Multi-Resolution Mesh Representations

Dwight Rodgers

Overview

- Introduction
  - What is “Multi-Resolution”and Why?
  - Standard Types of Multi-Resolution Data
- Current Technology
  - Simple Multi-Resolution Formats
  - Wavelet-Based MultiRes
  - MultiRes as a Pseudo-Scripting Language
- Directions
  - Rapid Topological Changes
  - Procedural Surface Generation as MultiRes

What is Multi-Resolution and Why?

- A Definition of MultiResolution:
  - A MultiResolution format is a model format that supports the query:

$$approx(M_{mres}, x) \Rightarrow M_{renderable}$$
$$s.t.: polys(M_{renderable}) = f(x)$$

where *f(x)* is a monotonically increasing function, dependent on *M*

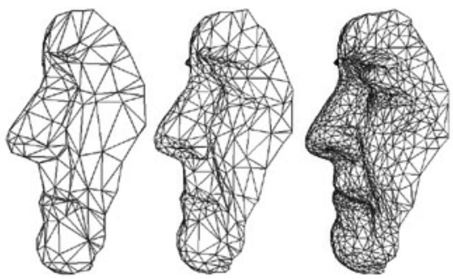
- Uses of Multi-Resolution
  - Display (Real-time reallocation of polygon counts)
  - Editing (Separate editing of course and fine detail)
  - Progressive Transmission
  - Lossy Compression

Standard Types of Multi-Resolution Data

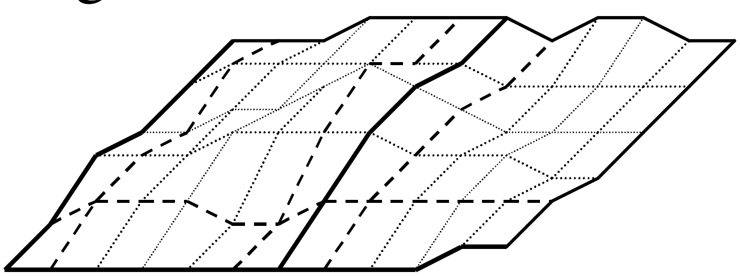
- Terrain
  - Used in Flight Simulators / Games
  - Planar geometry
  - Usually sampled on a regular grid
- Free-Standing Models
  - Planes, Boats, Tanks, Potted Plants, Computer Terminals, etc.
  - Closed polygonal meshes with few or no "handles"
  - Non-uniform sampling
  - Usually viewed from a distance
- Encompassing Models
  - Buildings, Caves, etc.
  - Very complex geometry, Lots of handles
  - Usually viewed from within

Simple Multi-Resolution Formats

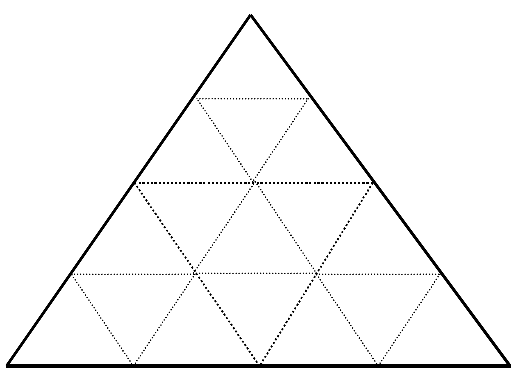
- Multiple LOD Models



- Regular Grids



- Heirarchical Triangulations



Wavelet Based Multi-Resolution

- Structure
  - Store Base Mesh + Wavelet Coefficients

- Function
  - Derive High Resolution Model by Repeatedly Subdividing Mesh
  - New Point Locations Derived from Previous Mesh Locations + Wavelet Function
  - Similar To “Face Insertion Scripting,” Except that Parameters are Waveletized

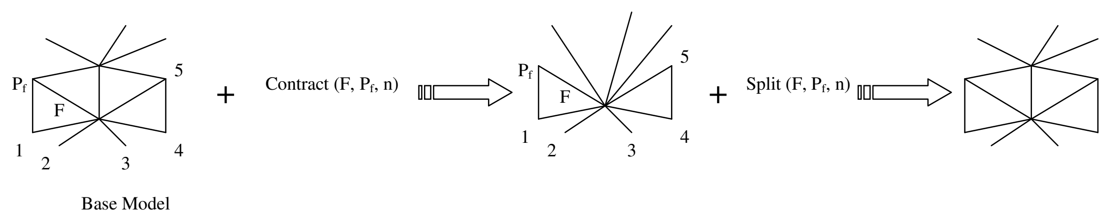
- Disadvantages
  - Very slow to compute
  - Valence of any vertex restricted to ≤ 6

