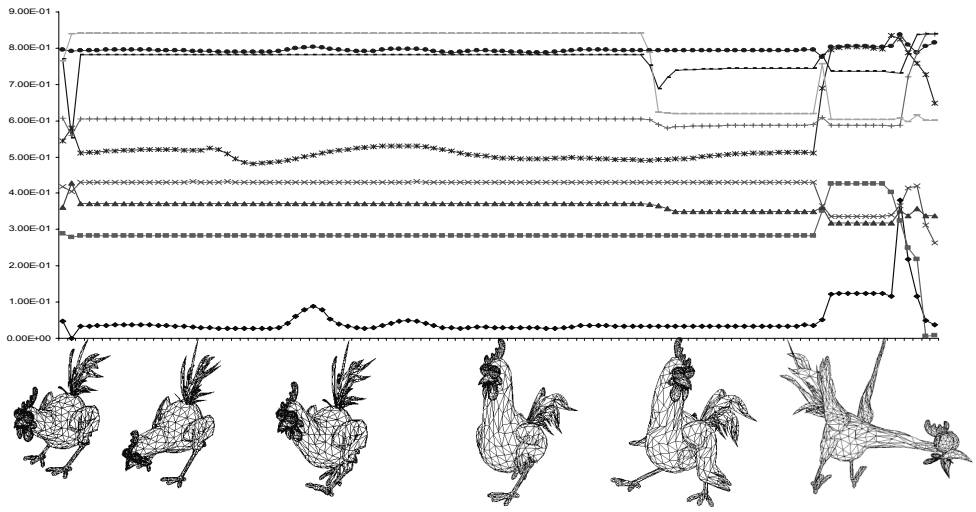


Figure 3: The decimation cost of different edges as a function of time: the x axis is time and the y axis is cost. Each plot series represents the cost of contracting one edge at different time steps. As can be seen, the cost can rise or drop depending on the local geometry of the mesh when the contraction is performed (the bottom line shows some snapshots of the mesh closely bellow their position in time). Edges that are part of the main body of the chicken tend to have smaller and more stable costs, while the cost of edges that are part of the wing or neck have a large jump in cost at the time of the surprised chicken.

In order to store all time dependent meshes using the same DAG we need to use the exact same sequence of decimation operation for all time-steps. Note that this does not mean that the multi-resolution model will be constant over time. As can be seen in Figure 3, the same decimation operation may have different costs in different time-steps. The reason for this is that the mesh attributes, which effect the cost of decimation, change over time. This also means that for the same tolerance in different time-steps we might get different cuts of the DAG, and there is a need for updating the cut over time even if the tolerance is constant.

In an online scenario, we start constructing the TDAG by decimating $\mathcal{M}_0$. If we choose to use the same DAG for all time-steps, then this decimation determines the decimation order for all consequent

time-steps. We can try and minimize the amount of cost changes over time if we abandon the online requirement for creating the TDAG. However, finding the set and order of the decimation operations that will minimize the cost change and the total cost over all time-steps is a very complex and costly optimization problem.

As a measure for the quality of decimation on all time-steps, we examine the maximum cost reached over time for every specific edge contracted. We then measure the average and median of those maximums over all edges. If E is the number of edges in the mesh, let n_i be the number of times edge i was used (contracted in one time-step mesh) during the construction of the TDAG. We define $N = \sum_{i=1}^{E} n_i$. Let c_{ij} be the cost of contracting edge i at time-step j. We define $c_{ij} = 0$ if edge i wasn't contracted at time j. Our average quality measure will then be:

$$\frac{\sum_{i=1}^{E} \max_j(c_{ij})}{N}$$

In our example (see Section 6) we used edge contraction and the quadric error metric [5] and chose to compare between choosing the first mesh, an "average" mesh (by averaging vertex positions over all time-steps) and an arbitrary mesh (number 277) to govern the decimation of more than 300 time-steps. We also measured the optimal decimation, where each mesh is decimated separately. The result of decimating the chicken sequence using an "average" mesh was not significantly better than decimating according to the first or to an arbitrary chosen mesh. In all of these cases the average quality measure doubled compared to the optimal decimation (see Figure 4). This stems from the fact that the high frequency deformation of this sequence is large (see the figures in this paper) and the "average" mesh is as far from the other meshes as any arbitrary mesh.

In this case (and probably also in many dynamic cases), not much is gained by using an average mesh to justify the abandonment of the online algorithm. Moreover, if we examine the median measure in Figure 4 it is much closer to the optimal case. This means that most of the costs of decimations are actually not much different from the optimal case. The large difference in the average measure probably comes from higher levels in the hierarchy of the DAG, which are used in cases where quality is much less a factor (e.g. when the mesh is very far from the viewer or for back faces).

Governing Mesh	No. of decimations	Total decimations	Average Measure	Median Measure
First	2884	868084	2.14	0.666
Arbitrary	2890	869890	1.90	0.650
Average	2885	868385	2.19	0.687
Optimal	-	865480	1.09	0.629

Figure 4: A comparison between different strategies for governing the creation of the TDAG. In the first three lines all meshes were decimated using the same edges and the same order which is governed either by the first, the average or an arbitrarily chosen mesh. In the last line each time-step mesh is decimated separately optimally. As can be seen, there is not much difference in choosing a single mesh to govern the decimation, in all cases the average measure is about doubled with respect to the optimal. Therefore the first mesh is as good as any other mesh to be chosen in an online algorithm.

