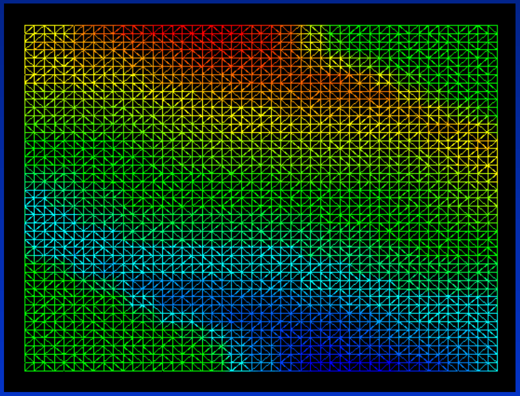
Multi-resolution Dynamic Meshes With Arbitrary Deformations

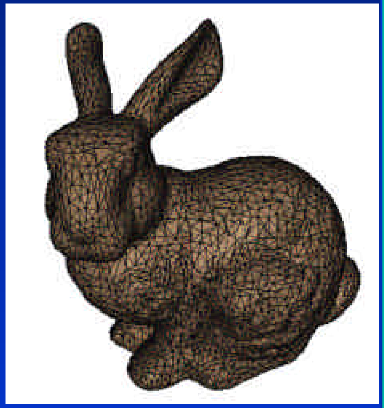
Ariel Shamir, V. Pascucci, C. Bajaj

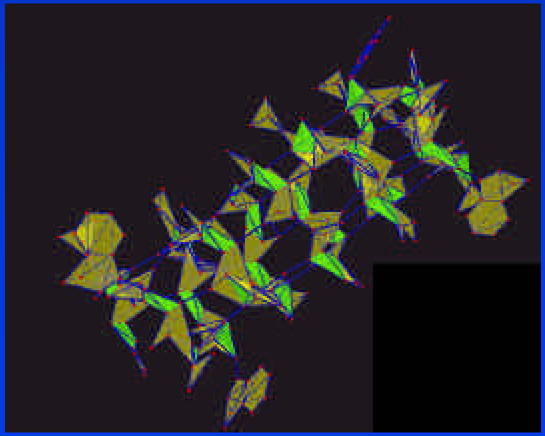
Center for Computational Visualization

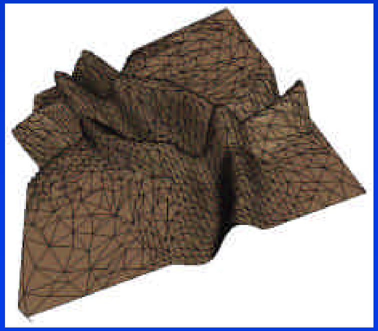
University of Austin, Texas

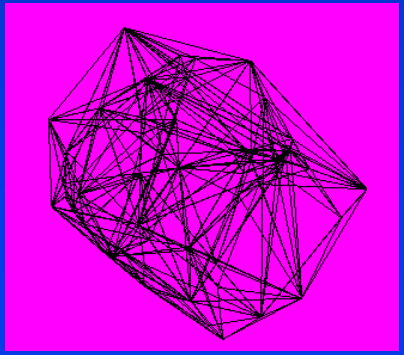
Meshes





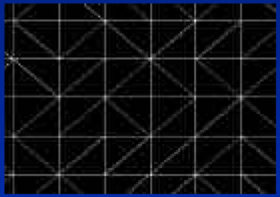






Mesh Definition

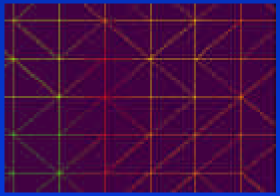
A mesh M = (P,F,I).



P – vertices, F - faces , I – attributes.

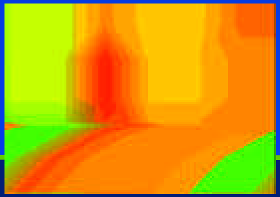
Connectivity (topology):

Assume triangular meshes for simplicity.

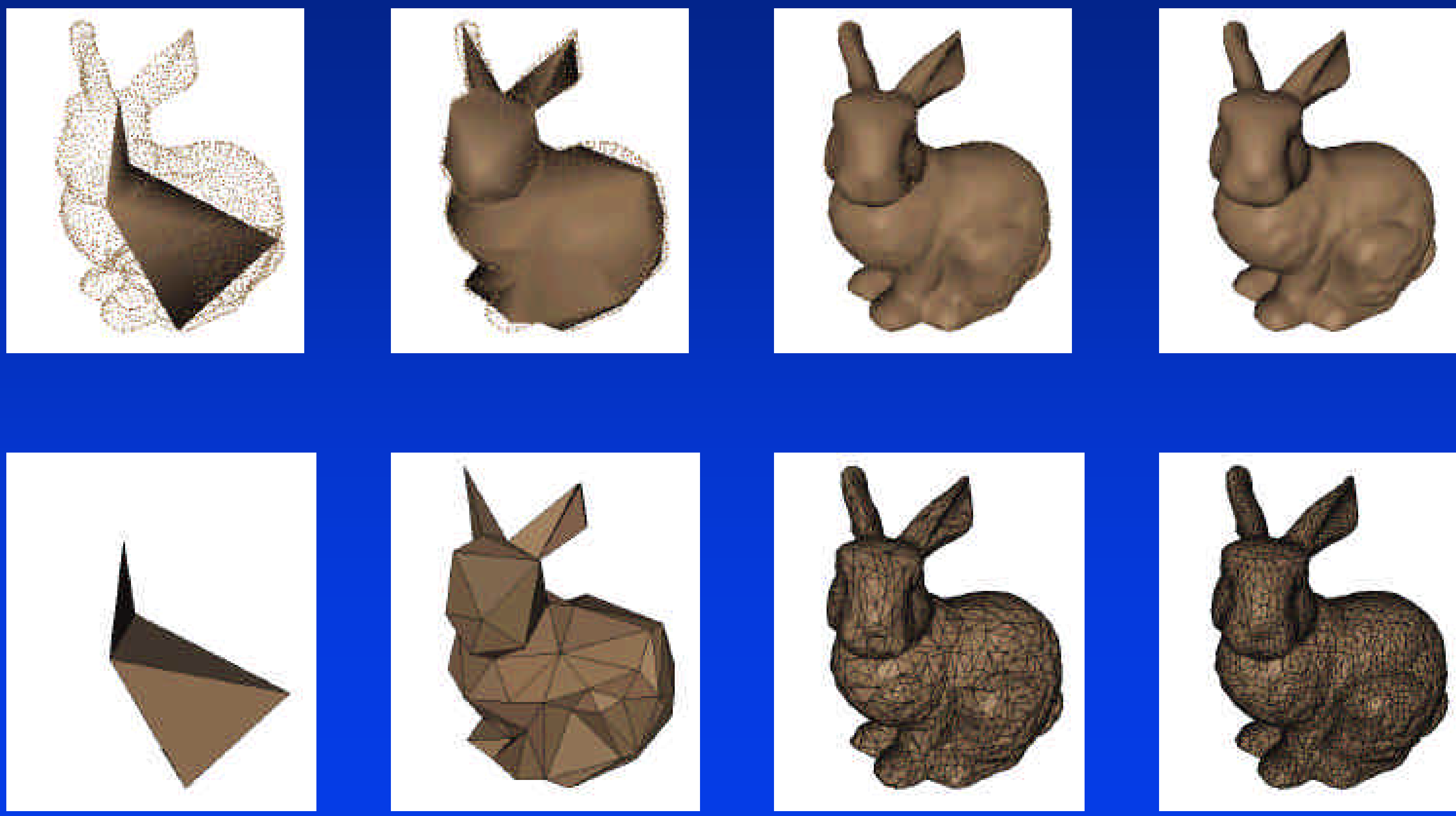


Attributes (geometry etc…):

Set of mappings A_k: M ® R^m *defined on the vertices. Examples: position, normal, height, intensity, direction, pressure, porosity.*



Multi-resolution



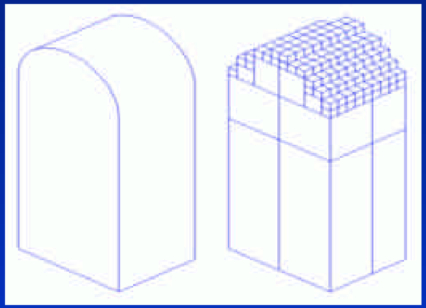
Multi-resolution Motivation

- *Level of detail rendering and visualization*
- *Progressive transmission*
- *Interactivity*
- *Multi-scale modeling and simulations*