Multi-Res as a Scripting Language

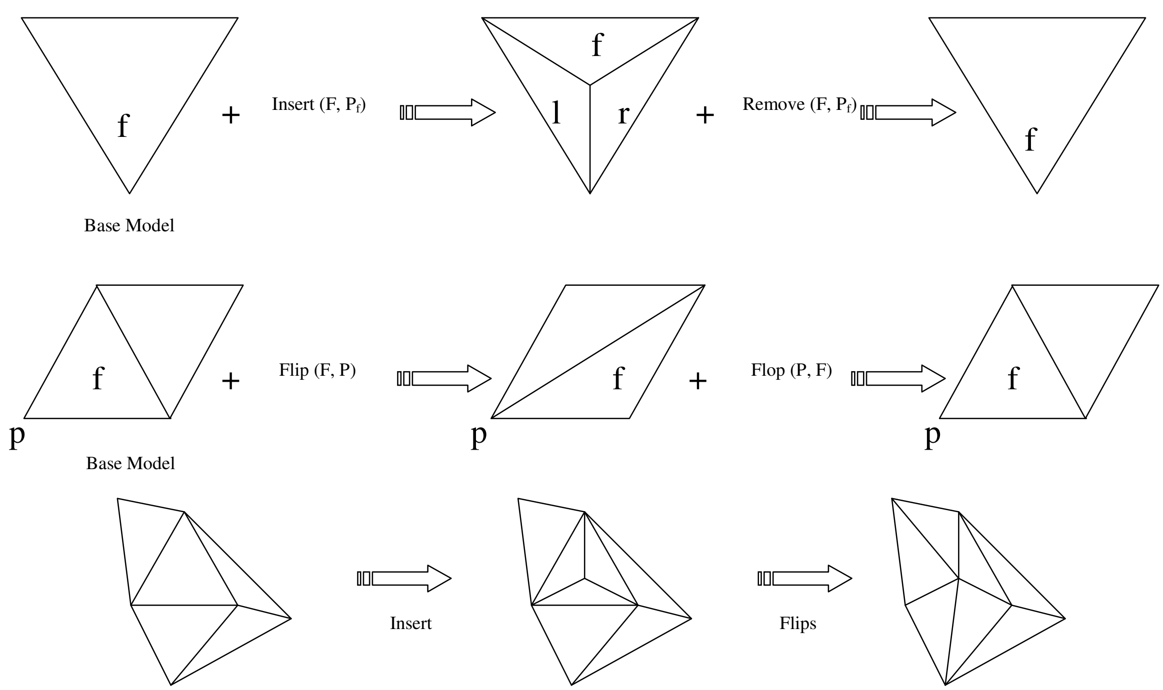
- Incremental:
  - Store Base Model + Instructions to Build
- Reductive:
  - Store Full Resolution Model + Instructions to Reduce
- Combined:
  - Store Any Model + Reversible Instructions

Instruction Sets

- Edge Contraction / Vertex Split



- Insertion / Deletion, Flip / Flop



Simple Scripting Example

- Heirarchical Triangulation
  - Base Model = Smallest mesh with correct number of handles & holes
  - Instructions = Ordered List of (Face, 3 Points) to Insert
- Advantages
  - Previous output can be passed back to query function
  - Query Function Operates in O(Delta N) time
  - Instructions easily reversible
- Foundation for Wavelet MultiRes

Edge Contraction / Vertex Splits

- Structure
  - Store Model and Interpolated Vertices
  - Store Ordered List of Vertex Pairs to Collapse
- Advantages
  - Complete basis for closed meshes
  - Allows handle removal and multi-object reduction:
    - » Vertices not explicitly joined by an edge may still be collapsed
- Disadvantages
  - Computation of vertex properties for collapsed edges
  - Collapses and Expansions are somewhat slow