It is important to stress that the cost of decimation should be calculated on the actual meshes $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_k$, and not on the residual meshes $R\mathcal{M}_t$. This is because the quality of approximation should be governed by the real mesh that will be displayed and not the residual.

Before discussing the possible usage of our model we summarize the online construction of the time dependent model. A mesh $\mathcal{M}_t$ from the sequence of meshes $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_k$ is encoded in the

following way: first, we extract and store A_t, the affine transformation with respect to the first mesh. We then create the residual mesh $R\mathcal{M}_t$ by applying A_t^{-1} to $\mathcal{M}_t$. Then, we encode the positional and attribute changes of $R\mathcal{M}_t$ from $R\mathcal{M}_{t-1}$ in the TDAG structure.

5 Level of Detail Utilization

Although level-of-detail models can be utilized to reduce the amount of time-dependent updates (by restricting them only to *active* vertices that are part of the current level of detail mesh), one must remember that dealing with time dependent meshes means the update of nodes attributes (position, color etc.) must be done continually through time for correct rendering.

As stated earlier, we focus on incremental queries in time: given a specific tolerance ϵ governing the overall level of detail, and given a level of detail mesh for the current time step mesh $\mathcal{M}_t$, we would like to create the next time-step level of detail mesh approximating $\mathcal{M}_{t+1}$. The complete procedure for creating the new mesh approximation involve three stages in the following order:

1. Updating the cut of $R\mathcal{M}_{t+1}$ according to the costs at $t+1$ (and possibly also a change in other parameters such as the direction of, and distance from the viewpoint).

2. Updating the attributes and positions of the active vertices in the resulting approximation residual mesh.

3. Applying the affine transformation A_{t+1}^{-1} on the resulting residual mesh.

Examining these three stages, it is evident that they are also ranked by their complexity. The first stage in creating the next time-step mesh is the most complex and time consuming, and the last is the most simple and rapid. However, in terms of their visual significance, the last stage is the most significant and the first stage the least.

The key idea in utilizing our multi-resolution model is to reverse the order of execution of these stages. This is feasible since those three stages can be carried out almost independently, resulting in three levels of temporal approximations for the mesh:

1. The coarsest approximation is created using the previous time step spatial level of detail mesh by applying the new time-step affine transformation. Note that such operation can sometimes be as simple as calling `glPushMatrix()` and `glMultMatrix()` in OpenGL.

2. A better approximation is created using the previous time-step level of detail mesh, applying the affine transformation, and also updating the attribute and position of the active vertices that are shared by both time-steps. Hence, both the low and the high frequency deformations are applied, but using the previous time-step level-of-detail mesh instead of the current.

3. The best approximation is created by carrying out all the stages of mesh creation: updating the cut, updating the attributes of active vertices and applying the transformation.

The relative independence of the three mesh update stages can also be utilized in a multi-threaded environment. The rendering thread, which reads the mesh and sends it down the graphics pipeline, can be responsible for the low-cost operation such as the affine transformations. The mesh creation thread, which writes the mesh, can be responsible for the cut update. This scheme resembles the internal double-buffer scheme of the graphics pipeline, using two working meshes - one for reading and one for writing and swapping between them. Similar to the frame buffer, if the writing thread lags behind in updating the cut, there will be no swapping of meshes. When a specific frame rate is required the writing thread

can be tuned in advance to skip some time-steps and calculate the cut for every other time-step, when the reading thread fills the gaps with the affine transformations. Since the visual significant part of the temporal update will be carried out, the unpleasant artifacts such as jumps in position, orientation and scale will be reduced. The usual artifacts created by using multi-resolution models will still need to be addressed. The implementation of such multi-threaded rendering scheme is still under way and remains as future work.

6 Results

The chicken sequence of meshes display a chicken character crossing a road realizing a truck is heading towards it, turning and trying to escape. The sequence includes 400 time-steps of 3030 vertices and 5664 triangles each. An uncompressed binary representation of the mesh takes around 14 megabytes and the TDAG and affine maps around 37.5 megabytes. On a 500 Mhz Intel Pentium 3 with 128 Mb of RAM, it takes around 0.3 second to calculate the affine map of each time-step and create the residual mesh and an average of 30 seconds to merge a new mesh into the TDAG.

Figure 5 presents samples of the original sequence of meshes (top) and the low frequency temporal information represented by applying the affine maps of each time step to the first mesh (bottom). Note how the general structure of the mesh remains more or less the same while the affine transformations change the position, orientation and stretch of the object. Figure 6 presents the sequence of meshes (top) and the high frequency temporal information, which are the residual meshes created by applying the inverse affine transformations of each time-step to the original mesh. Note in contrast to Figure 5 how the position and orientation remain constant, while the local and internal deformations of the meshes are revealed.