Geometric Indexing/subdivision

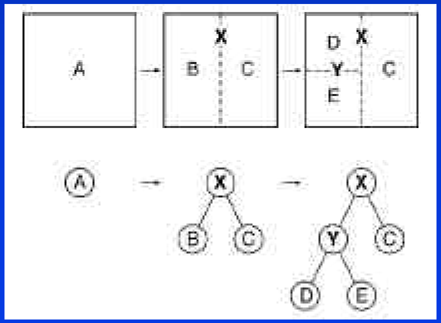
Quad-tree, Oct-tree and 2^n-tree

[Wilheilms,vangelder92] [Shephard,georges91]
[Bajaj,Pascucci,Rabbiolo,schikore98]



kd-tree

[bentley75]
[Shen,Hansen,Livnat,johnson96]



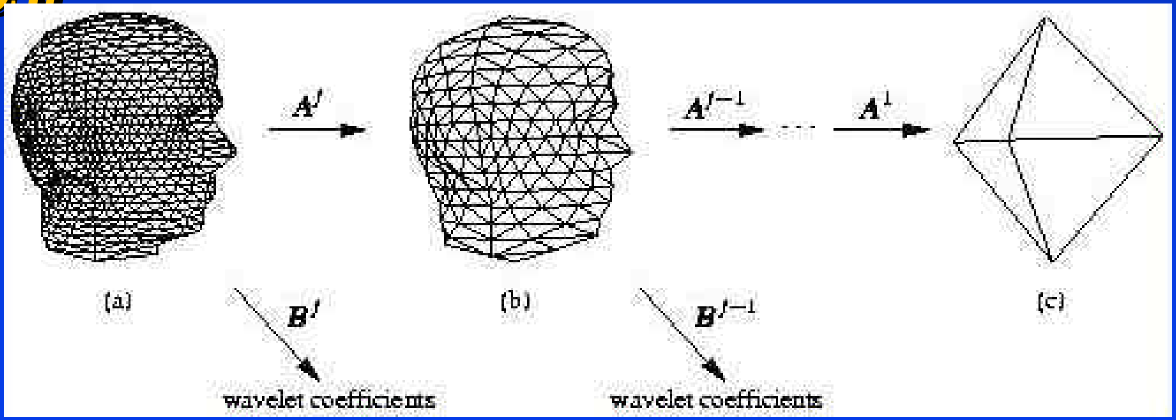
BSP-tree

[Thibault,naylor87] [Paterson,yao90]
[Venecek,phd89] [Paoluzzi,baldazzi98]

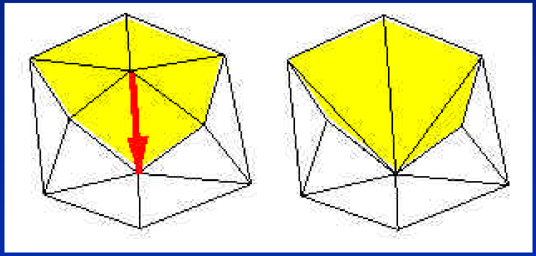
Refinment (Top Down)

Recursive subdivision and wavelets analysis

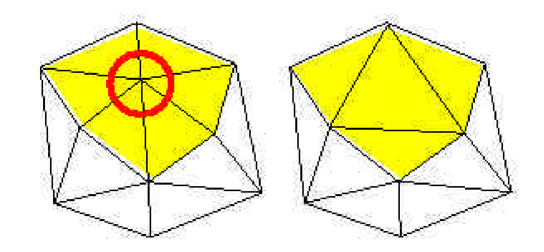
[Lounsbery,DeRose,Warren97] [Kobbelt96]
[Zorin,Schroeder,Sweldens96/97]
[Forsey,Weng93]



Decimation (Bottom Up)



Edge Contraction
[Hoppe96] [Popovic,Hoppe97] [Staadt,Gross98] [Trotts,Hamann,Joy,Wiley98][Gueziec96]



Vertex Removal
[Bajaj,Schikore98][De Berg,Dobrindt98] [Lee,Dobkin,Sweldens,Cowsar,Schroder98]

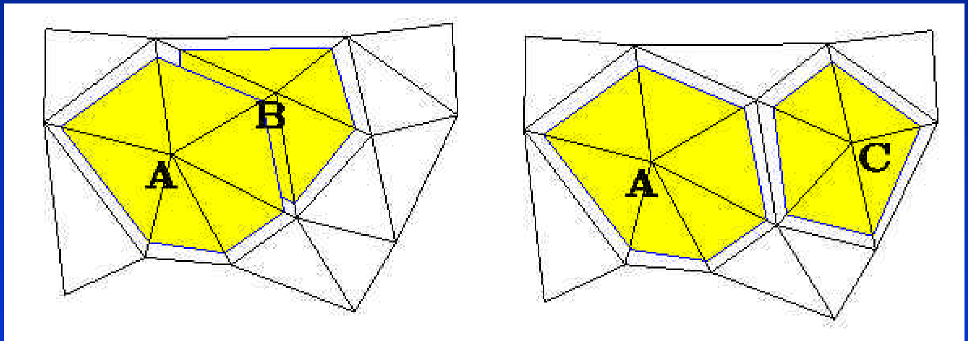
Triangle Contraction
[Hamann97]

Hole Re-triangulation
[Defloriani89] [Cohen,Varshney,Manocha,Turk96]

General Re-triangulation
[Turk92]