Insert/Delete, Flip/Flop

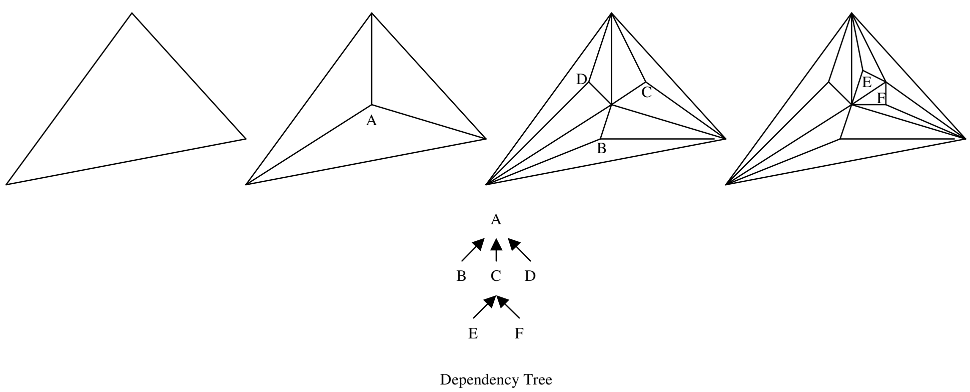
- Structure
  - Store All Vertices of Highest Resolution Model
  - Store Face-Lists of Lowest Resolution Model
  - Store List of Insertions and Flips
- Function
  - Resolution increase = insertion of new vertex from list + flips
- Advantages
  - Operations each take only a couple processor cycles
  - Data structure is very small
  - Vertices never move:
    - » Properties don’t need re-computation
    - » Vertex lists can be pre-loaded into hardware
- Disadvantages
  - Cannot merge objects or handles

Instruction Interference Graphs

- Concept:
  - Allow incremental/reductive instructions to execute out of order
- Incentive:
  - View Dependent Multi-Res
  - Localized Resolution Changes
- Problem:
  - Instruction parameters usually rely on state of model before execution
- Solution:
  - Develop an acyclic directed graph stating which instructions must be executed before which others

Examples of Interference Graphs

- Insertion/Deletion:
  - Very nonrestrictive graphs, due to locality of operations:
  - Graphs are always trees

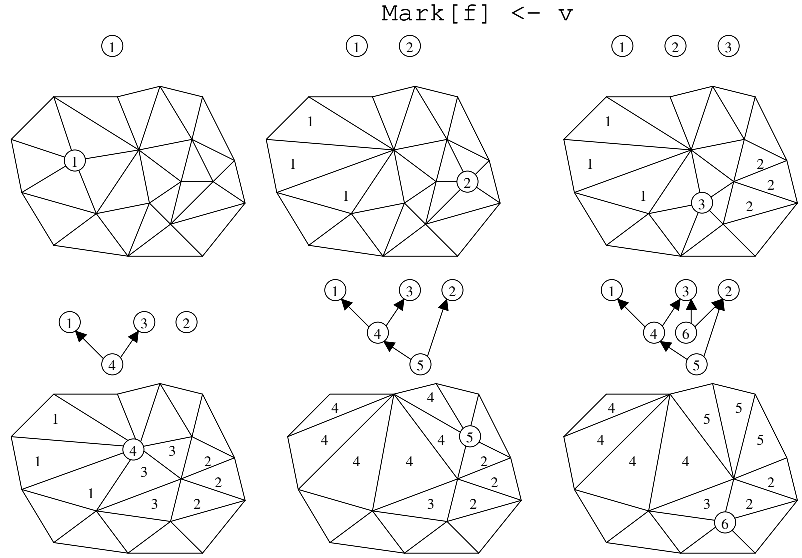


- Insertion/Deletion, Flip/Flop
  - Build Graphs Top Down
  - Results are not restricted to trees

Constructing Interference Graphs

- Psuedo-Code For Delete, Flop Instruction Set:

```
For Each f in Faces
        Mark[f] <- NULL
While not done
        v <- ChooseNextVertex
        For Each Face f s.t. f contains v
                Add Mark[f] to Parents[v]
        PerformFlips
        RemoveVertex(v)
        For Each Modified Face, f
                Mark[f] <- v
```



Evaluating Instruction Sets

- Speed of Operations
- Average # Of Operations Required from $M \mapsto M'$
- Size of Operation Parameters
- Locality of Operations
  - Flexibility of Interference Graphs
- Flexibility of Resultant Meshes

Directions

- Rapid Topological Changes:
  - Current Method:
    - » Topological changes provided by edge-contraction on non-adjacent vertices.
  - However:
    - » Instruction set is not fast
    - » Does not provide flexible interference graphs

- Procedural Surface Generation as MultiRes:
  - Example: Procedural Trees