Lastly, Figure 7 shows examples of the results of the three levels of approximations for the mesh: applying only the affine maps (top row), applying the affine map and the attributes update (second from top), and then also updating the cut (the bottom three rows show level of detail created by three different tolerances)

7 Summary

This paper presented a scheme for creating a level-of-detail model for time dependent meshes, which allows the utilization of both temporal and spatial level-of-detail approximations. This is made possible by a specific decomposition of the temporal information into low and high frequency deformations and the usage of a multi-resolution model for the spatial information.

As discussed in Section 5 the advantages of using this scheme can be exploited in a multi-threaded environment. The first item in future extensions is the creation of a ”double meshed” multi-threaded rendering system for time-dependent meshes. Other directions include lowering the size of the TDAG file representation by using fewer dependencies in the DAG, and exploiting the high temporal coherency in the TDAG for the use of some compression mechanism.

8 Acknowledgements

The chicken character is © Copyright 1996, Microsoft Corporation. It was created by Andrew Glassner, Tom McClure, Scott Benza, and Mark Van Langeveld.

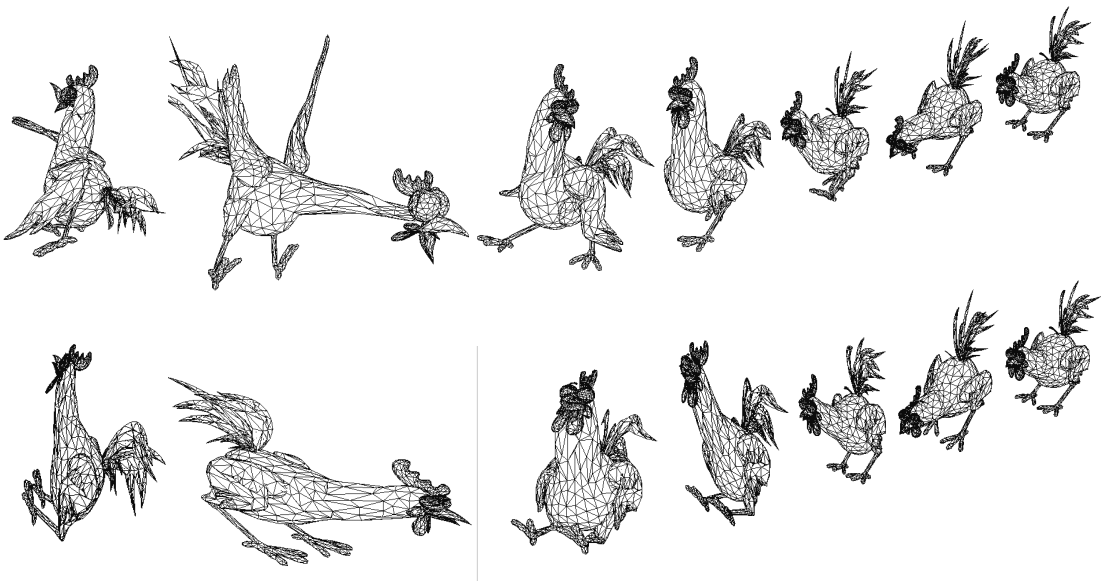


Figure 5: The low frequency deformation information of the chicken sequence (top) is applied to the first time-step mesh, creating the bottom sequence. Note that the mesh itself does not deform much, but its position, orientation and size (affine attributes) adhere to those of the original sequence.

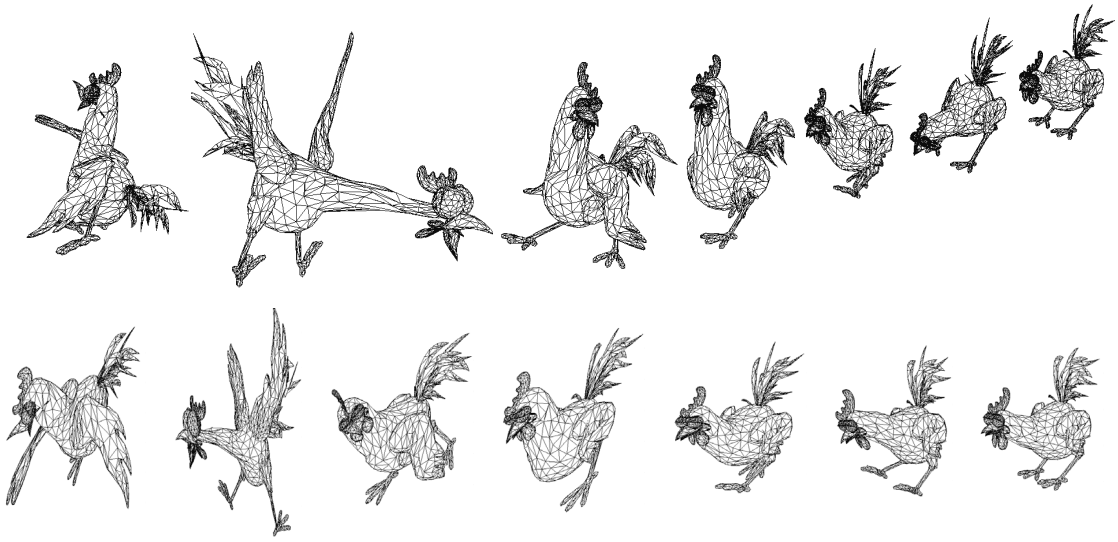


Figure 6: The application of the inverse of the affine transformation matrix to all meshes maps them to a general position, size and orientation in space. Only the high frequency deformation information of the original mesh sequence (top) changes in the residual meshes (bottom).