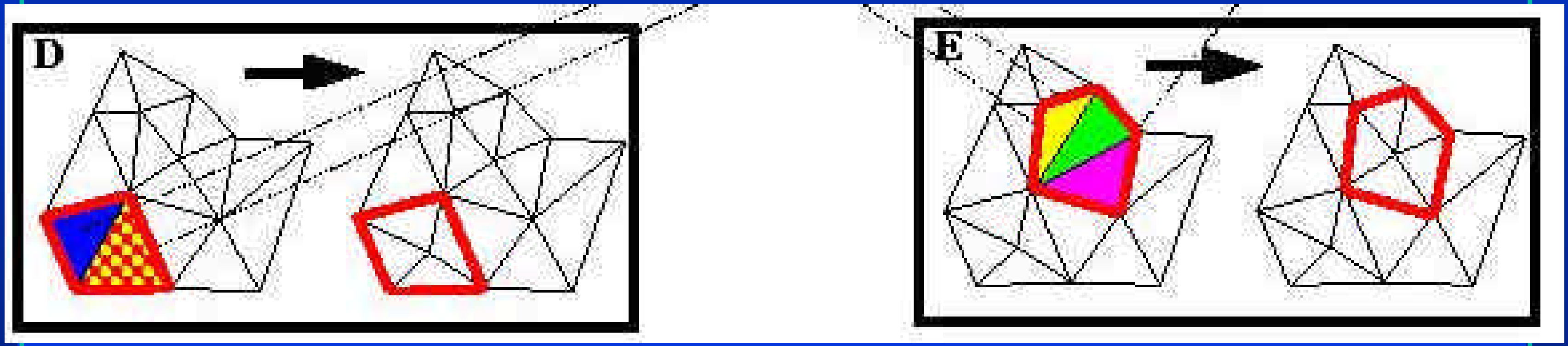
Graph Model for Multi-resolution

- *Decimate independently in levels*

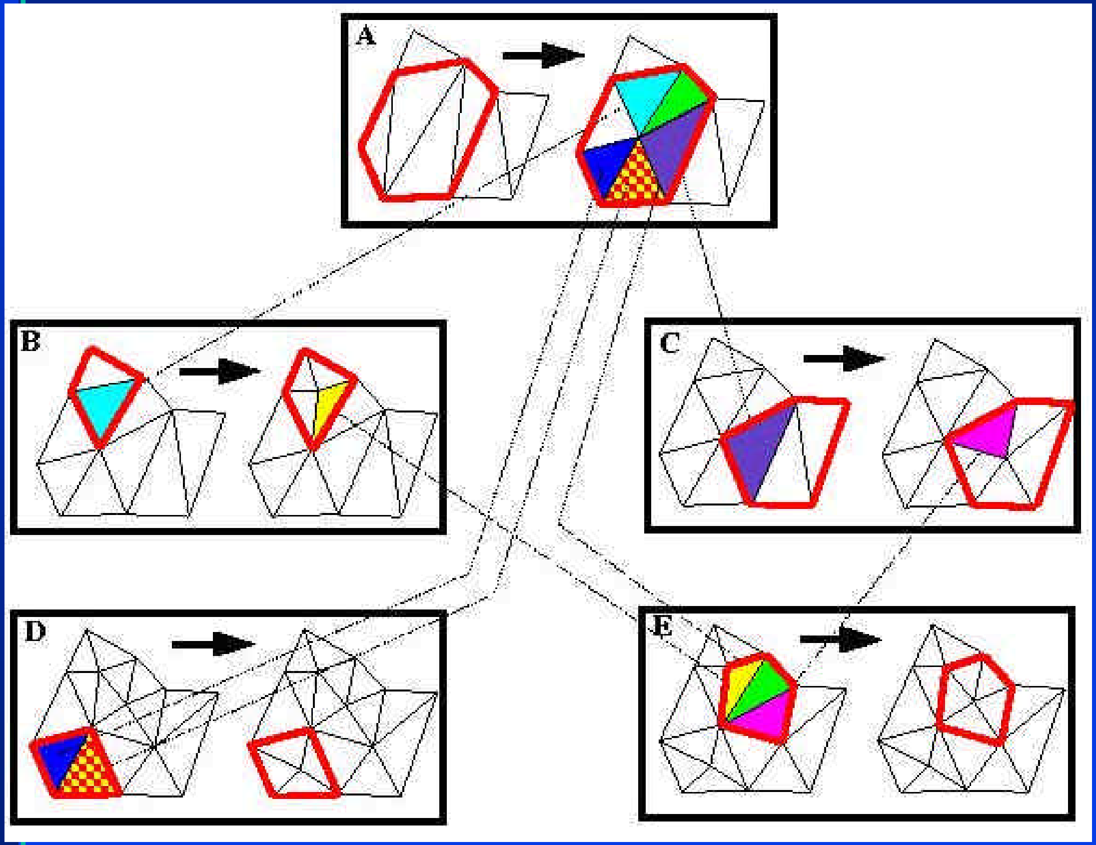


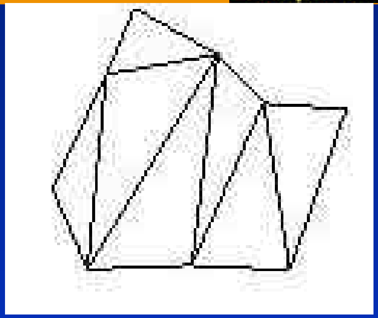
- *Create dependencies graph*
- *Any cut in the graph corresponds to an adaptive resolution model*

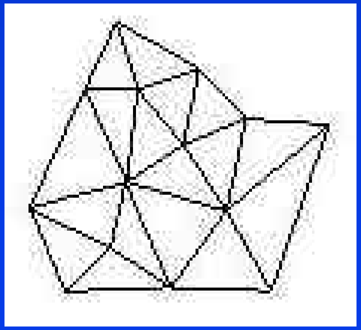
Node: Single Operation



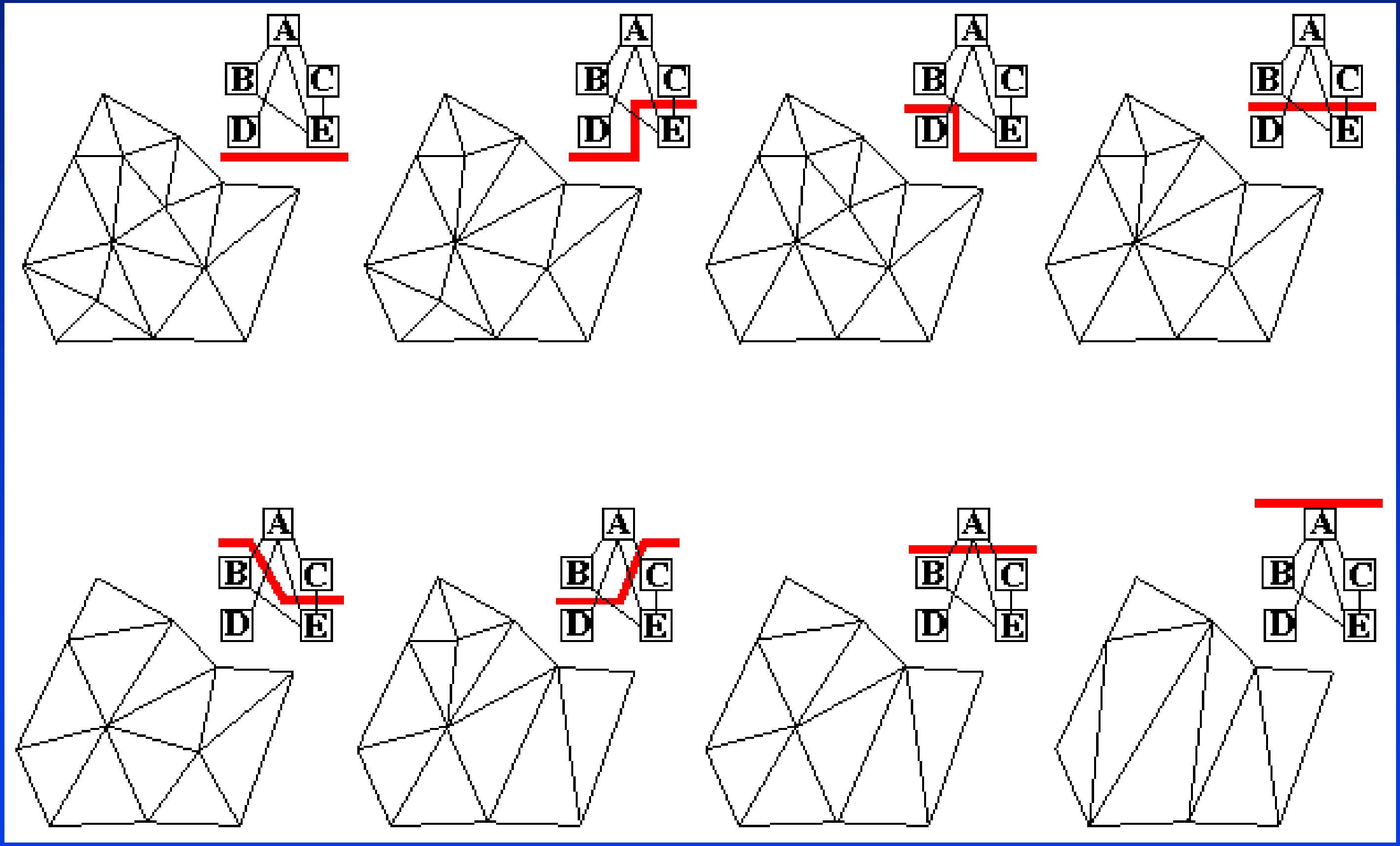
Multi-resolution Graph







Cuts in the Graph



DAG creation

```
Loop until M is coarse enough
    clear dependencies for this level
    fill Q with decimation elements from M
    while Q not empty
        e = Q->first()
        if e is dependent
            continue
        if e->cost() > tol
            break
        applyDecimation(e,M,G)
    raise tol
```

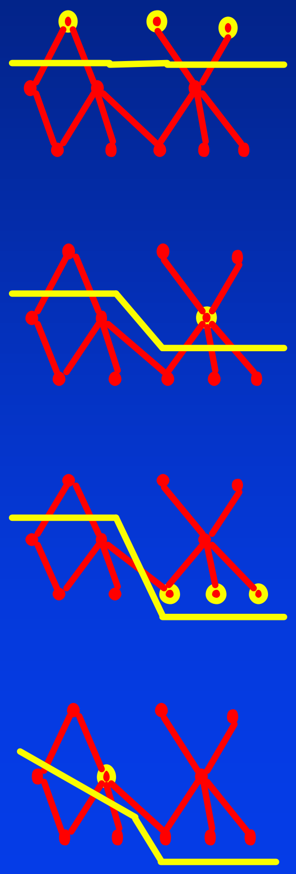
Multi-resolution Scheme

- ***Define decimation primitive***
- ***Define error (estimate) for priority and traversal***
- ***Create multi-resolution structure (independent decimation in levels)***
- ***Traversal***

Traversal

From roots to leaves:

```
Insert roots to Q
while Q not empty
      n = Q->front()
      apply n
      for all c children of n
                if cost(c) > tol
                          Add_to_queue(c,Q)
```



Better: start from "cut", first expand and then contract (not minimal).

Parameters in the Process

Decimation/refinement primitive

- Affects the neighborhood, the error

Cost function (error estimation)

