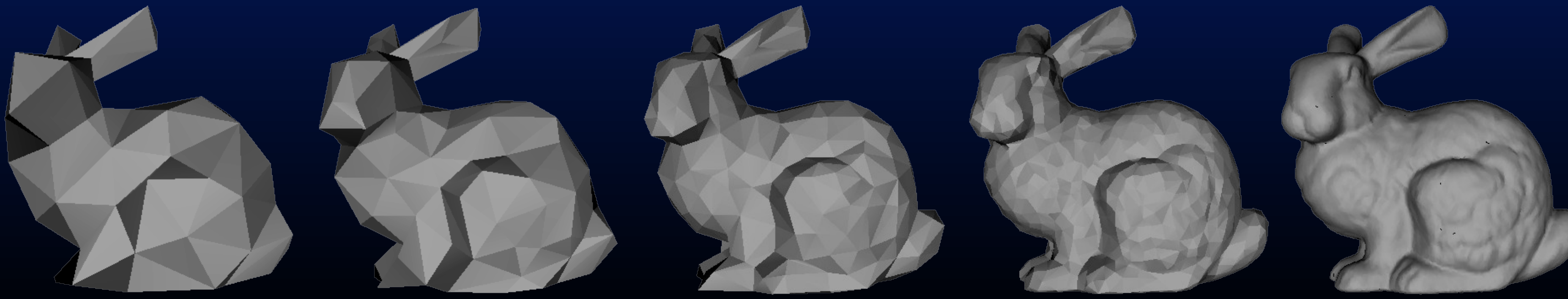
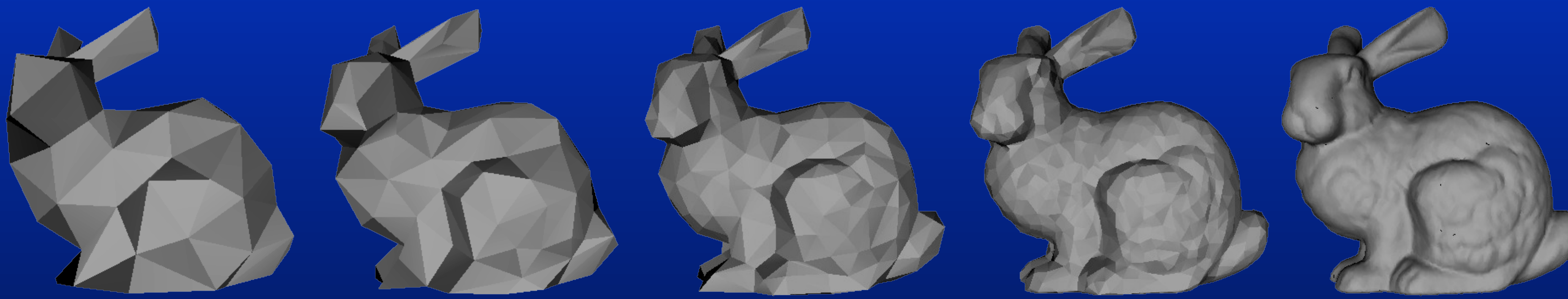
SAN ANTONIO

SIGGRAPH 2002

Advanced Issues In Level Of Detail



Algorithms for Generalized LODs



Amitabh Varshney

Graphics and Visual Informatics Laboratory

Department of Computer Science

University of Maryland at College Park

http://www.cs.umd.edu/gvil

SAN ANTONIO SIGGRAPH 2002

LOD Algorithms Classification

	View-Independent	View-Dependent
Topology Preserving	Turk 92 Schroeder et al 92 Cohen et al 96 Hoppe 96 Cignoni et al 98 Lindstrom & Turk 99	Xia & Varshney 96 Hoppe 97 De Floriani et al 98 Gueziec et al 98 Klein et al 98
Topology Simplifying	Rossignac & Borrel 93 He et al 96 El-Sana & Varshney 97 Schroeder 97 Garland & Heckbert 97	Luebke & Erikson 97 El-Sana & Varshney 99

Geometry & Topology Simplifications

- Geometry Simplification
  - Reducing the number of geometric primitives (vertices, edges, triangles)
- Topology Simplification
  - Reducing the number of holes, tunnels, cavities
- Geometry + Topology Simplification
  - Aggressive simplifications
  - May not be suitable for some applications

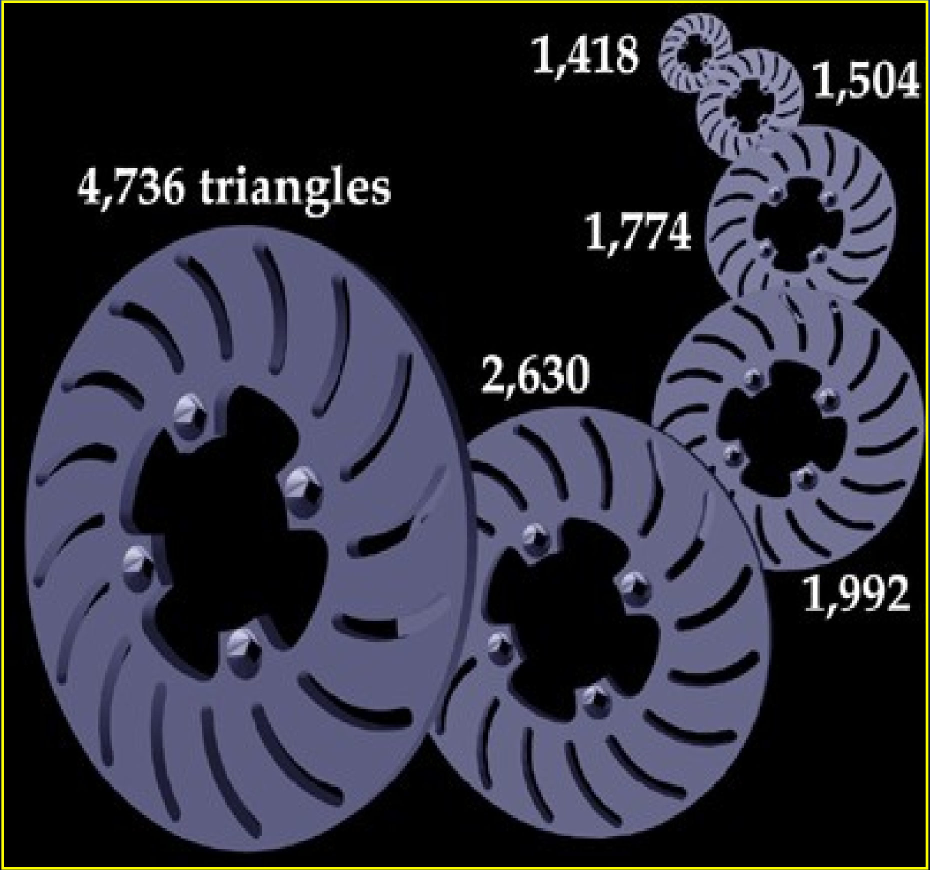
Outline

- Geometry and Topology Simplifications
- Implementing View-dependent LODs
- Variable-Precision Rendering

Outline

- Geometry and Topology Simplifications
  - **View-Independent (Discrete) Simplification of Topology**
  - View-dependent Simplification of Topology
- Implementing View-dependent LODs
- Variable-Precision Rendering

Why Discrete Simplification of Topology?



Discrete Simplification of Topology

Local Algorithms

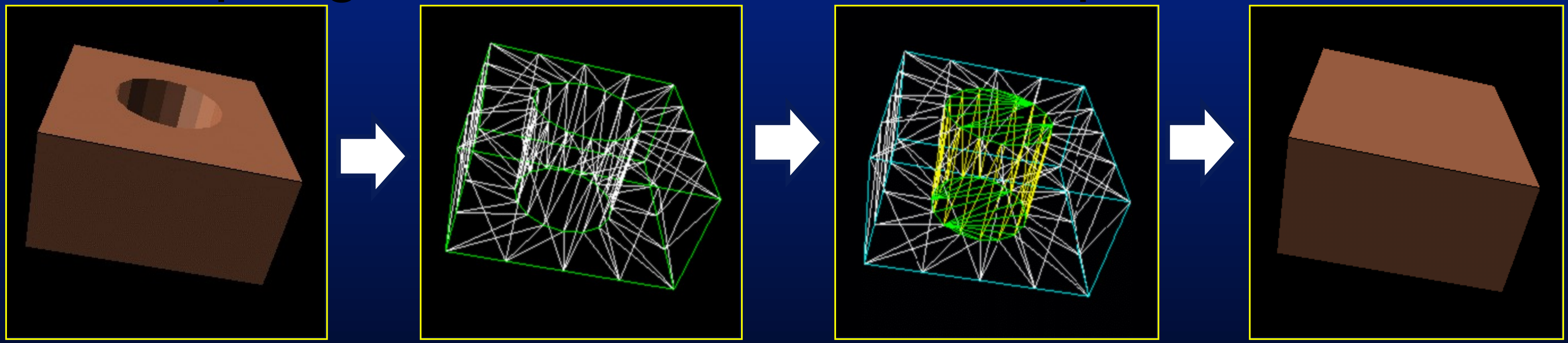
- Collapsing vertex pairs / virtual edges
  – Schroeder, *Visualization 97*
  – Popovic and Hoppe, *Siggraph 97*
  – Garland and Heckbert, *Siggraph 97*
- Collapsing primitives in a cell
  – Rossignac and Borrel, *Modeling in Comp. Graphics 93*
  – Luebke and Erikson, *Siggraph 97*

Global Algorithms

- Low-pass filtering in Volumetric domain
  - He *et al.*, *Visualization 95*
- Rolling a sphere (L_2), cube (L_1, L_∞)
  - El-Sana and Varshney, *Visualization 97*

Our Approach to Discrete Topology Simplification

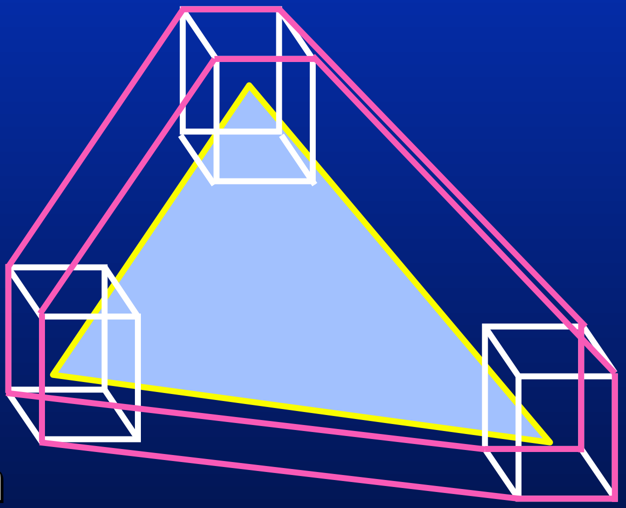
- Similar to α-Hulls
- Roll a sphere of radius α over the object
- Fill-up regions inaccessible to the sphere