Figure 7: Examples of the three approximation levels for a multi-resolution dynamic mesh. The top row shows the coarsest level by applying just the affine transformations to the first time-step level of detail mesh. The second row shows the middle approximation by applying also the attributes update on the first mesh. The bottom three rows are created using the full update procedure for each time-steps, but using a different tolerance in each row. Note, for instance the differences in the triangles around the neck of the chicken in the different update schemes: in the coarsest and middle approximation (top two rows), the set of triangles does not change. In the third row in particular, the update of the cuts changes the set of triangles.

References

[1] M. de Berg and K. T. G. Dobrindt. On levels of detail in terrains. *Graphical Models and Image Processing*, 60:1–12, 1998.

[2] M. Eck, T. DeRose, T. Duchamp, T. Hoppe, H. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *ACM Computer Graphics Proceedings, SIGGRAPH'95*, Annual Conference Series, pages 173–180, 1995.

[3] A. Finkelstein, C. E. Jacobs, and D. H. Salesin. Multiresolution video. In *ACM Computer Graphics Proceedings, SIGGRAPH'96*, Annual Conference Series, pages 281–290, 1996.

[4] L. De Floriani, P. Magillo, and E. Puppo. Data structures for simplicial multi-complexes. In *Proceedings Symposium on Spatial Databases*, Hong Kong, China, July 1999.

[5] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In Turner Whitted, editor, *ACM Computer Graphics Proceedings, SIGGRAPH'97*, Annual Conference Series, pages 209–216. ACM SIGGRAPH, Addison Wesley, August 1997.

[6] Tran S. Gieng, Bernd Hamann, Kenneth I. Joy, Gregory L. Schussman, and Issac J. Trotts. Constructing hierarchies for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):145–161, April 1998.

[7] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. In *ACM Computer Graphics Proceedings, Annual Conference Series, SIGGRAPH'97, Multiresolution Surface Modelling, Course Notes No. 25*, 1997.

[8] H. Hoppe. Progressive meshes. In *ACM Computer Graphics Proceedings, SIGGRAPH'96*, Annual Conference Series, pages 99–108, 1996.

[9] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings IEEE Visualization'98*, pages 35–42. IEEE Comp. Soc. Press, 1998.

[10] R. Klein and J. Kramer. Multiresolution representations for surface meshes. In *Proceedings of the SCCG*, 1997.

[11] J. E. Lengyel. Compression of time-dependent geometry. In *Proceedings of the 1999 ACM Symposium on Interactive 3D Graphics*, Atlanta, Georgia, April 1999.

[12] M. Levoy. Polygon-assisted jpeg and mpeg compression of synthetic images. In *ACM Computer Graphics Proceedings, SIGGRAPH'95*, pages 21–28, 1995.

[13] A. Maheshwari, P. Morin, and J. R. Sack. Progressive tins: Algorithms and applications. In *Proceedings 5th ACM workshop on Advances in geographic information systems*, Las Vegas, 1997.

[14] INTERNATIONAL ORGANISATION FOR STANDARDISATION CODING OF MOVING PICTURES and AUDIO ISO/IEC JTC1/SC29/WG11 N2995. *MPEG4 standard specifications*, http://drogo.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm edition.

[15] J. Rossignac and P. Borrel. Multi-resolution 3d approximation for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Geometric Modeling in Computer Graphics*, pages 455–465. Springer Verlag, 1993.

[16] William J. Schroeder. A topology modifying progressive decimation algorithm. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization ́97*, pages 205–212. IEEE, November 1997.

[17] A. Shamir, V. Pascucci, and C. Bajaj. Multi-resolution dynamic meshes with arbitrary deformation. In *Procceedings of the IEEE Visualization Conference VIS'00*, pages 423–430, 2000.

[18] H. Shen, L. Chiang, and K. Ma. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (tsp) tree. In *Proceedings of the IEEE Visualization Conference VIS'99*, pages 371–378, 1999.

[19] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann Publishers, 1996.

[20] P. M. Sutton and C. D. Hansen. Isosurface extraction in time-varying fields using a temporal branch-on-need tree (t-bon). In *Proceedings of the IEEE Visualization Conference VIS'99*, pages 147–154, 1999.

[21] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus. Progressive forest split compression. In *ACM Computer Graphics Proceedings, SIGGRAPH'98*, Annual Conference Series, pages 123–132, 1998.

[22] C. Touma and C. Gotsman. Triangle mesh compression. In *Proceedings of Graphics Interface '98*, pages 26–34, 1998.

[23] D. S. Wallach, S. Kunapalli, and M. F. Cohen. Accelerated mpeg compression of dynamic polygonal scenes. In *ACM Computer Graphics Proceedings, SIGGRAPH'94*, pages 193–197, 1994.

List of references for the tutorial:

View-Dependent Rendering for Polygonal Datasets

- Introduction
- The Multi-Triangulation: A Framework for Multiresolution Triangle Meshes
- Generalized View-Dependent Rendering
- Client/Server View-Dependent Rendering
- Integrating Occlusion Culling with View-Dependent Rendering

References

[1] M.K. Agoston. Algebraic Topology: a First Course. *Pure and Applied Mathematics*, M. Dekker (ed.), New York, 1976.