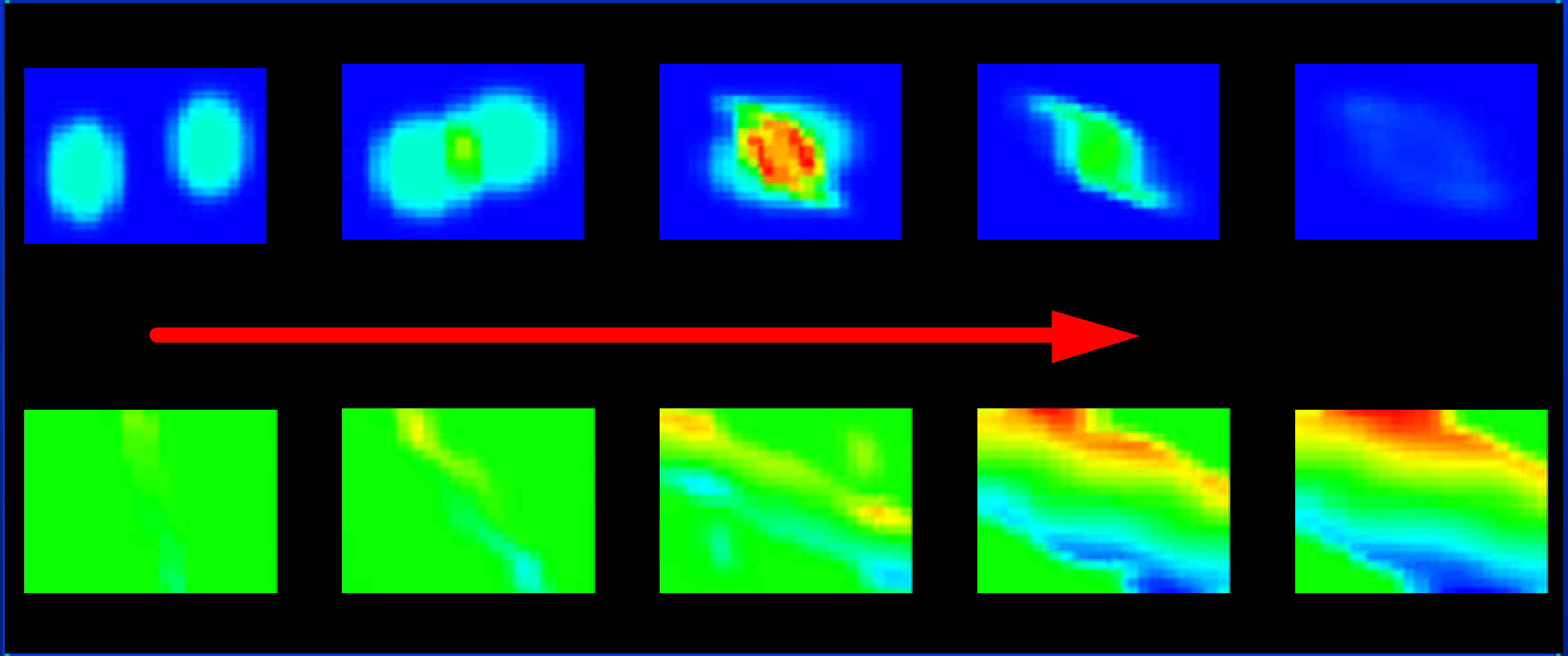
- Affects both the priority queue and traversal (different?)

Tolerances

- Affects the shape of graph

Dynamic Mesh Changes

Attributes change



Dynamic Mesh Changes

Attributes change

Positions change

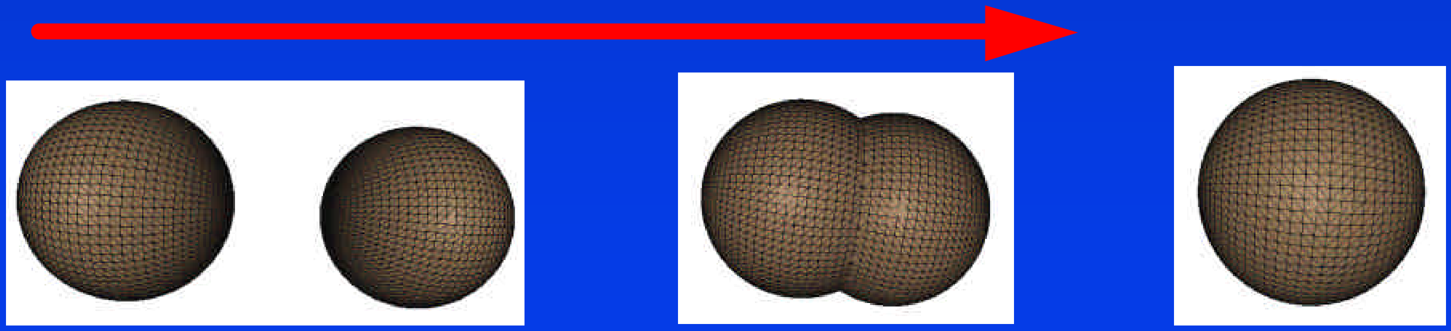
Dynamic Mesh Changes

Attributes change

Positions change

Connectivity

Topology



Dynamic Changes

Where?

- Movies (attributes), simulations (attributes, positions), animations…

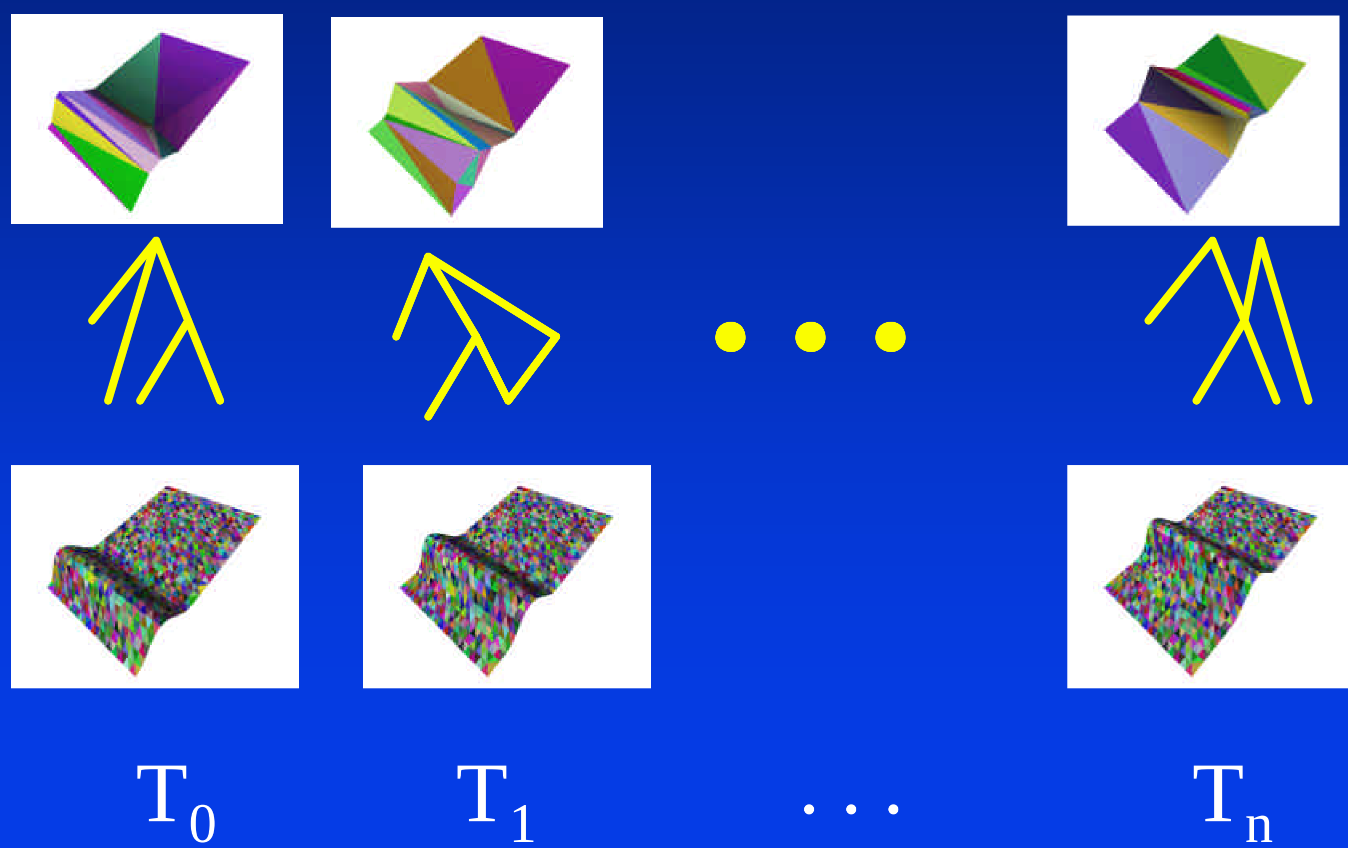
Encoding?

- MPEG4, Java3D, VRML,…

Connectivity, Topology change: much more difficult!

Multi-Resolution???

Dynamic Multi-resolution?



Model Queries

Two parameters: time t, tolerance e.

Given $M_0, M_1, \ldots M_k$:

1. *Random: given e ,t find M_t^{e}.*
2. *Incremental.*

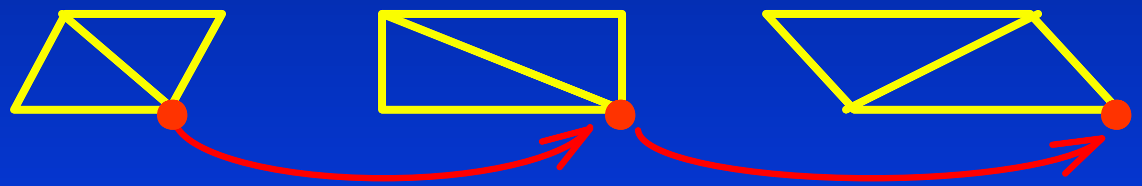
   In time: given M_t^{e} find M_{t+d}^{e},

Or in resolution: given M_t^{e1} find M_t^{e2},

Or in both.