El-Sana and Varshney, Visualization 97, IEEE TVCG 98

Alpha Prisms

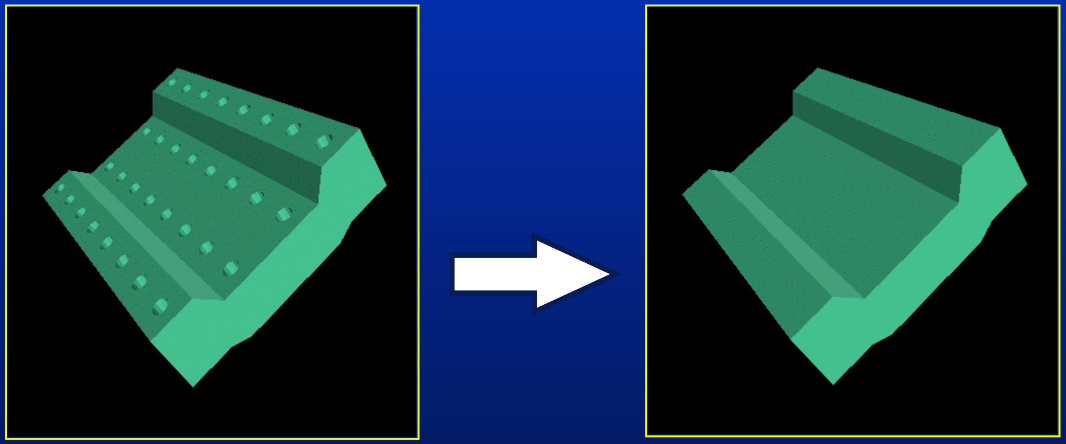
- Alpha prism
  - Convolve triangle with α-side cube (L_∞ metric)
  - Convex polyhedron
- Compute union of α prisms
  - Fills-up all features less than α
- Generate the surface from the union