[2] J. M. Airey, J. H. Rohlf, and F. P. Brooks, Jr. Towards image realism with interactive update rates in complex virtual building environments. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, volume 24, No. 2, 41–50, March 1990.

[3] C. Andújar, C. Saona-Vázquez, I. Navazo, and P. Brunet. Integrating occlusion culling and levels of detail through hardly-visible sets. *Computer Graphics Forum*, 19(3):499–506, August 2000.

[4] C.L. Bajaj, F. Bernardini, J. Chen, and D.R. Schikore. Automatic reconstruction of 3D CAD models. *Theory and Practice of Geometric Modeling*, W. Straßer, R. Klein, and R. Rau (eds.), Springer-Verlag, 1996.

[5] D. Bartz, M. Meiner, and T. Huttner. Extending graphics hardware for occlusion queries in opengl. In *1998 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, 97–104, August 1998.

[6] M. Bertolotto, E. Bruzzone, L. De Floriani, and E. Puppo. Multiresolution representation of volume data through hierarchical simplicial complexes. *Aspects of Visual Form Processing*, World Scientific, Singapore, 1994, 73–82.

[7] M. Bertolotto, L. De Floriani, and P. Marzano. Pyramidal simplicial complexes. *4th International Symposium on Solid Modeling*, Salt Lake City, Utah, U.S.A., May 17-19. ACM Press, 1995, 153–162.

[8] J. Bey. Tetrahedral mesh refinement. Computing **55**, 1995, 355–378.

[9] Peter J.C. Brown. A fast algorithm for selective refinement of terrain meshes. *COMPUGRAPHICS 96*, GRASP, 1996, 70–82. A longer version is available as Technical Report No. 417, Computer Laboratory, Cambridge University, CB2 3QG, UK, February 1997.

[10] S. T. Bryson and S. Johan. Time management, simultaneity and time-critical computation in interactive unsteady visualization environments. In *Proceedings of IEEE Visualization 96*, 255–262, October 1996.

[11] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. Computer-Aided Design **10**(6), 1978, 350–355.

[12] P. Cignoni, D. Costanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral volume with accurate error evaluation. *IEEE Visualization'00*, IEEE Computer Society, 2000, 85–92.

[13] P. Cignoni, L. De Floriani, P. Magillo, E. Puppo, and R. Scopigno. Selective Refinement Queries for Volume Visualization of Unstructured Tetrahedral Meshes *To appear on IEEE Transactions on Visualization and Computer Graphics*, 2002.

[14] P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and rendering of volume data based on simplicial complexes. *1994 Symposium on Volume Visualization*, ACM Press, 1994, 19–26.

[15] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data. IEEE Transactions on Visualization and Computer Graphics **3**(4), 1997, 352–369.

[16] P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution. The Visual Computer **13**, 1997, 199–217. A preliminary version appeared on *Int. Symp. Scientific Visualization '95*, World Scientific, 1995, 50-68.

[17] D. Cohen-Or, G. Fibich, D. Halperin, and E. Zadicario. Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes. *Computer Graphics Forum*, 17(3):C243–C253, 1998.

[18] Y. Coorg and S. Teller. Real-time occlusion culling for models with large occluders. In *1997 Symposium on Interactive 3D Graphics*, 83–90, April 1997.

[19] E. Danovaro, L. De Floriani, M. Lee, H. Samet. Multiresolution Tetrahedral Meshes: an Analysis and a Comparison. *Proceedings International Conference on Shape Modeling*, Banff (Canada), May 17-22, 2002.

[20] E. Danovaro, L. De Floriani. Half-Edge Multi-Tessellation: A Compact Representation for Multi-Resolution Tetrahedral Meshes. *Proceedings 1st International Symposium on 3D data Processing Visualization Transmission*, Padova (Italy), June 19-21, 2002, to appear.

[21] E. Danovaro, L. De Floriani, P. Magillo, and E. Puppo. Compressing multiresolution triangle meshes. *7th International Symposium on Spatial and Temporal Databases, SSTD 2001*, Los Angeles, CA, USA, 2001.

[22] E. Danovaro, L. De Floriani, P. Magillo, and E. Puppo. Representing vertex-based simplicial multi-complexes. *Digital and Image Geometry*, Lecture Notes in Computer Science **2243**, G. Bertrand, A. Imiya, and R. Klette (eds.), Springer-Verlag, New York, 2001, 128–147.

[23] E. Danovaro, L. De Floriani, P. Magillo, and E. Puppo. Data structures for 3D multi-tessellations: an overview. Technical Report DISI-TR-02-01, Department of Computer and Information Science, University of Genova (Italy), 2002.

[24] F. Durand, G. Drettakis, J. Thollot, and C. Puech. Conservative visibility preprocess-ing using extended projections. In *Proceedings of SIGGRAPH 2000*, pages 239–248, July 2000.

[25] M. de Berg and K. Dobrindt. On levels of detail in terrains. *11th ACM Symposium on Computational Geometry*, Vancouver (Canada), ACM Press, 1995, C26–C27.