Node Mappings

1. ***Correspondence between vertices of meshes in different time steps.***



2. ***Correspondence between nodes in different levels of details: vertex removal, edge contraction etc.***

The Node Fields

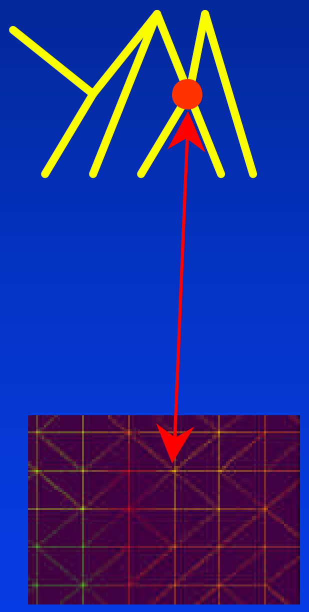
Vertex attributes

Vertex positions

Vertex decimation error

Parent links in DAG

Child links in DAG



Time Tags

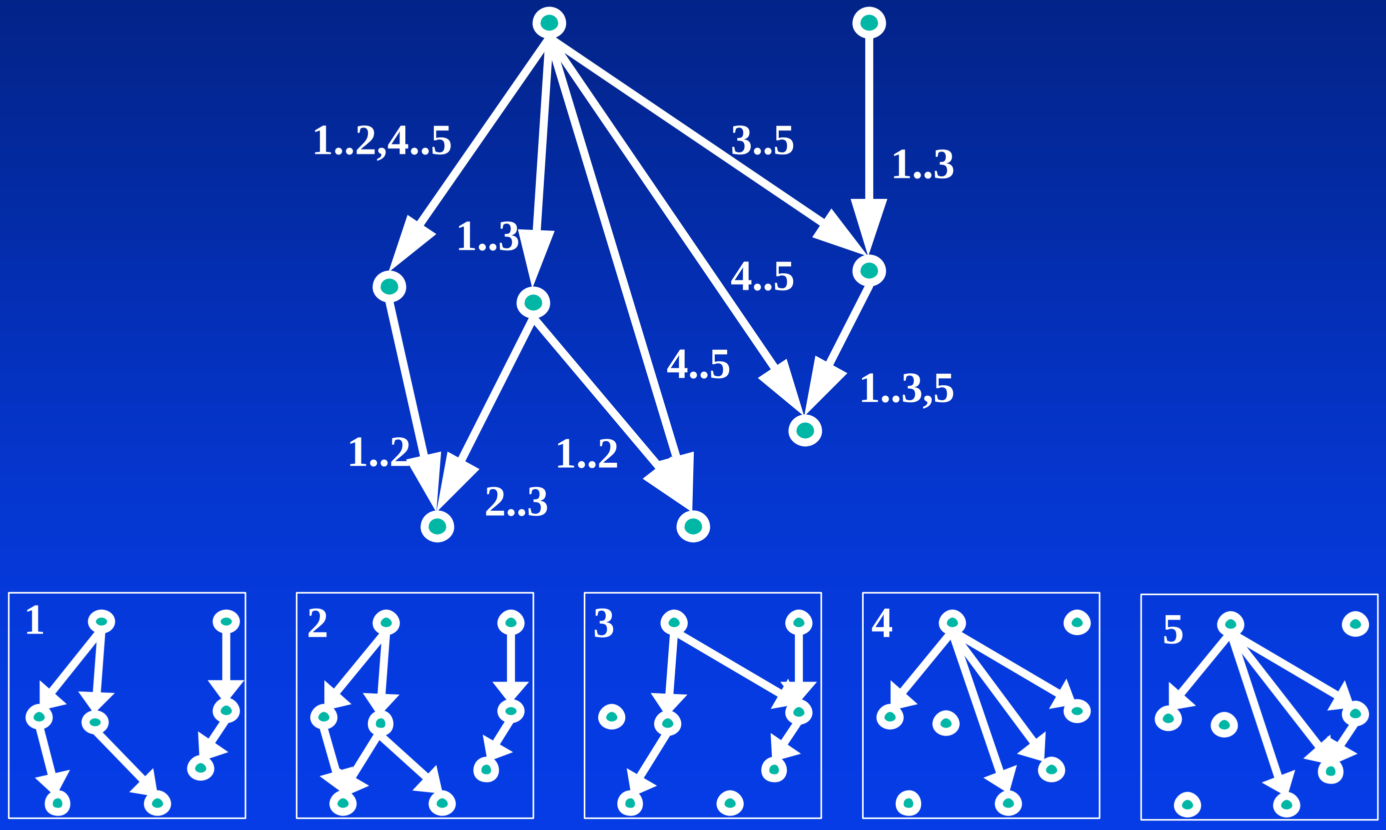
All fields are augmented with time stamps:

- Attributes.
- Position.
- Graph links (decimation dependencies).

The time stamps are a series of ranges (t_0,t_1) signifying <u>birth time</u> and <u>death time</u>.

A value is <u>alive</u> in its field if the current time is included in its tag.

Multiple DAGS: T-DAG



Type of Node Fields

***Single* value fields: f(t): R→ R (color)**

***Multiple* value fields: f(t): R→ 2^R (parent and child links)**

Time-dependent Storage and Retrieval

- *Single value – array (O(log(n)) random and O(1) incremental.*
- *Multiple value – when d is the upper bound on the size of the value set. O(d) for a window w of times and O(log(n)+wd) for random.*

Time Window

Save all alive values for each t s.t.

t_0 <= t <= t_1 → O(wd) storage.

Inside the window O(d) retrieval.

Update of window:

- Store all field values once sorted by birth time and once sorted by death time.