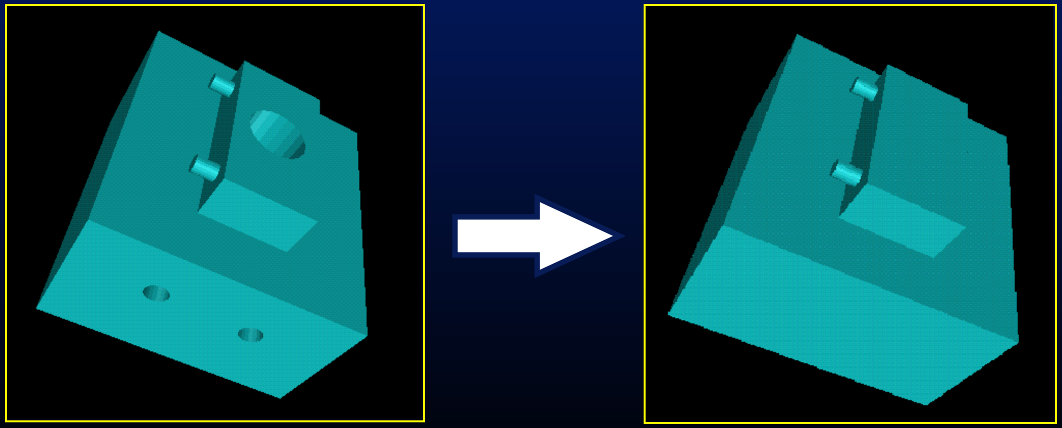


GVIL

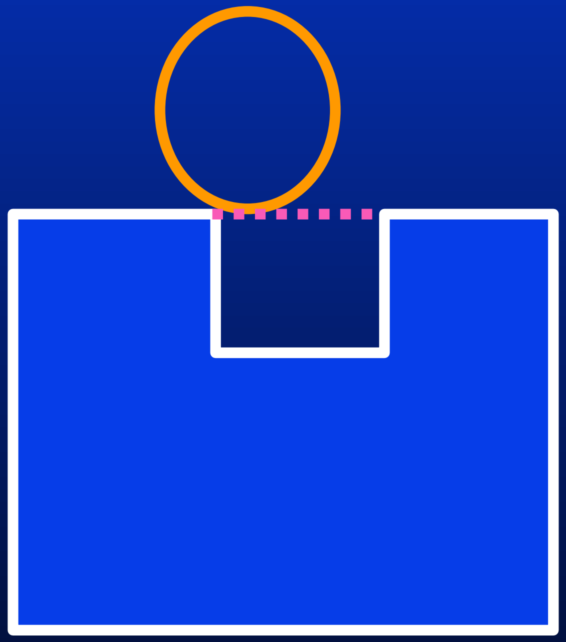
SAN ANTONIO SIGGRAPH 2002

Results

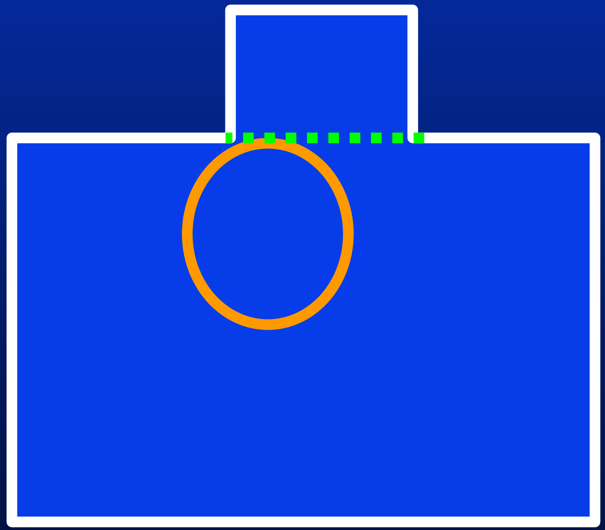




Concavities and Convexities

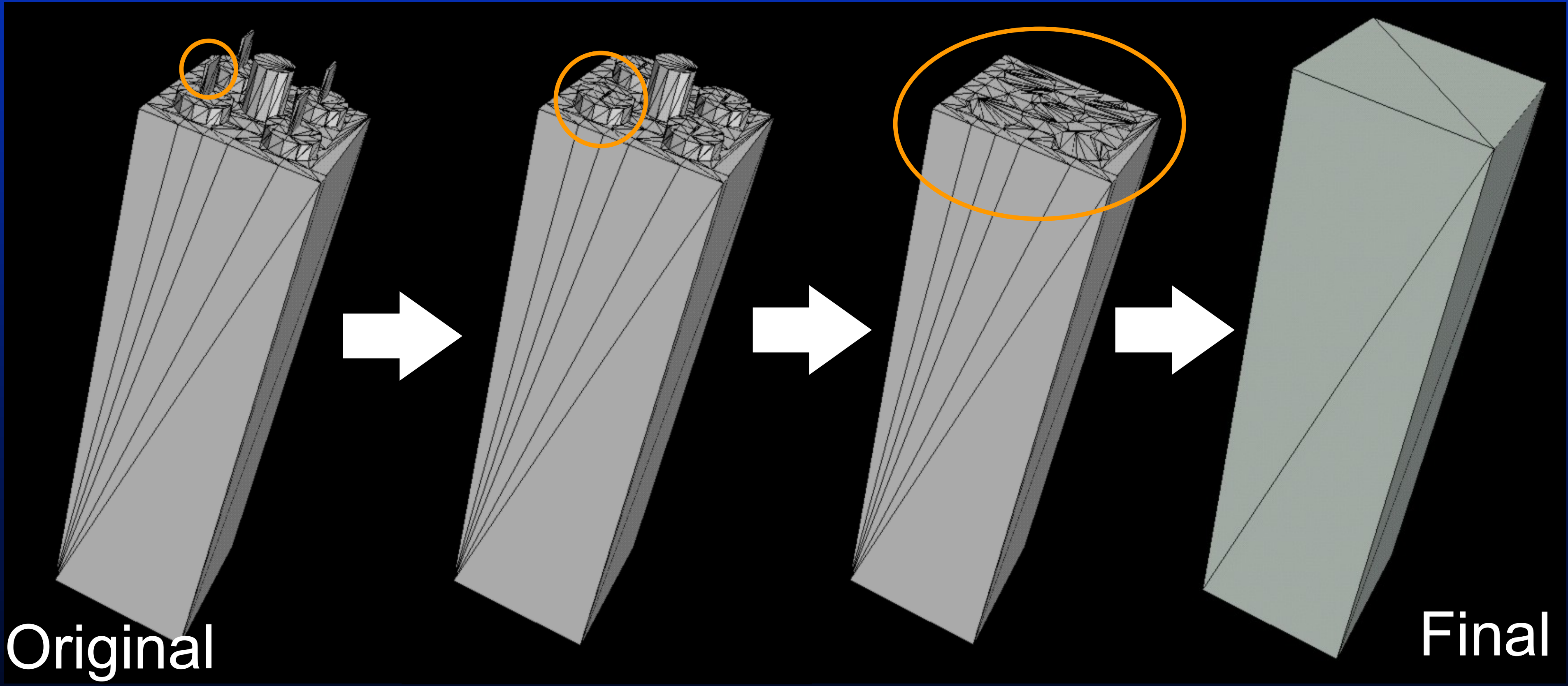


Concavity



Convexity

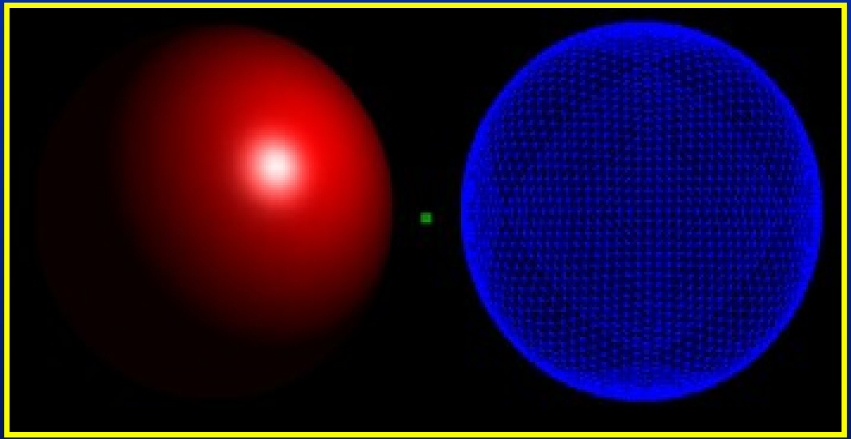
Results



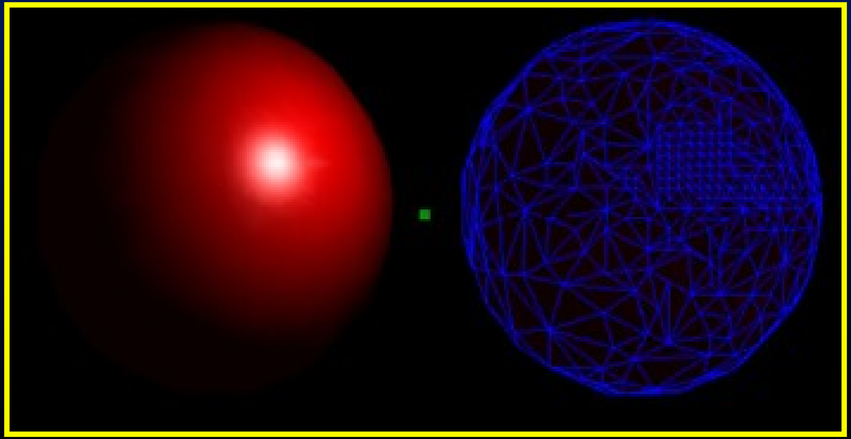
Outline

- Geometry and Topology Simplifications
  - Discrete Simplification of Topology
  - **View-dependent Simplification of Topology**
- Implementing View-dependent LODs
- Variable-Precision Rendering

Illumination- and View-Dependent Detail



8192 triangles



537 triangles

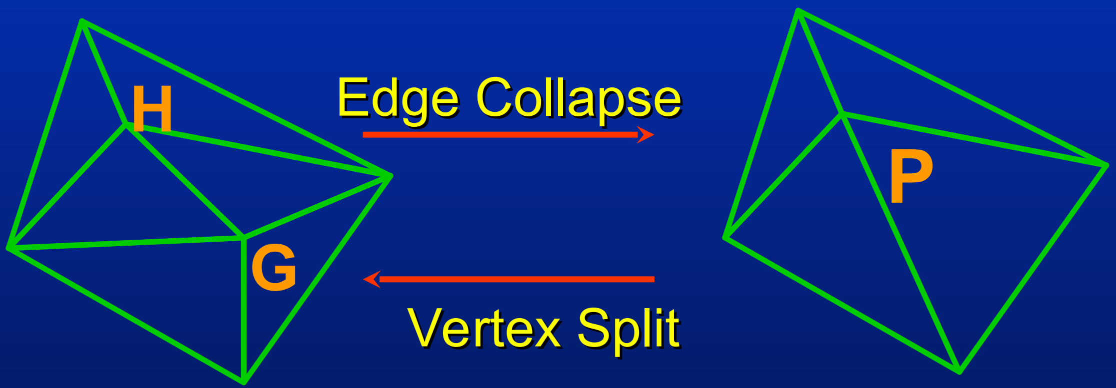
View-Dependent Topology Simplification

- Aggressive simplification
- Varied topology simplification
- Connect different objects
- Efficient fold-over prevention policy
- Real-time

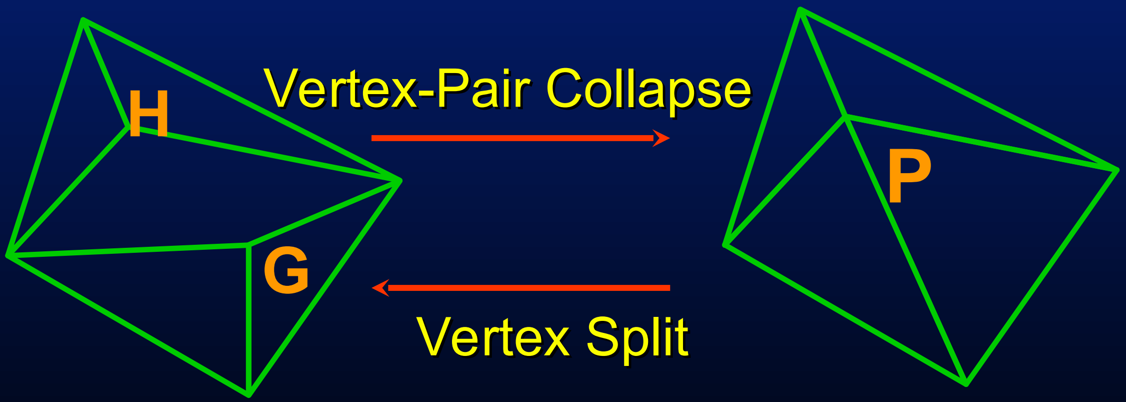
SAN ANTONIO SIGGRAPH 2002

Edge and Vertex-Pair Collapses

Edge Collapse



Vertex-Pair Collapse
(Virtual Edge)



Simplifying Genus

- Allow virtual edge collapses
- Limit potentially $O(n^2)$ virtual edges
- Typical constraints:
  - Delaunay edges
  - Edges that span neighboring cells in a spatial subdivision: octree, grids, etc.
  - Maximum edge length

Virtual Edges

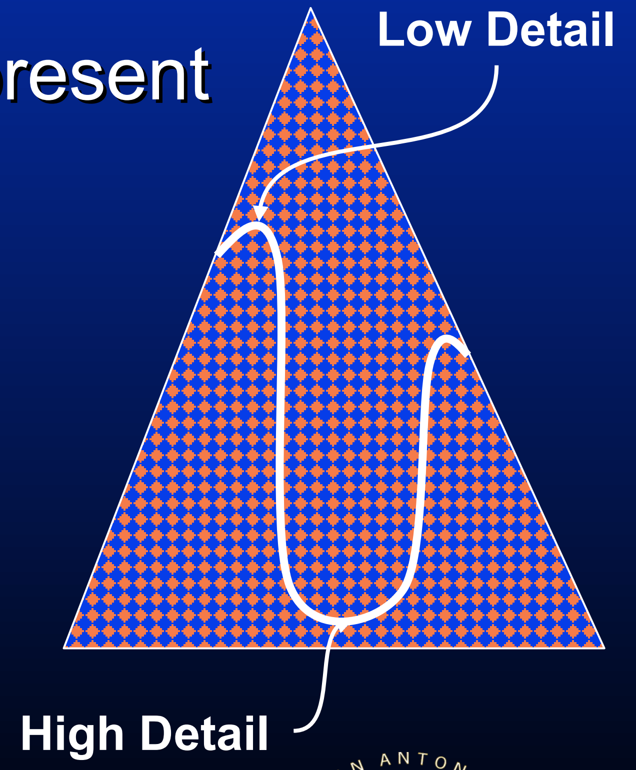
- Subdivide the dataset into patches
  - Initialize each triangle to a patch
  - Merge two patches that:
    - **Share at least one edge**
    - **Their normals differ less than threshold**
- Construct Delaunay triangulation using only the vertices on patch boundaries

View Dependence Tree

- Use an appropriate distance metric
- Construct the set of virtual edges
- 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

View Dependence Tree

- Hierarchy of vertex-pair collapses
- Different levels in this hierarchy represent different levels of detail
- Construct the hierarchy offline
- Run-time navigation involves:
  - Vertex split: Refinement
  - Vertex collapse: Simplification