[26] L. De Floriani. A pyramidal data structure for triangle-based surface description. IEEE Computer Graphics and Applications **8**(2), 1989, 67–78.

[27] L. De Floriani, P. Magillo, F. Morando, and E. Puppo. Dynamic view-dependent multiresolution on a client-server architecture. CAD Journal **32**(13), 2000, 805–823.

[28] L. De Floriani, P. Magillo, and E. Puppo. Building and traversing a surface at variable resolution. *IEEE Visualization 97*, Phoenix, AZ (USA), 1997, 103–110.

[29] L. De Floriani, P. Magillo, and E. Puppo. Data structures for simplicial multi-complexes. *Advances in Spatial Databases*, Lecture Notes in Computer Science **1651**, Guting, Papadias, and Lochovsky (eds.), Springer Verlag, 1999, 33–51.

[30] L. De Floriani and P. Magillo, Multiresolution Meshes, in *Principles of Multiresolution in Geometric Modeling - PRIMUS summer school*, M.S.Floater and A.Iske and E.Quak (Editors), 2001, pp.193–234.

[31] L. De Floriani and E. Puppo. Hierarchical triangulation for multiresolution surface description. ACM Trans. on Graphics **14**(4), 1995, 363–411.

[32] L. De Floriani, E. Puppo, and P. Magillo. A formal approach to multiresolution modeling. *Geometric Modeling: Theory and Practice*, R. Klein, W. Straßer, and R. Rau (eds.), Springer-Verlag, 1997, 302–323.

[33] D.P. Dobkin and M.J. Laszlo. Primitives for the manipulation of three–dimensional subdivisions. Algorithmica **4**, 1989, 3–32.

[34] D. Doo and M. Sabin. Analysis of the behaviour of recursive subdivision surfaces near extraordinary points. Computer-Aided Design **10**(6), 1978, 356–360.

[35] M. Duchaineau, M. Wolinsky, D.E. Sigeti, M.C. Miller, C. Aldrich, and M.B. Mineed-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. *IEEE Visualization'97*, 1997, 81–88.

[36] G. Dutton. Improving locational specificity of map data - a multiresolution, metadata-driven approach and notation. International Journal of Geographic Information Systems **10**(3), 1996, 253–268.

[37] N. Dyn. Interpolatory subdivision schemes. This volume, Chapter 2.

[38] N. Dyn, D. Levin, and J.A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. ACM Trans. on Graphics **9**(2), 1990, 160–169.

[39] J. El-Sana and A. Varshney. Generalized view-dependent simplification. Computer Graphics Forum **18**(3), 1999, C83–C94.

[40] J. El-Sana. Multi-User View-Dependent Rendering *IEEE Visualization 2000*, IEEE Computer Society, 2000, 335–342.

[41] J. El-Sana. Remote View-Dependent Rendering. Immersive Projection Technology and Virtual Environments, *Springer-Verlag*, 2001,175–185.

[42] J. El-Sana, N. Sokolovsky, and C. Silva. Integrating Occlusion Culling with View-Dependent Rendering. *IEEE Visualization 2000*, IEEE Computer Society, 2001, 371–378.

[43] J. El-Sana and Y. Chiang External memory view-dependent simplification. Computer Graphics Forum **19**(3), 2000, C139–C150.

[44] J. El-Sana, F. Evans, A. Kalaiah, A. Varshney, S. Skiena, and E. Azanli. Efficiently Computing and Updating Triangle Strips for Real-Time Rendering CAD Journal **32**(13), 2000, 753– 772.

[45] W. Evans, D. Kirkpatrick, and G. Townsend. Right triangular irregular networks. Algorithmica **30**(2), 2001, 264–286. Special Issue on Algorithms for Geographical Information.

[46] T. Funkhouser, P. Min, and I. Carlbom. Real-time acoustic modeling for distributed virtual environments. In *Proceedings of SIGGRAPH 1999*, pages 365–374, August 1999.

[47] T. Gerstner. Multiresolution visualization and compression of global topographic data. GeoInformatica, to appear, 2001. Short version appeared in *Spatial Data Handling*, P. Forer, A.G.O. Yeh, J. He (eds.), IGU/GISc, 2000, 14-27.

[48] T. Gerstner and R. Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. *IEEE Visualization 2000*, IEEE Computer Society, 2000, 259–266.

[49] T. Gerstner and M. Rumpf. Multiresolutional parallel isosurface extraction based on tetrahedral bisection. *1999 Symposium on Volume Visualization*, ACM Press, 1999.

[50] T.S. Gieng, B. Hamann, K.I. Joy, G.L. Schussman, and I.J. Trotts. Constructing hierarchies of triangle meshes. IEEE Transactions on Visualization and Computer Graphics **4**(2), 1997, 145–160.

[51] E. Gobbetti and E. Bouvier. Time-critical multiresolution redering of large complex models. *Computer-Aided Design*, 32(13):785–803, 2000.

[52] D. Gomez and A. Guzman. Digital model for three-dimensional surface representation. Geo-Processing **1**, 1979, 53–70.

[53] C. Gotsman, S. Gumhold, and L. Kobbelt. Simplification and Compression of 3D Meshes. This volume, Chapter 11.