Lists t = 6

Time Window { Time 5 : v3, v10
Time 6: v3, v10
Time 7: v3

Birth time

0, 0, 4, 5, 8, 8, 9,10,10,13,15
v2,v0,v3,v10,v5,v4,v1,v9,v7,v6,v8

Death time

2, 3, 6, 8, 10,10,11,12,13,17,17
v0,v2,v10,v5,v3,v1,v4,v9,v7,v8,v6

Lists t = 7

Time Window

- Time 6: v3, v10
- Time 7: v3
- Time 8: v3,v4,v5

Birth time

0, 0, 4, 5, 8, 8, 9,10,10,13,15

v2,v0,v3,v10,v5,v4,v1,v9,v7,v6,v8

Death time

2, 3, 6, 8, 10,10,11,12,13,17,17

v0,v2,v10,v5,v3,v1,v4,v9,v7,v8,v6

Lists t = 8

Time Window
- Time 7: v3
- Time 8: v3,v4,v5
- Time 9: v3,v4,v1

Birth time

0, 0, 4, 5, 8, 8, 9,10,10,13,15

v2,v0,v3,v10,v5,v4,v1,v9,v7,v6,v8

Death time

2, 3, 6, 8, 10,10,11,12,13,17,17

v0,v2,v10,v5,v3,v1,v4,v9,v7,v8,v6

Lists t = 9

Time Window
- Time 8: v3,v4,v5
- Time 9: v3,v4,v1
- Time 10: v3,v4,v1,v9,v7

Birth time

0	0	4	5	8	8	9	10	10	13	15
v2	v0	v3	v10	v5	v4	v1	v9	v7	v6	v8

Death time

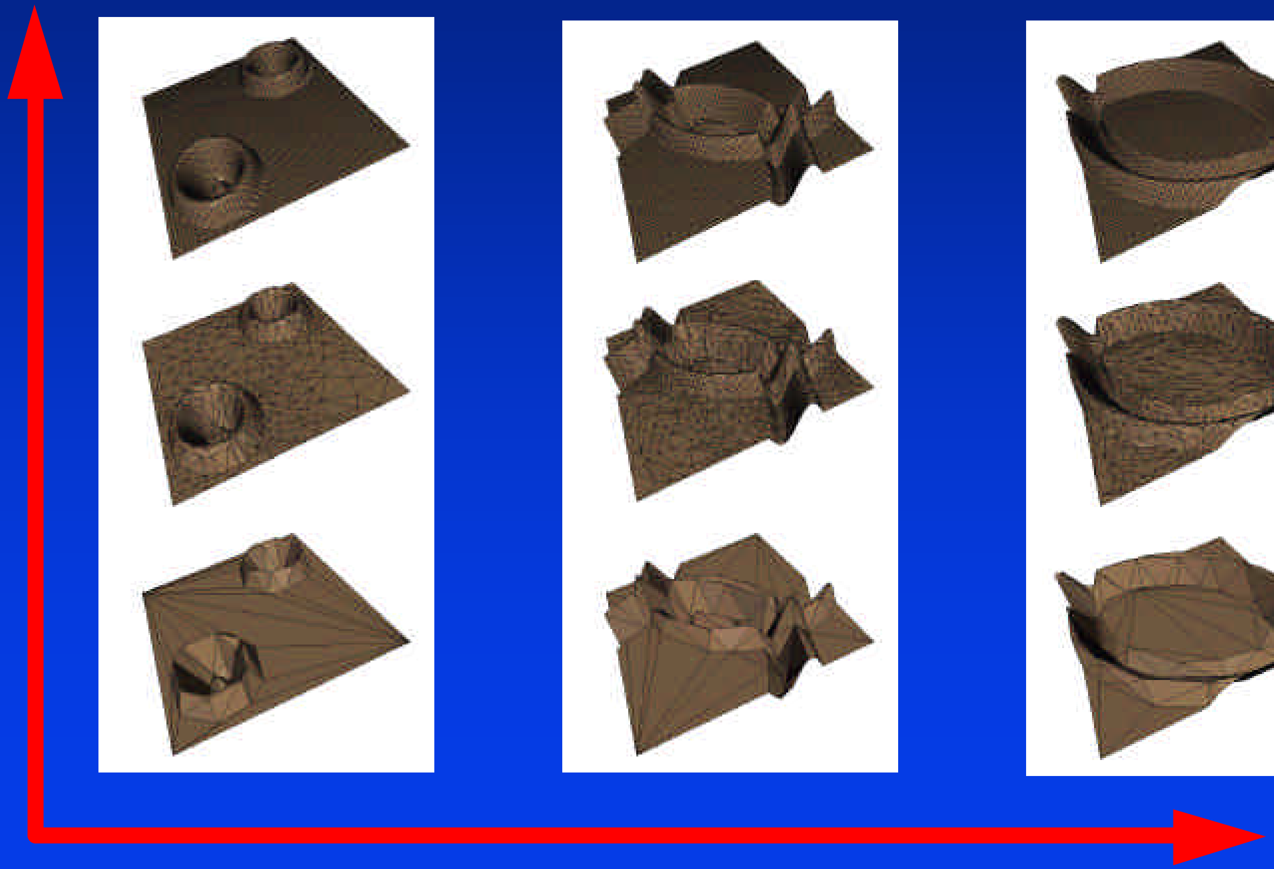
2	3	6	8	10	10	11	12	13	17	17
v0	v2	v10	v5	v3	v1	v4	v9	v7	v8	v6

Traversal

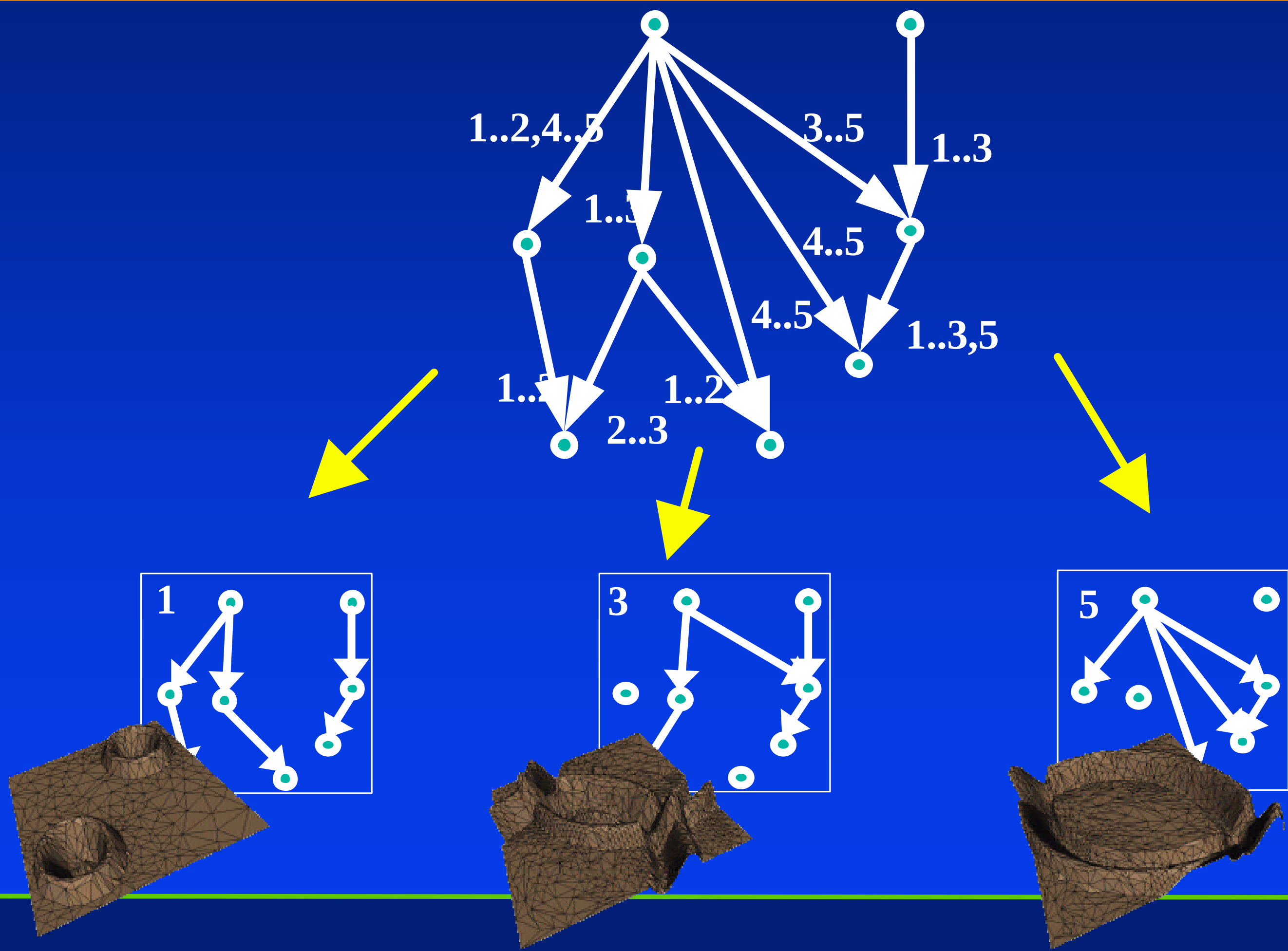
A mesh is created by top-down traversal for each time step following alive links and using alive values.

Lazy evaluation: only nodes which are traversed are incremented in time (could lead to random instead of incremental queries later – but only a subset of the nodes are visted).

2D of Time x Resolution



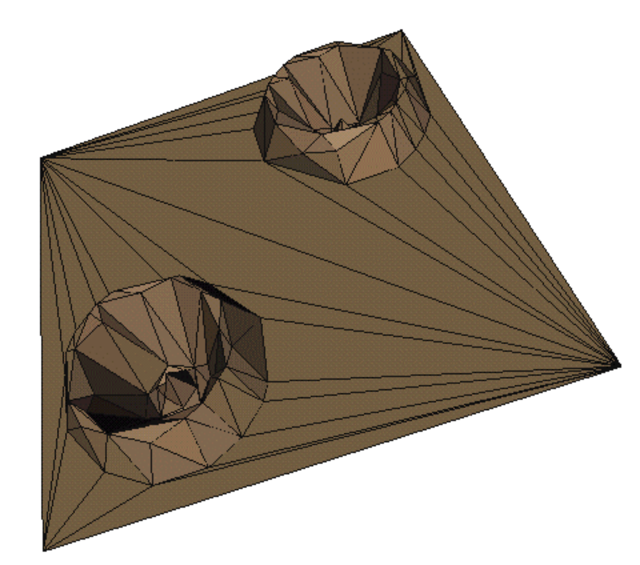
Efficient Construction?

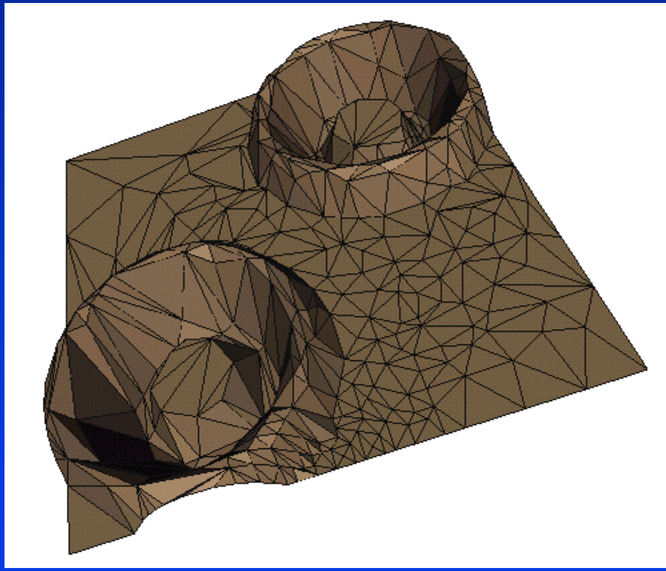


Guideline

Preserving the structure of the DAG results in efficient storage (longer time tags) and retrieval (not much change in the window of time), but also implies less adapted DAG locally (in time) and result in a deeper cut (larger mesh) for a given tolerance.