SAN ANTONIO SIGGRAPH 2002

Run-time Traversal

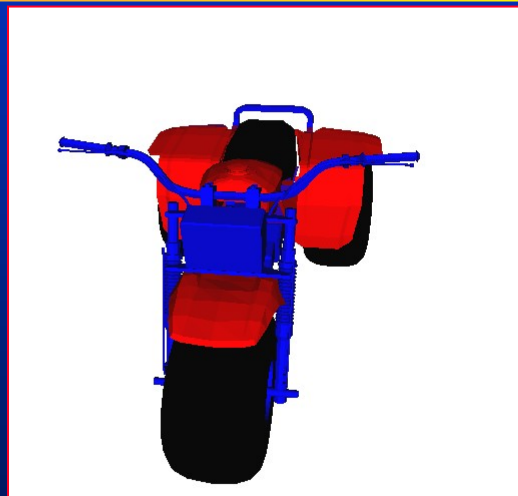
```
for each active node n do
  switch(NextStat(n)){
    case SPLIT   : if ( CanSplit(n))
                         Split(n);
    case MERGE: if ( CanMerge(n) &&
                         CanMerge(Sibling(n))
                         Merge(n);
    case STAY    : // No Change on the active-nodes list }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

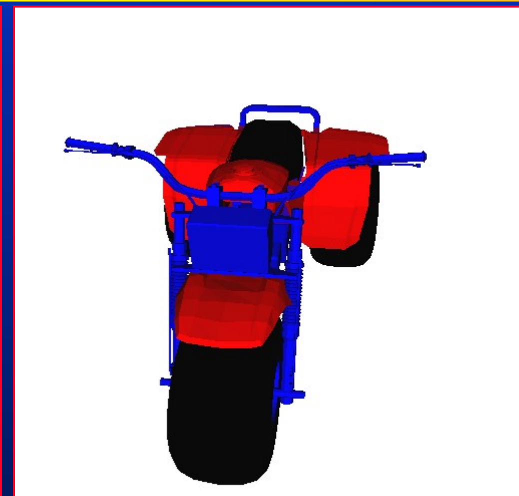
Simplification Factors

- Screen-space projection
- Local illumination
- Visibility culling
- Silhouette boundaries
- Object Speed
- LOD transfer function
- Prevent fold-overs
  - Implicit Dependencies

Results

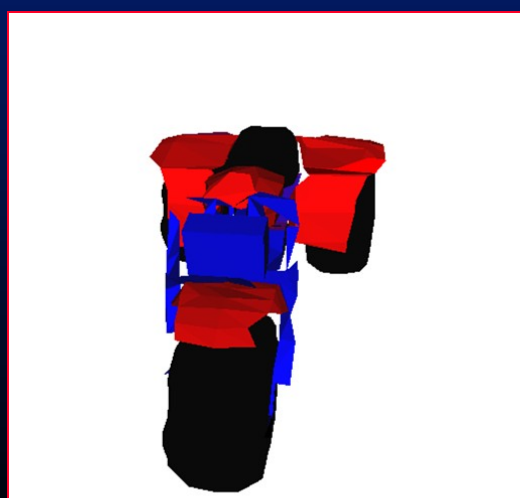
13.5K tris

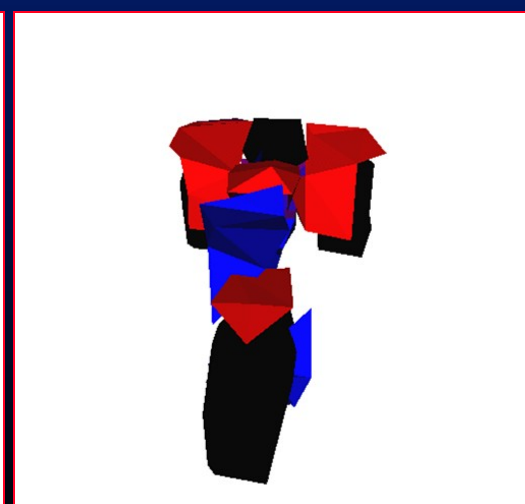




8.2K tris

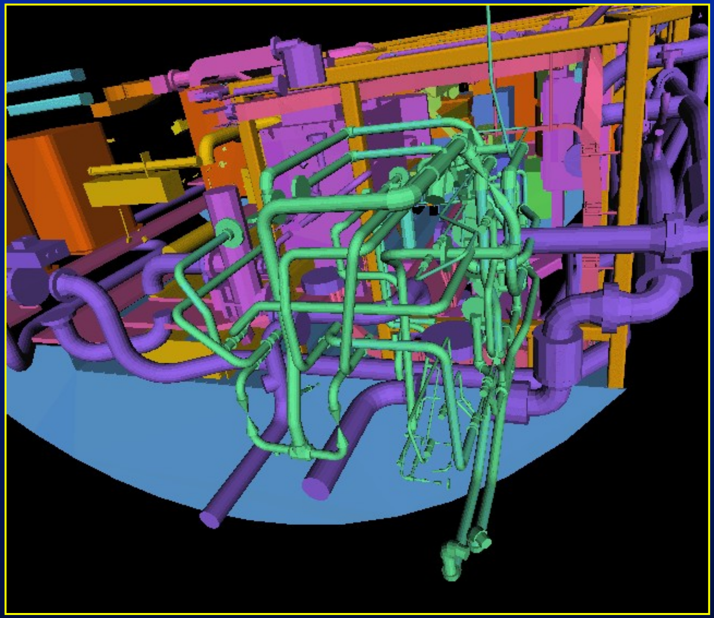
2.0K tris

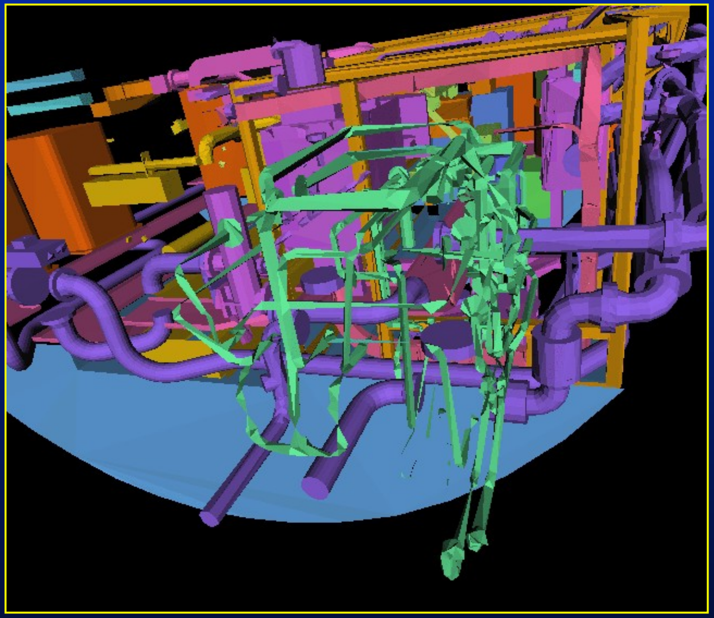




200 tris

Results



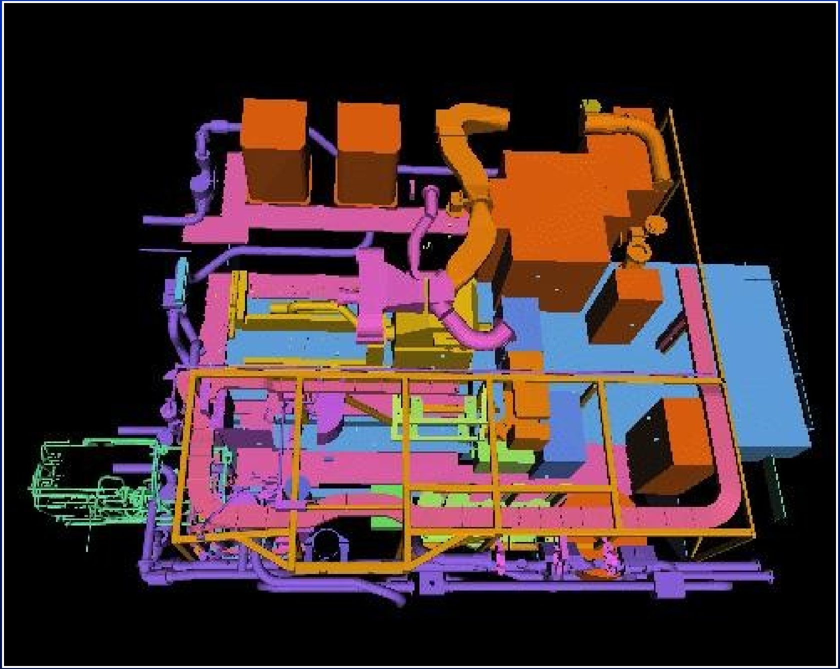


Close

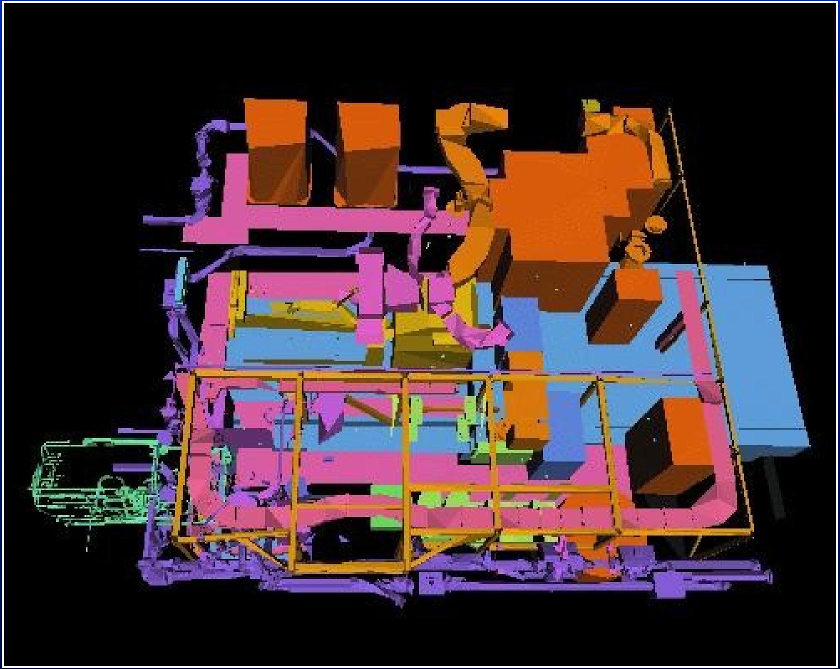
Far

El-Sana and Varshney, Eurographics 99

Results



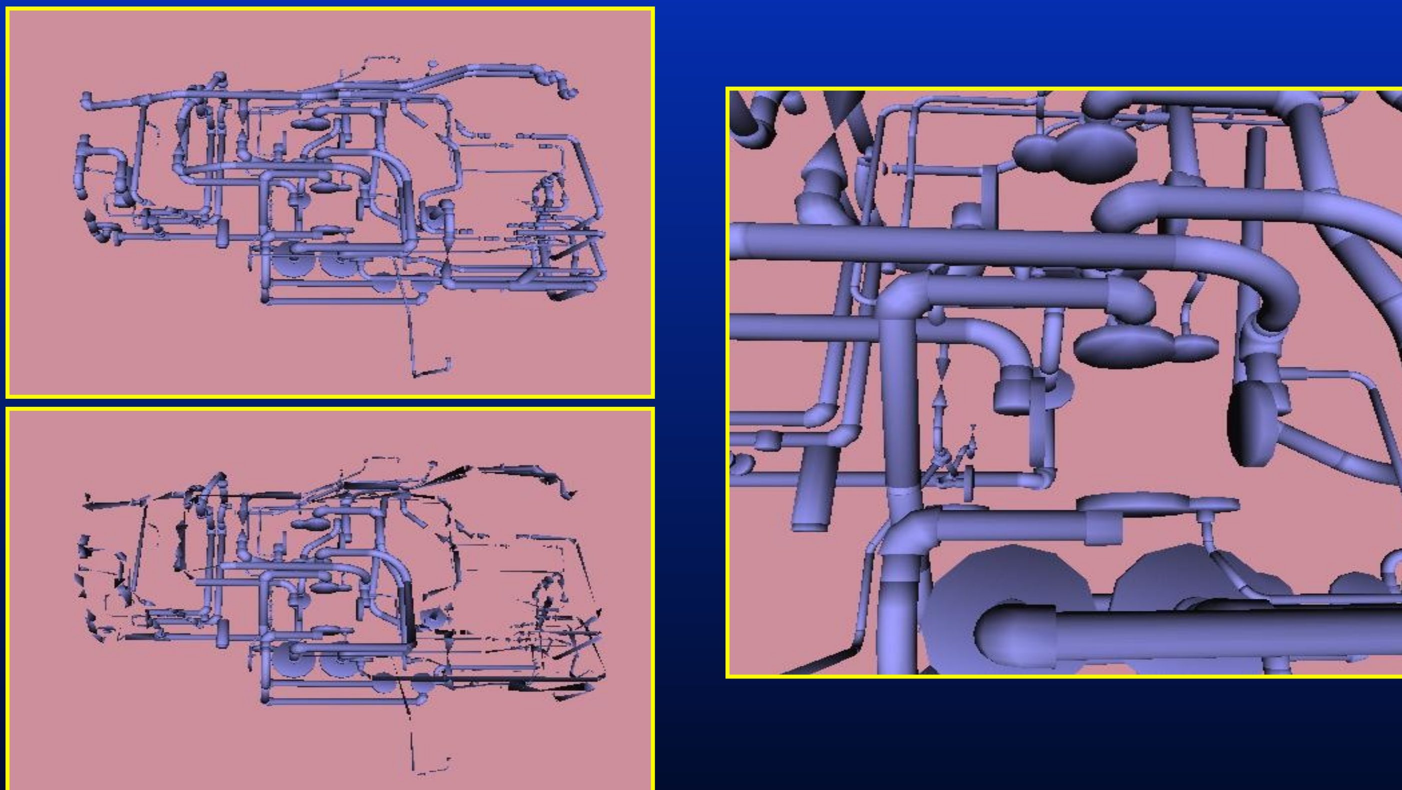
Original (340K tris)



Simplified (49K tris)

Auxilliary Machine Room Dataset

Foveation Results



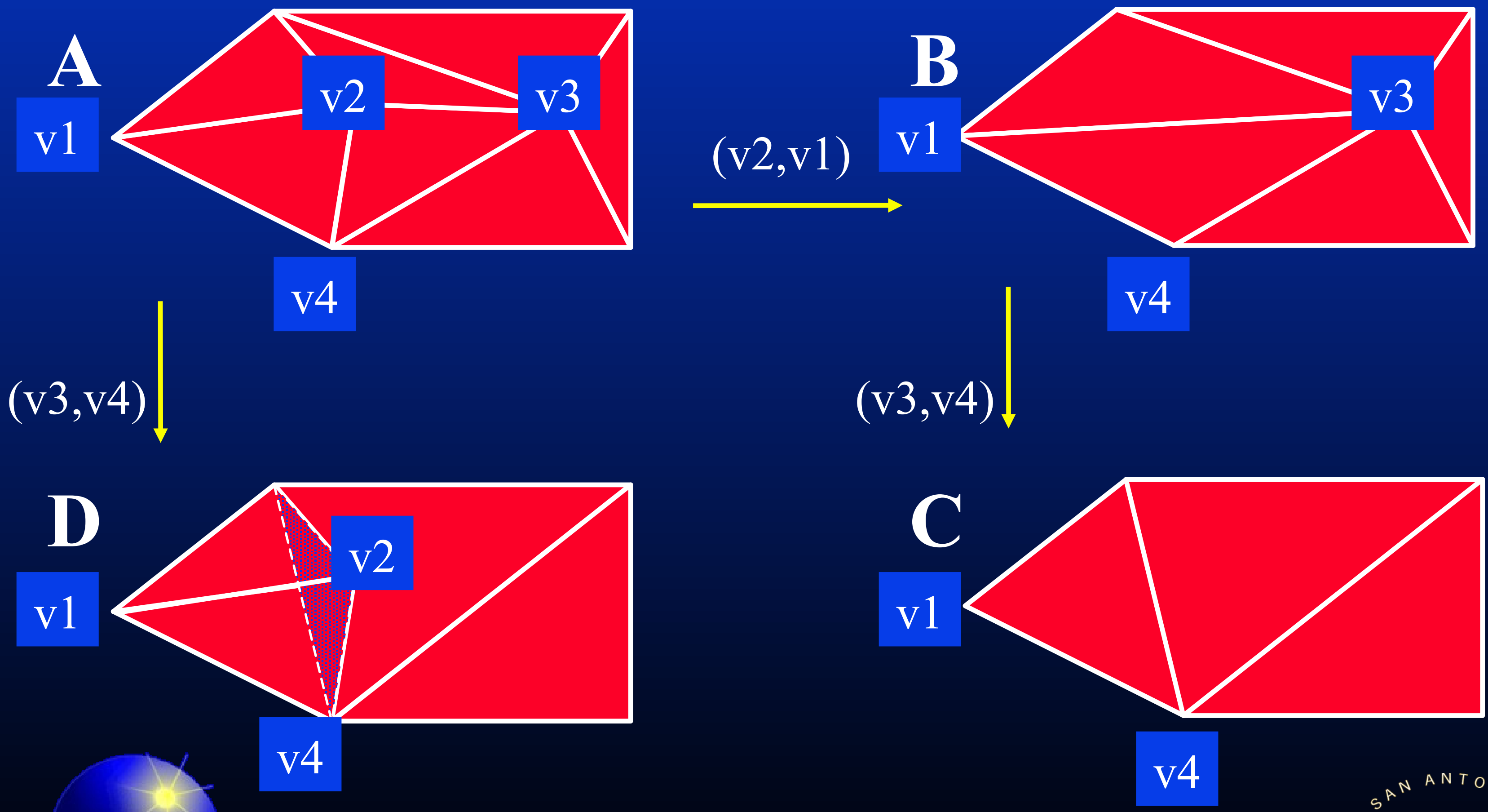