[54] N. Greene, M. Kass, and G. Miller. Hierarchical Z-buffer visibility. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, 231–240, 1993.

[55] G. Greiner and R. Grosso. Hierarchical tetrahedral-octahedral subdivision for volume visualization. The Visual Computer **16**, 2000, 357–365.

[56] M.H. Gross, R. Gatti., and O.G. Staadt. Fast multiresolution surface meshing *IEEE Visualization'95*, IEEE Computer Society, 1995, 135–142.

[57] M.H. Gross and O.G. Staadt. Progressive tetrahedralizations. *IEEE Visualization'98*, Research Triangle Park, NC, IEEE Computer Society, 1998, 397–402.

[58] M.H. Gross, O.G. Staadt, and R. Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures. IEEE Transactions on Visualization and Computer Graphics **2**(2), 1996, 130–144.

[59] R. Grosso, C. Luerig, and T. Ertl. The multilevel finite element method for adaptive mesh optimization and visualization of volume data. *IEEE Visualization '97*, Phoenix, AZ, IEEE Computer Society, 1997, 387–394.

[60] A. Guéziec, G. Taubin, F. Lazarus, and W. Horn. Simplicial maps for progressive transmission of polygonal surfaces. *ACM VRML98*, 1998, 25–31.

[61] B. Hamann. A data reduction scheme for triangulated surfaces. Comput. Aided Geom. Design **11**(2), 1994, 197–214.

[62] D.J. Hebert. Symbolic local refinement of tetrahedral grids. Journal of Symbolic Computation **17**, 1994, 457–472.

[63] D. Holliday and G. Nielson. Progressive volume model for rectilinear data using tetrahedral Coons patches. *Data Visualization*, W. de Leeuw and R. van Liere (eds.), Springer Verlag, 2000.

[64] H. Hoppe. Progressive meshes. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 1996, 99–108.

[65] H. Hoppe. View-dependent refinement of progressive meshes. *Computer Graphics Proceedings, Annual Conference Series, (SIGGRAPH)*, ACM Press, 1997, 189–198.

[66] T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff, and H. Zhang. Accelerated occlusion culling using shadow frustra. In *In Proc. 13th Annu. ACM Sympos. Comput. Geom.*, 1–10, 1997.

[67] J. Kim and S. Lee. Truly selective refinement of progressive meshes. *Graphics Interface 2001*, Ottawa, Canada, 2001, 101–110.

[68] D.G. Kirkpatrick. Optimal search in planar subdivisions. SIAM J. Comput. **12**, 1983, 28–35.

[69] R. Klein and S. Gumhold. Data compression of multiresolution surfaces. *Visualization in Scientific Computing '98*, Springer-Verlag, 1998, 13–24.

[70] R. Klein and W. Straßer. Generation of multiresolution models from CAD data for real time rendering. *Theory and Practice of Geometric Modeling (Blaubeuren II)*, R. Klein, W. Straßer, and R. Rau (eds.), Spinger-Verlag, 1997.

[71] J. T. Klosowski and C. T. Silva. The prioritized-layered projection algorithm for visible set estimation. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):108–123, 2000.

[72] L. Kobbelt. $\sqrt{3}$ subdivision. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 2000.

[73] L. Kobbelt, S. Campagna, J. Vorsatz, and H.P. Seidel. Interactive multi-resolution modeling of arbitrary meshes. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 1998.

[74] M. Lee, L. De Floriani, and H. Samet. Constant-time neighbor finding in hierarchical meshes. *International Conference on Shape Modeling*, Genova (Italy), May 7-11 2001, 286–295.

[75] M. Lee and H. Samet. Navigating through triangle meshes implemented as linear quadreees. ACM Trans. on Graphics **19**(2), 2000.

[76] P. Lindstrom, D. Koller, W. Ribarsky, L.F. Hodges, N. Faust, and G.A. Turner. Real-time, continuous level of detail rendering of height fields. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 1996, 109–118.

[77] P. Lindstrom and V. Pascucci. Visualization of large terrains made easy. *IEEE Visualization'01*, San Diego, CA, 2001, 363–370.

[78] D. Luebke. Robust view-dependent simplification for very large-scale CAD visualization. Technical Report TR CS-99-33, University of Virginia, 1999.

[79] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 1997, 199–207.

[80] P. Magillo. *Spatial Operations on Multiresolution Cell Complexes.* PhD thesis, Dept. of Computer and Information Sciences, University of Genova (Italy), 1999.

[81] J.M. Maubach. Local bisection refinement for N-simplicial grids generated by reflection. SIAM J. Comput. **16**(1), 1995, 210–227.

[82] M. Ohlberger and M. Rumpf. Hierachical and adaptive visualization on nested grids. Computing **56**, 1997, 365–385.

[83] R. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. *IEEE Visualization'98*, Research Triangle Park, NC, IEEE Computer Society, 1998, 19–26.

[84] P.Lindstrom. Out-of-core simplification of large polygonal models. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 2000, 259–270.

[85] R Pajarola. FastMesh: Efficient View-dependent Meshing. *Proceedings Pacific Graphics 2001*, 2001, 22–30.