Different Time Cuts for the Same Tolerance





Basic Construction

- *Bottom up decimation for time step t uses a priority queue and creates a DAG for time t.*
- *Note that the DAG for time t and t+1 are the same iff the two decimation orders are the same.*

Merging Queues

Instead of merging two DAGs, the decimation of time t+1 uses an enhanced priority taking into account the order in the previous time t, creating similar DAGs.

→ On-line construction scheme!

TDAG creation

```
Loop until M is coarse enough
    clear dependencies for this level
    fill Q with decimation elements from M
    H is previous history of decimations
    while H is not empty
            e = H->getnext(), find e` matching e in Q.
            if e’ is dependent
                or  e’->cost() > tol
                or LargeDiff(e->cost(),e’->cost())
                    continue
            applyDecimation(e’,M,G)
    While Q is not empty…
```

LargeDiff Function

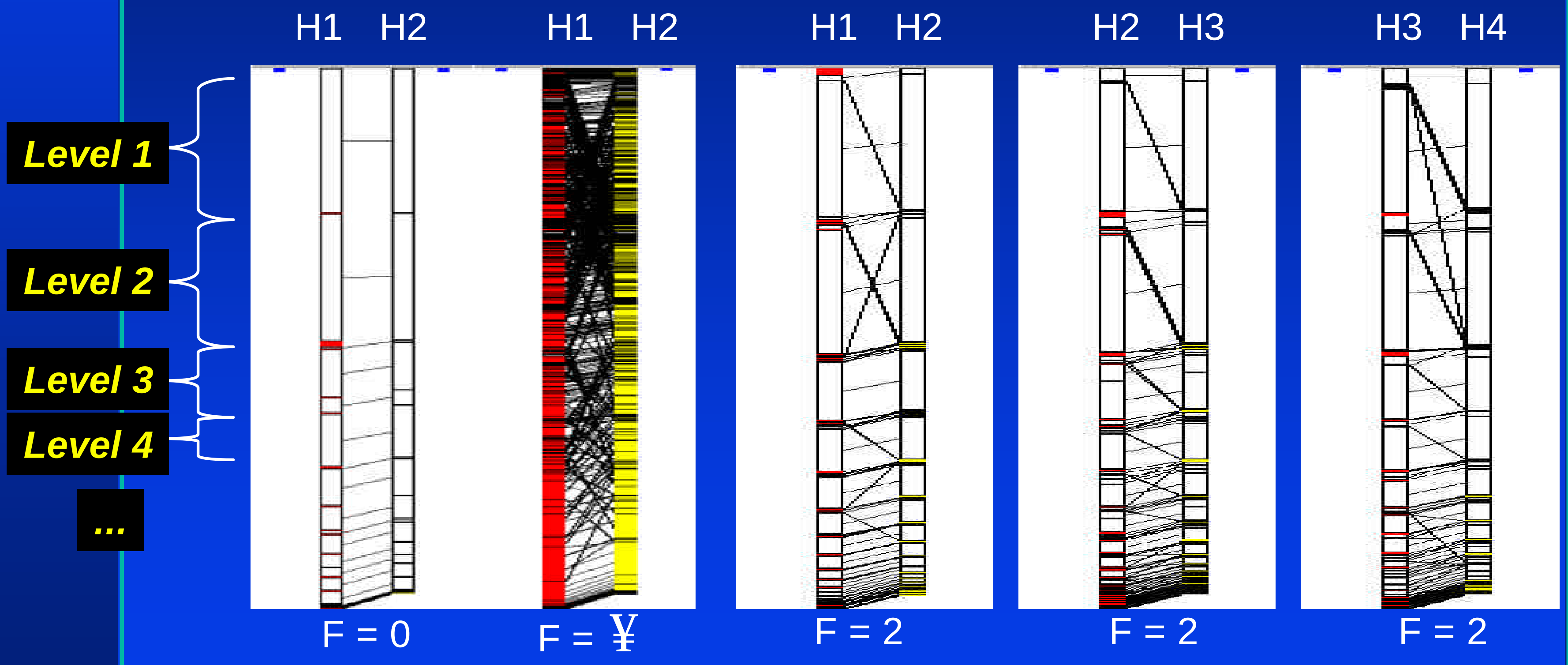
$F * |x - y| > max(|x|, |y|)$

F is the conforming factor:

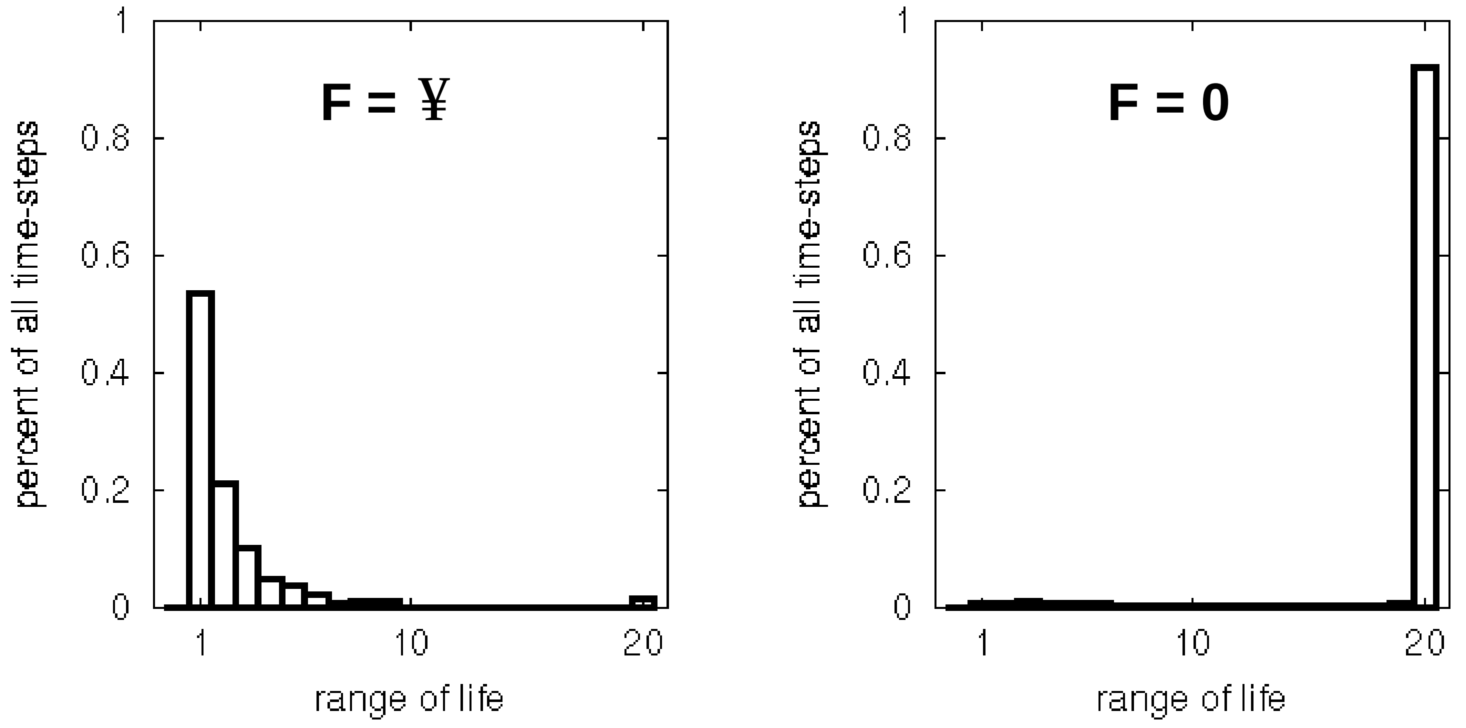
F → 0 means greater conformity.

F → ¥ means independent decimation for each time step.