El-Sana and Varshney, Eurographics 99

Outline

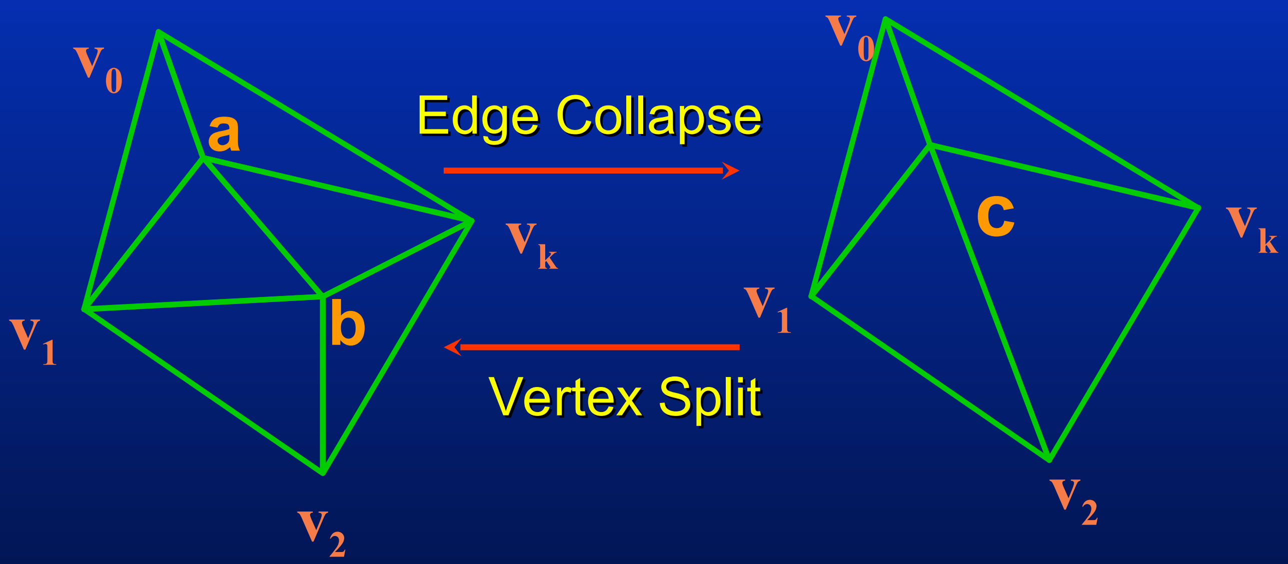
- Geometry and Topology Simplifications
- Implementing View-dependent LODs
  - Explicit and Implicit Dependencies
  - Maintaining triangle strips
- Variable-Precision Rendering

SAN ANTONIO SIGGRAPH 2002

Mesh Folding Problem

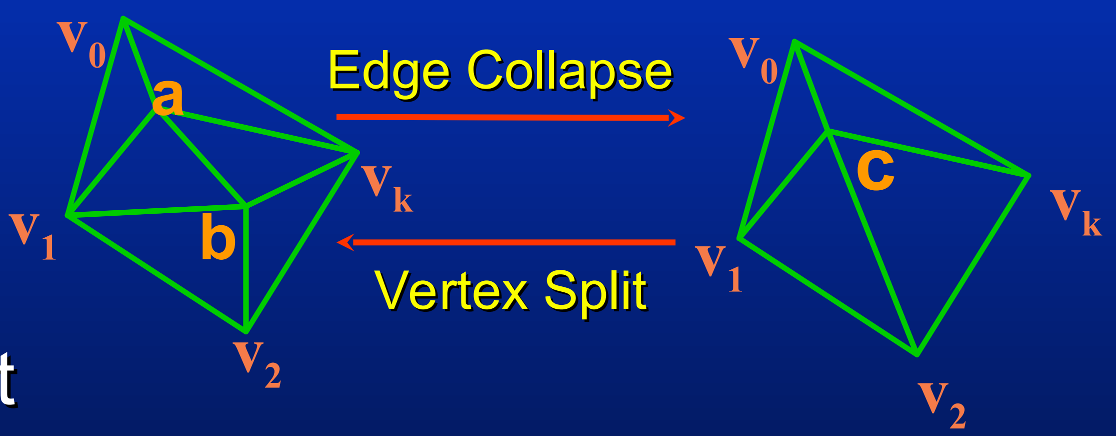


Explicit Dependencies



Neighborhood of an edge collapse is determined and fixed during preprocessing and used for validity checks at run-time

Explicit Dependencies



- Vertex split
  - Vertex *c* can split to *(a, b)* only if vertices v_0 , v_1 , … v_k are present and adjacent to *c* at run-time.
- Vertex-pair collapse
  - Vertex-pair *(a, b)* can collapse to vertex *c* only when all the vertices v_0 , v_1 , … v_k are present and adjacent to *(a, b)*.

SAN ANTONIO SIGGRAPH 2002

Implicit Dependencies

- Observations
  - Collapsibility graph is a Directed Acyclic Graph
  - Validity check involves determining the age of a node relative to its neighbors
- Solution
  - Each node is assigned a unique integer as *id*
  - Assign new nodes progressively increasing id-number

Implicit Dependencies

- Vertex ***v*** can split if:
  - Its *id* is greater than the *id* of all its neighbors