[86] J. Popovic and H. Hoppe. Progressive simplicial complexes. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 1997, 217–224.

[87] E. Puppo. Variable resolution terrain surfaces. *Eight Canadian Conference on Computational Geometry*, Ottawa, Canada, August 12-15 1996, 202–210. Extended version appeared with title Variable Resolution Triangulations, Computational Geometry **11**(3-4), 1998, 219–238.

[88] K.J. Renze and J.H. Oliver. Generalized unstructured decimation. IEEE Computational Geometry & Applications **16**(6), 1996, 24–32.

[89] M. Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. International Journal of Numerical Engineering **20**, 1984.

[90] N. Scott, D. Olsen, and E. Gannet. An overview of the visualize fx graphics accelerator hardware. *The HewlettPackard Journal*, 28–34, May 1998.

[91] F. X. Sillion and G. Drettakis. Feature-based control of visibility error: A multi-resolution clustering algorithm for global illumination. *Proceedings of SIGGRAPH 95*, pages 145–152, 1995.

[92] A. Shamir and V. Pascucci C. Bajaj. Multi-resolution dynamic meshes with arbitrary deformation. *IEEE Visualization '2000 Proceedings*, pages 423–430, 2000.

[93] R. Southern, S. Perkins, B. Steyn, A. Muller, P. Marais, and E. Blake. A stateless client for progressive view-dependent transmission. *WEB 3D 2001*, February 2001.

[94] J. Ruppert and R. Seidel. On the difficulty of tetrahedralizing 3-dimensional non-convex polyhedra. *5nd ACM Symposium on Computational Geometry*, 1989, 380–392.

[95] M. Sabin. Subdivision of Box-Splines. This volume, Chapter 1.

[96] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS.* Addison Wesley, Reading, MA, 1990.

[97] L. Scarlatos and T. Pavlidis. Hierarchical triangulation using cartographics coherence. Computer Vision, Graphics, and Image Processing : Graphical Models and Image Processing **54**(2), 1992, 147–161.

[98] G. Schrack. Finding neighbors of equal size in linear quadtrees and octrees in constant time. Computer Vision, Graphics, and Image Processing : Image Understanding **55**(3), 1992, 221–230.

[99] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen. Decimation of triangle meshes. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 1992, 65–70.

[100] J.R. Shewchuck. Tetrahedral mesh generation by Delaunay refinement. *14th Annual Symposium on Computational Geometry*, Minneapolis, Minnesota, ACM Press, 1998, 86–95.

[101] R. Sivan and H. Samet. Algorithms for constructing quadtree surface maps. *5th International Symposium on Spatial Data Handling*, Charleston, 1992, 361–370.

[102] P. M. Sutton and C. D. Hansen. Accelerated isosurface extraction in time-varying fields. IEEE Transactions on Visualization and Computer Graphics **6**(2), 2000, 97–107.

[103] G. Taubin, A. Guéziec, W. Horn, and F. Lazarus. Progressive forest split compression. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 1998, 123–132.

[104] S. Teller and C. H. Séquin. Visibility preprocessing for interactive walkthroughs. *Computer Graphics: Proceedings of SIGGRAPH'91*, 25, No. 4:61–69, 1991.

[105] I.J. Trotts, B. Hamann, and K.I. Joy. Simplification of tetrahedral meshes with error bounds. IEEE Transactions on Visualization and Computer Graphics bf 5(3), 1999, 224–237.

[106] I.J. Trotts, B. Hamann, K.I. Joy, and D.F. Wiley. Simplification of tetrahedral meshes. *IEEE Visualization'98*, Research Triangle Park, NC, 1998, 287–295.

[107] L. Velho. Mesh simplification using four-face clusters. *International Conference on Shape Modeling*, Genova (Italy), May 7–11 2001.

[108] L. Velho, L. Henriquez de Figueredo, and J. Gomes. A unified approach for hierarchical adaptive tesselation of surfaces. ACM Trans. on Graphics **4** (18), 1999, 329–360.

[109] L. Velho and J. Gomes. Variable resolution 4-k meshes: Concepts and applications. Computer Graphics Forum **19**(4), 2000, 195–214.

[110] B. Von Herzen and A.H. Barr. Accurate triangulations of deformed, intersecting surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 1987, 103–110.

[111] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. The Visual Computer **15**, 1999, 100–111.

[112] J.C. Xia, J. El-Sana, and A. Varshney. Adaptive real-time level-of-detail-based rendering for polygonal models. IEEE Transactions on Visualization and Computer Graphics **3**(2), 1997, 171–183.

[113] H. Zhang, D. Manocha, T. Hudson, and K. Hoff III. Visibility culling using hierarchical occlusion maps. In *Proceedings of SIGGRAPH 1997*, pages 77–88, August 1997.

[114] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. *IEEE Visualization'97*, IEEE Computer Society, 1997, 135–142.

[115] D. Zorin. A method for analysi of C^1-continuity of subdivision surfaces. SIAM J. Numer. Anal. **37**(4), 2000.

[116] D. Zorin and P Schröder (eds.). Subdivisions for Modeling and Animation, *ACM SIGGRAPH 2000*, Course Notes No.23), ACM Press, July 2000.

[117] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH)*, ACM Press, 1997, 259–268.