Decimation Histories

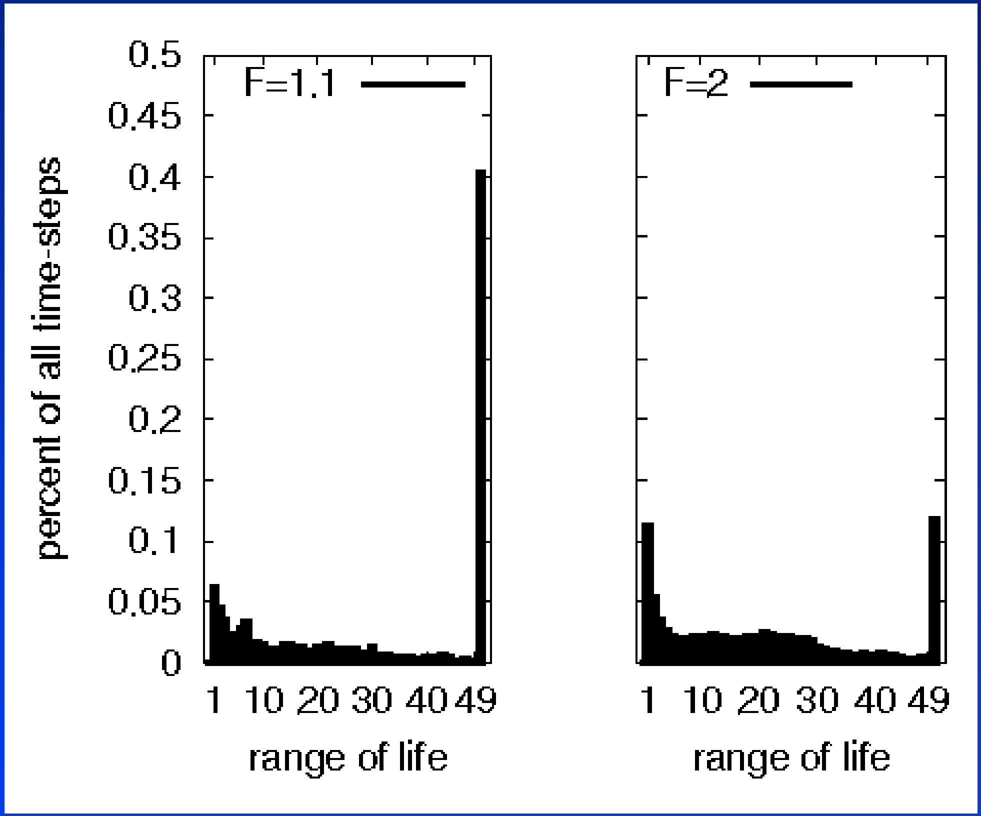


Child Links Life Span

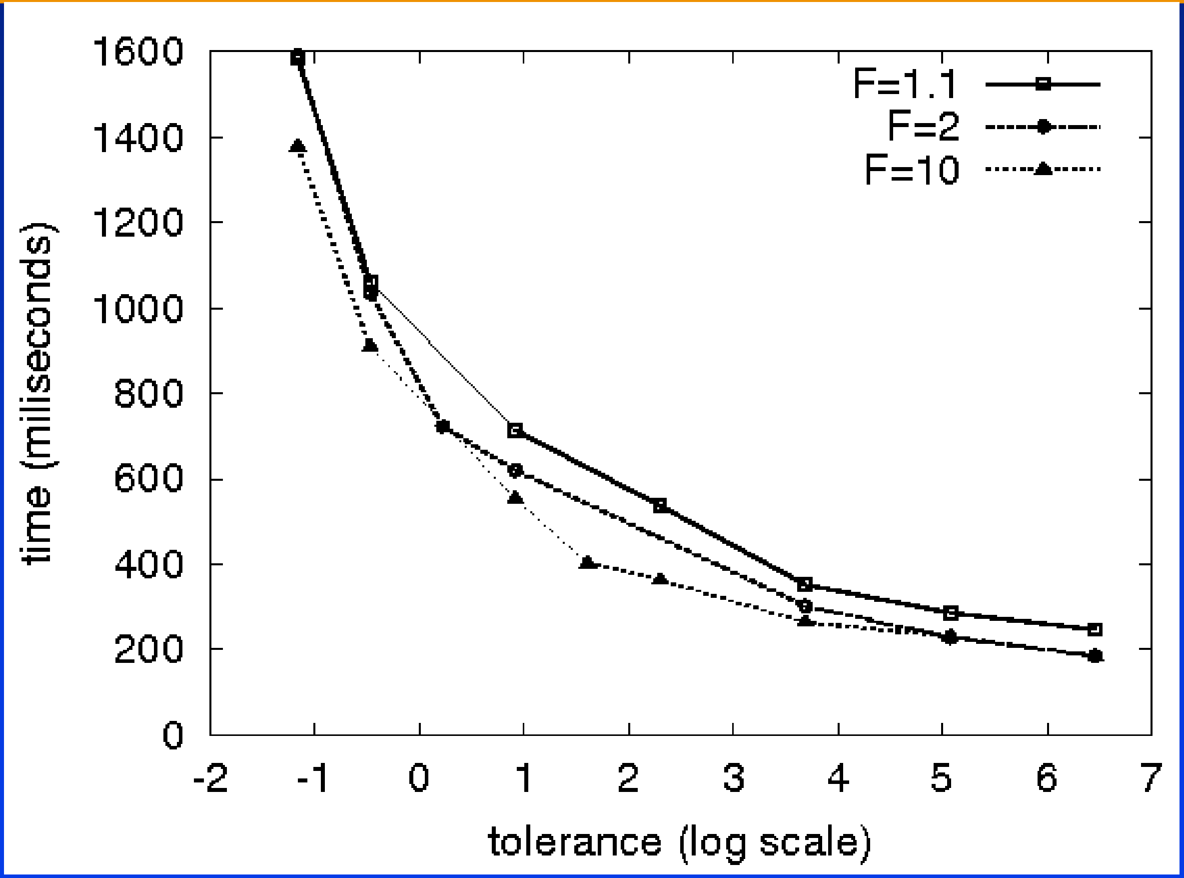


Right TDAG size =~ 1/3 * Left TDAG size

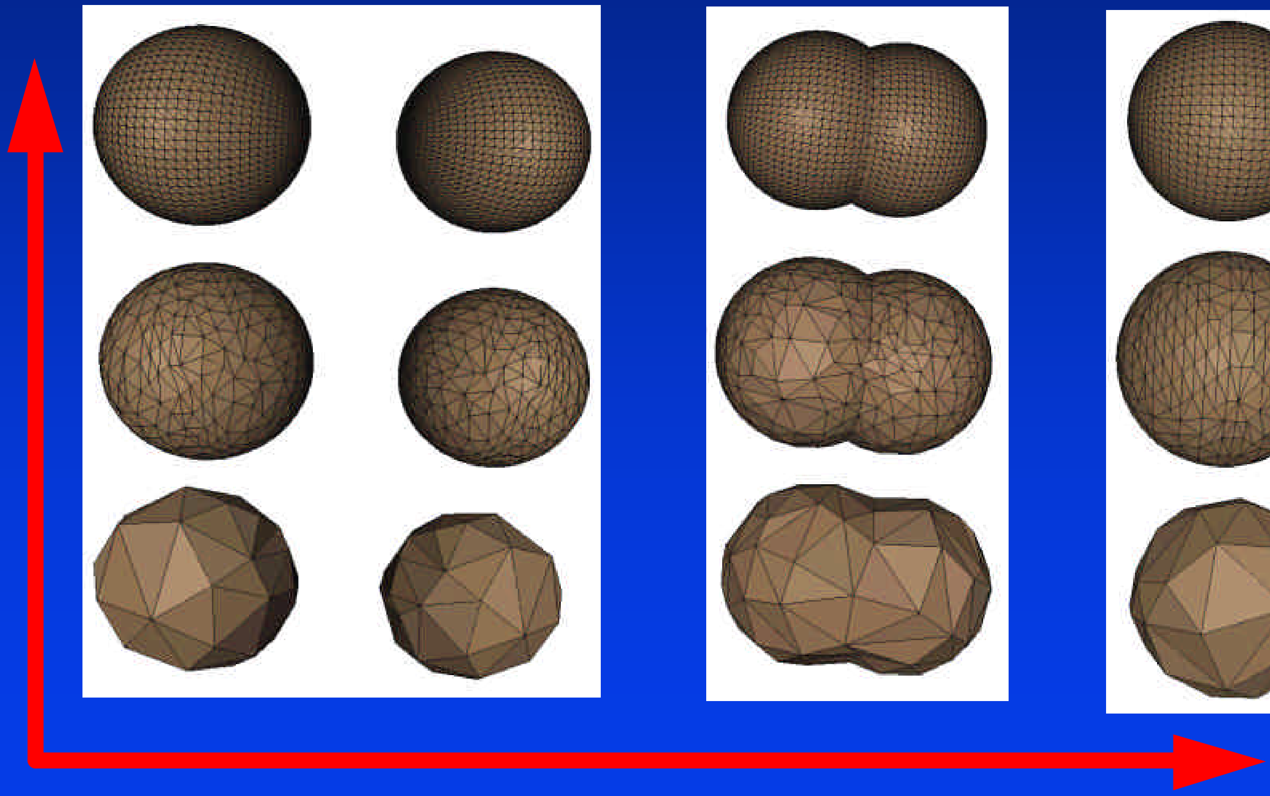
Parent Link Life Span



Rendering Time



Topology Change



Summary

- *Dynamic multi-resolution model.*
- *General framework (different decimation schemes and errors).*
- *Covers a wide range of dynamic meshes and changes.*
- *Possible space-time tradeoffs in construction.*

Future…

Dynamic update of cut.

Fractions of time.

Large models: out of core handling!

Time dependent constraints on LOD (faster needs less details).