- Vertex-pair ***(u, v)*** can collapse to ***w*** if:
  - ***w*** 's *id* is less than the *id* of the parents of the neighbors of the two vertices (*u*,*v*)

Implicit Dependencies

- Vertex needs to maintain only two values:
  - Max ID of all its neighbors
  - Min ID of parents of all its neighbors
- Run-time checks become constant time
  - check against the above two values instead of all neighbors
- Localized memory accesses
  - Eg: Stanford Dragon (871K triangles, 874K nodes)
    - avg memory access distance for dependency checks comes down to ~1 byte from 14 MB

Outline

- Geometry and Topology Simplifications
- Implementing View-dependent LODs
  - Explicit and Implicit Dependencies
  - Maintaining triangle strips
- Variable-Precision Rendering

Recent Research on Triangle Strips

Akeley, Haeberli, Burns, *1990*

Arkin et al. Visual Computer 96

Evans, Skiena, Varshney, *Visualization 96*

Duchaineau et al., Visualization 97

Gumhold & Strasser, *Siggraph 98*

El-Sana, Azanli, Varshney, Visualization 99

Xiang, Held, Mitchell, I3D *99*

Deering, *Siggraph 95*

Bar-Yehuda & Gotsman, ACM TOG 96

Chow, *Visualization 97*

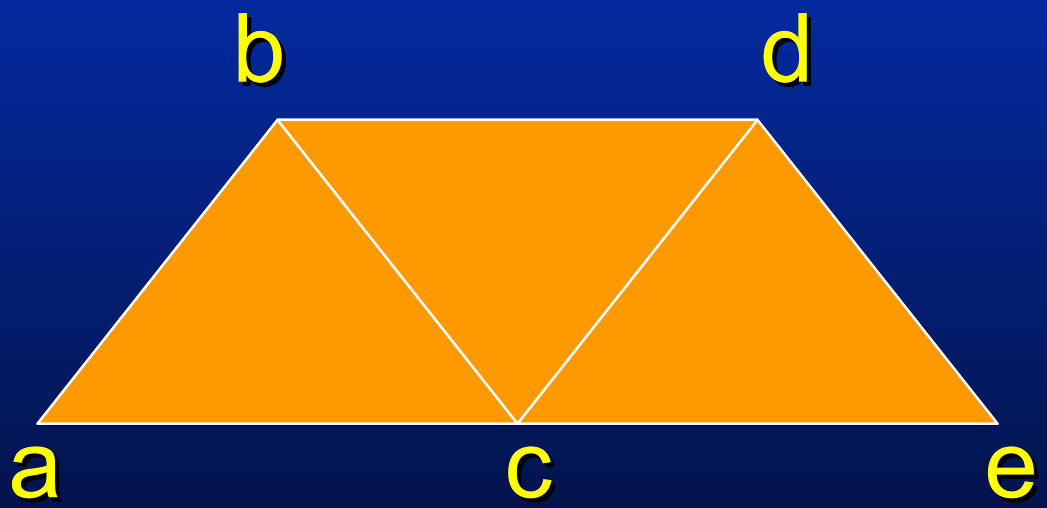
Speckmann & Snoeyink, *CCCG 97*

Taubin et al., *Siggraph 98*

Hoppe, *Siggraph 99*

Velho et al., Visual Computer 99

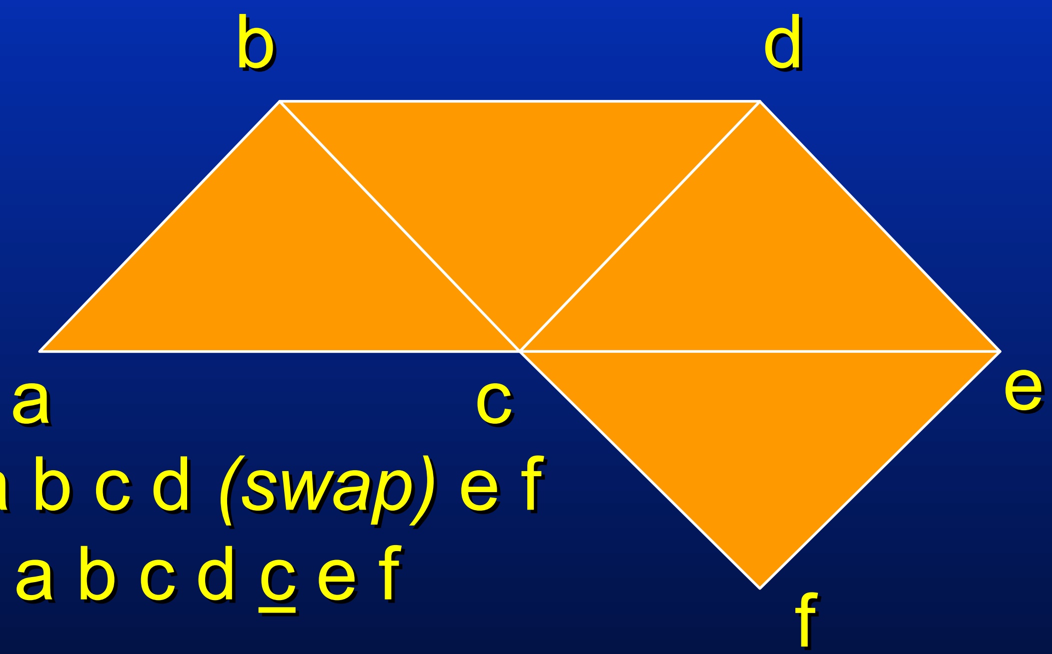
Triangle Strip



Triangles: (abc), (bcd), (cde)

Triangle Strip: abcde

Generalized Triangle Strips



Triangle Strip: a b c d *(swap)* e f

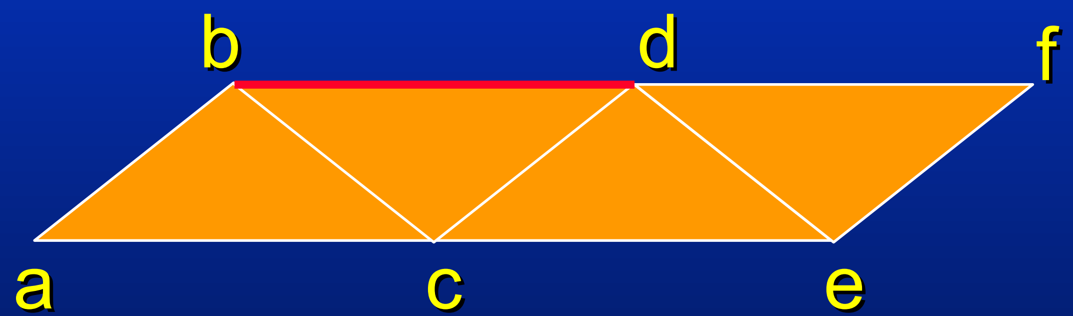
a b c d <u>c</u> e f

Repeating vertices changes direction

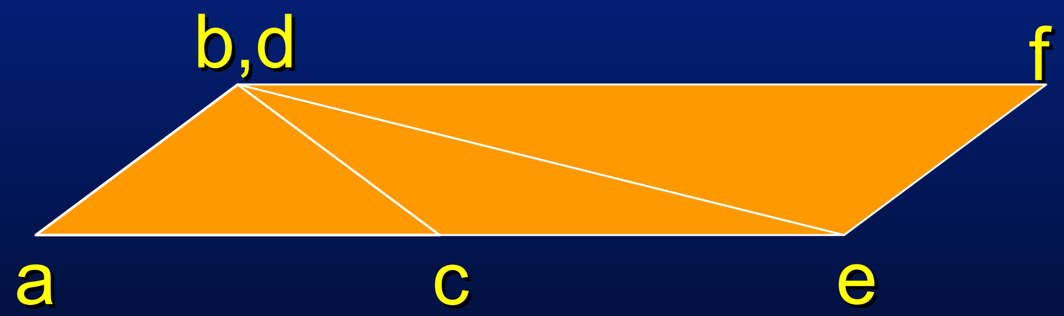
Triangle Strips with LODs

- Triangle strips
  - 2X speedup
  - Hardware / Software support
- Discrete LODs
  - Off-line computation of triangle strips per LOD
- View-dependent simplification
  - Connectivity changes every frame
  - Requires run-time update of triangle strips

Edge Collapse in a Triangle Strip



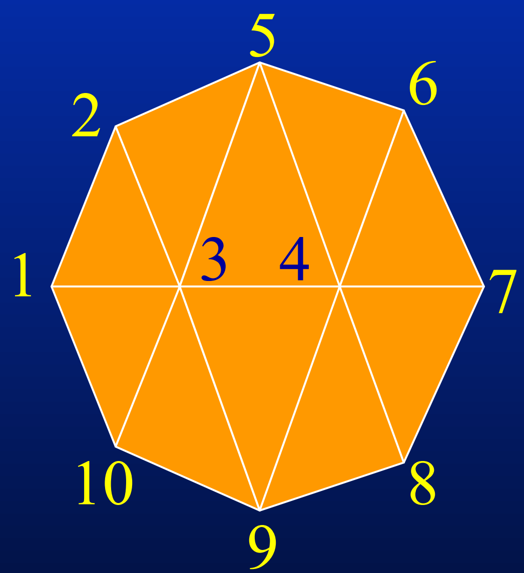
(abcdef)



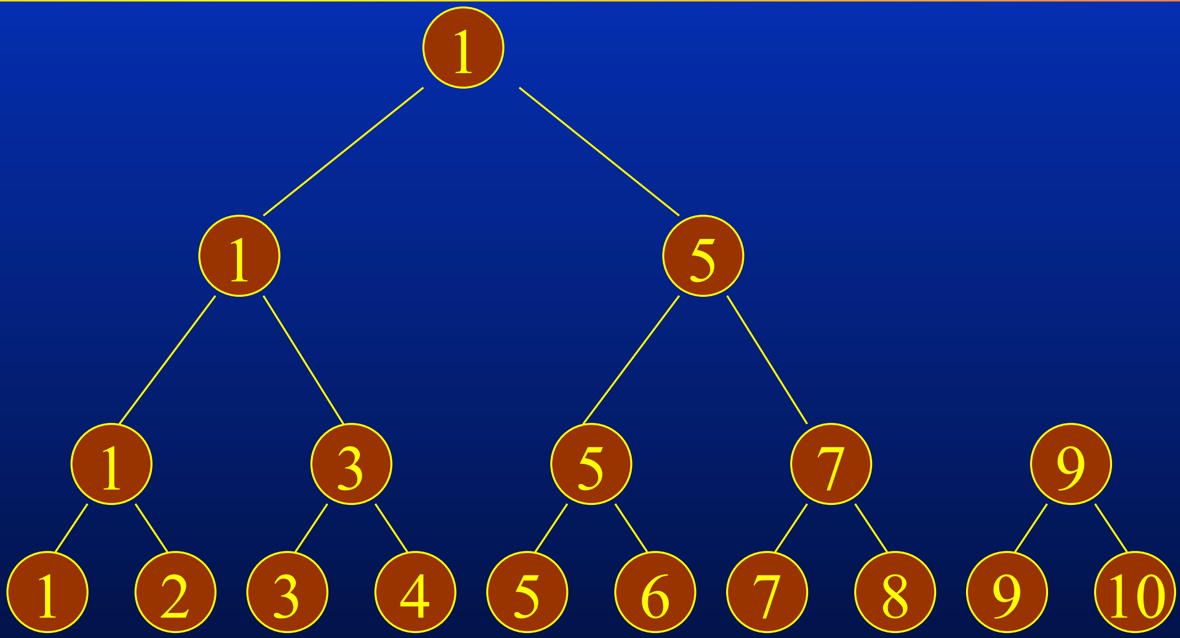
(abcbef)

Repeating vertices can represent edge collapses

Merge Tree

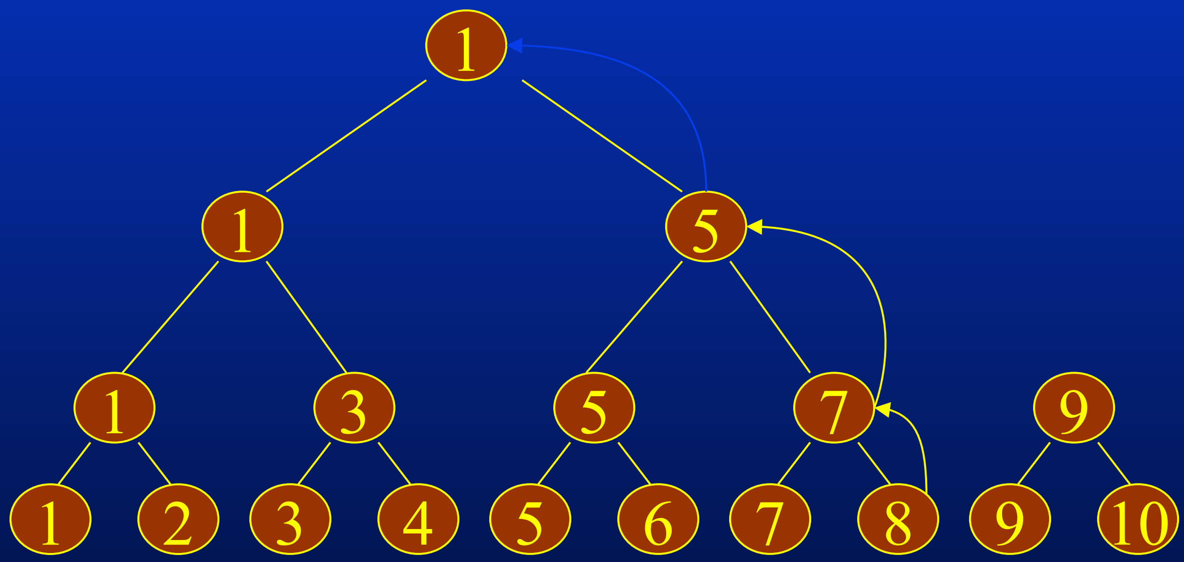


Mesh



Merge Tree

Following Parent Pointers

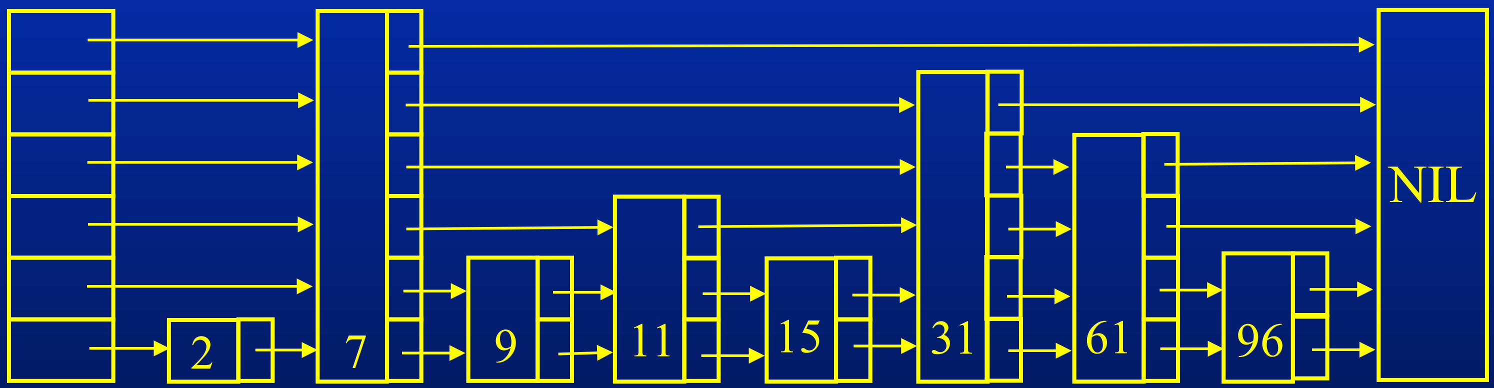


- Replace each Triangle Strip vertex by its closest active ancestor
- Need efficient pointer hopping

GVIL

SAN ANTONIO SIGGRAPH 2002

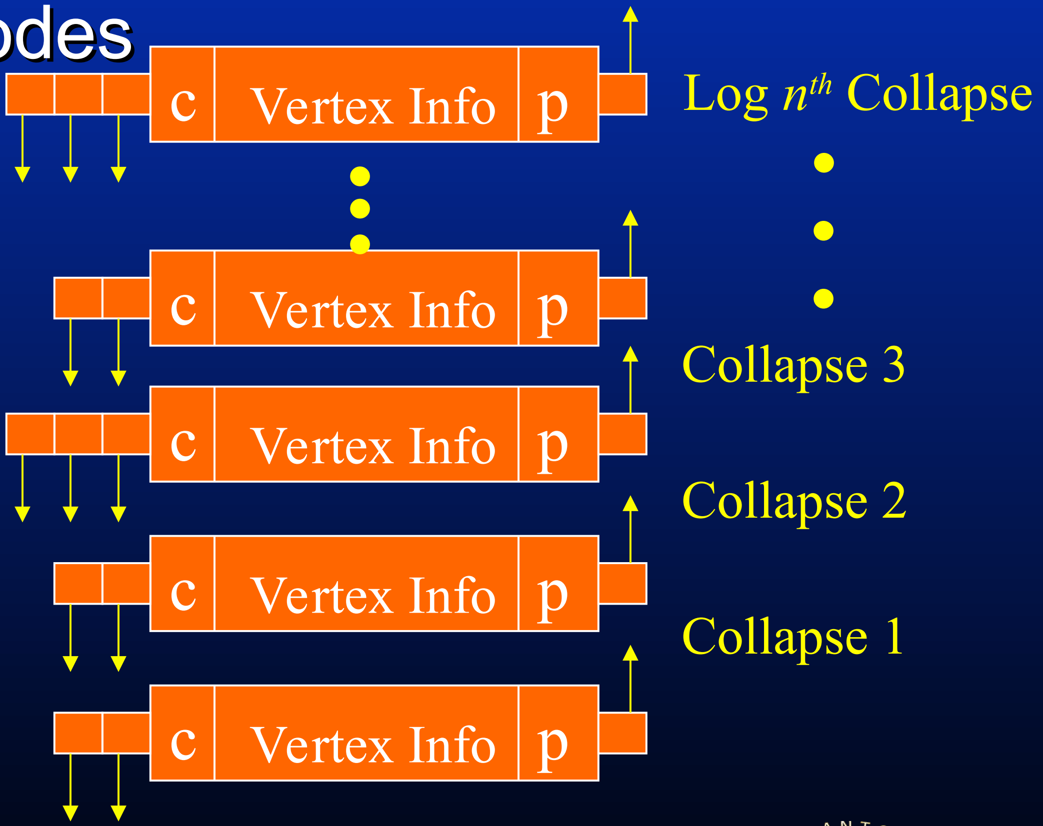
Skip Lists



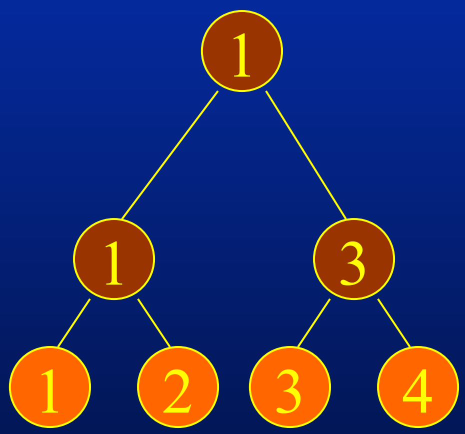
- Pugh [CACM 1990]
- Probabilistic balancing
- Fast searches
- Compressed trees

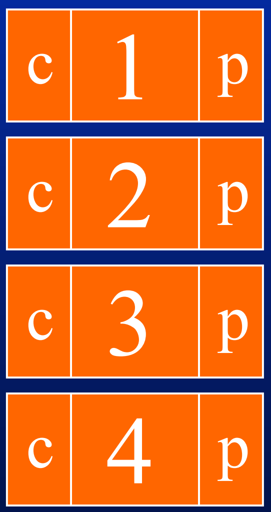
Skip Strip Data Structure

- Array of Skip Strip Nodes
- Merge
  - Increment p
  - Increment c
- Split
  - Decrement p
  - Decrement c



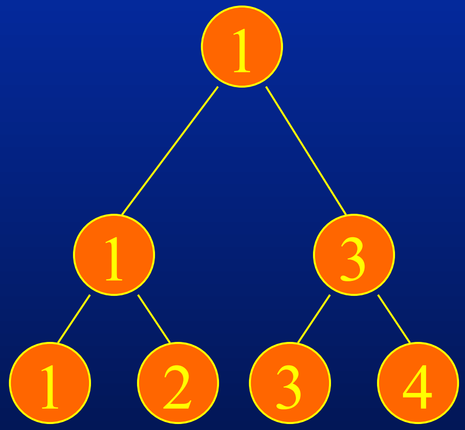
Building a Skip Strip

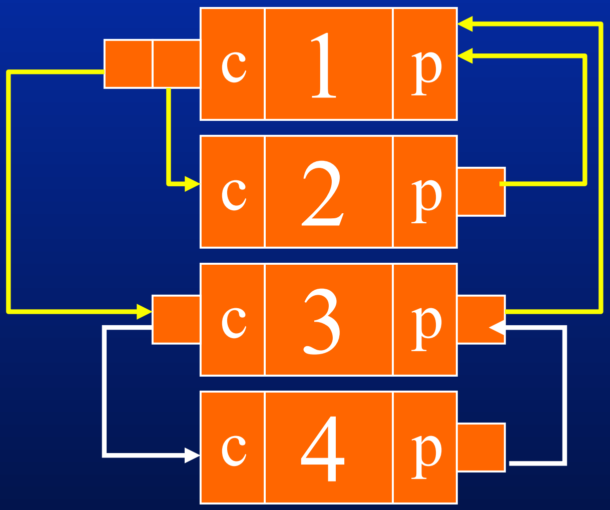




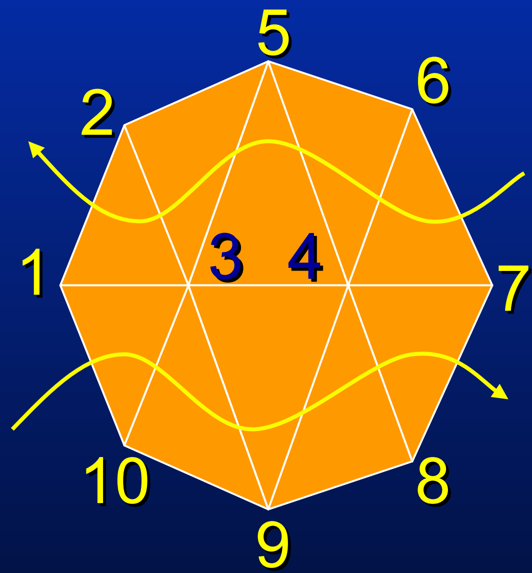
Building a Skip Strip

Building a Skip Strip





Skip Strips with LODs

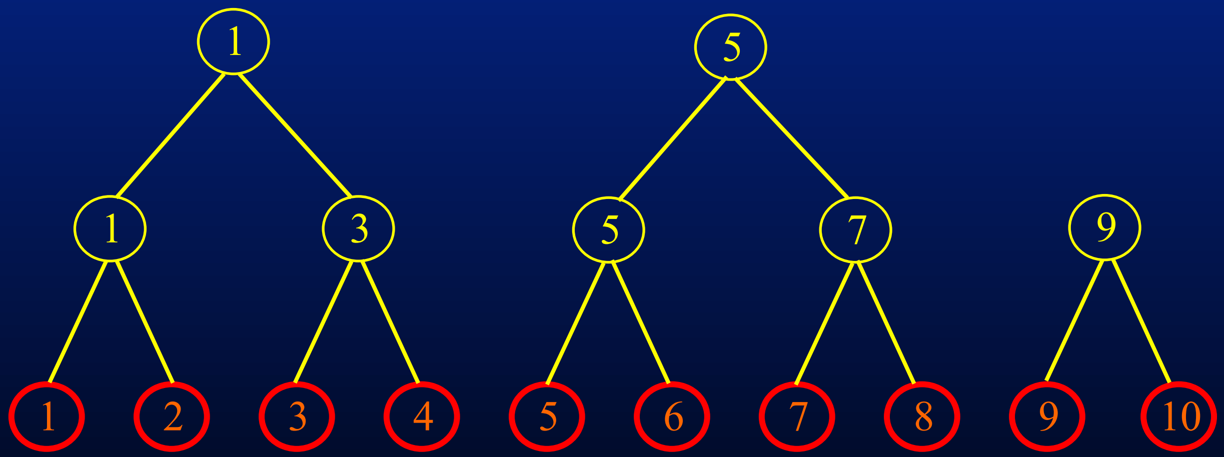


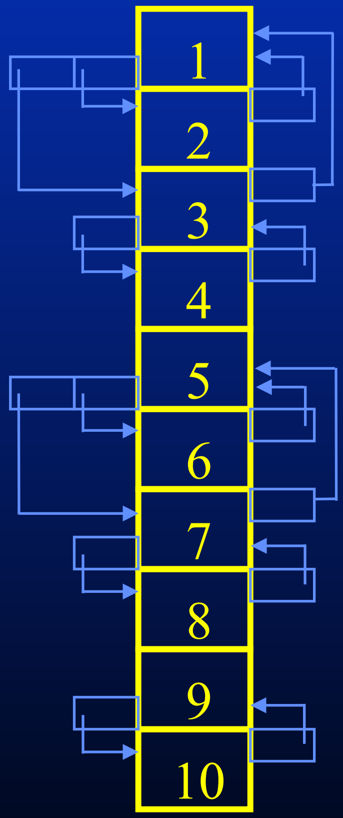
Triangle Strip A (7 6 4 5 3 2 1)

Triangle Strip B (1 10 3 9 4 8 7)

Skip Strip Example

Triangle Strip A: 7 6 4 5 3 2 1
Display Strip A: 7 6 4 5 3 2 1
Triangle Strip B: 1 10 3 9 4 8 7
Display Strip B: 1 10 3 9 4 8 7

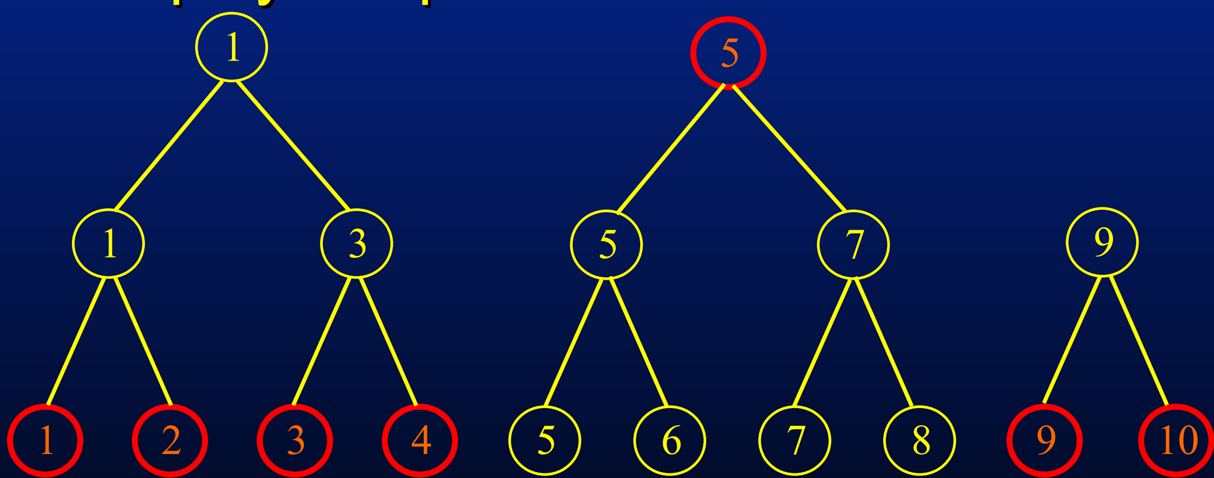




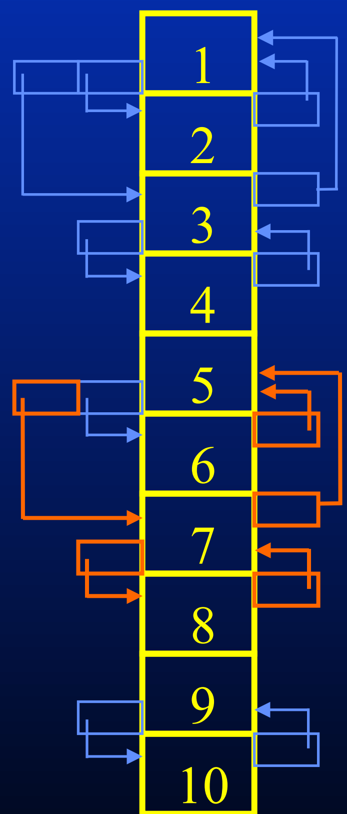
Highest Resolution

Skip Strip Example

Triangle Strip A: 7 6 4 5 3 2 1
Display Strip A: 5 5 4 5 3 2 1
Triangle Strip B: 1 10 3 9 4 8 7
Display Strip B: 1 10 3 9 4 5 5

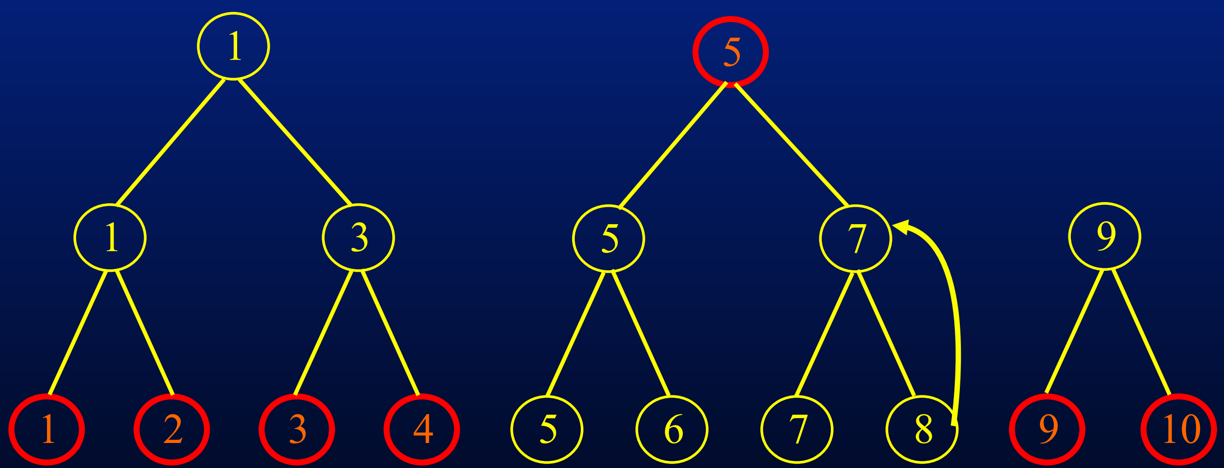


Lower Resolution

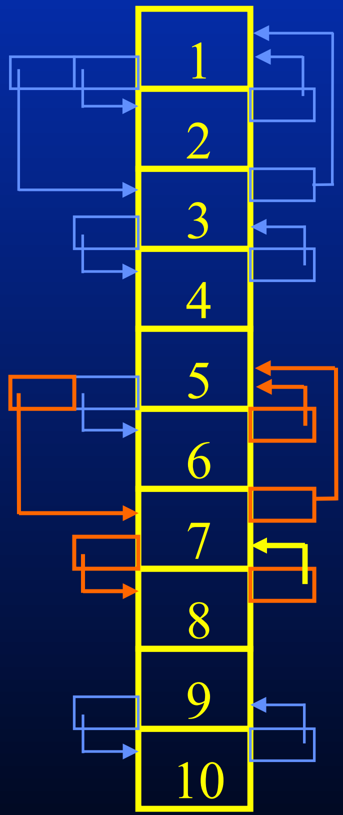


Skip Strip Example

Triangle Strip A: 7 6 4 5 3 2 1
Display Strip A: 5 5 4 5 3 2 1
Triangle Strip B: 1 10 3 9 4 8 7
Display Strip B: 1 10 3 9 4 5 5

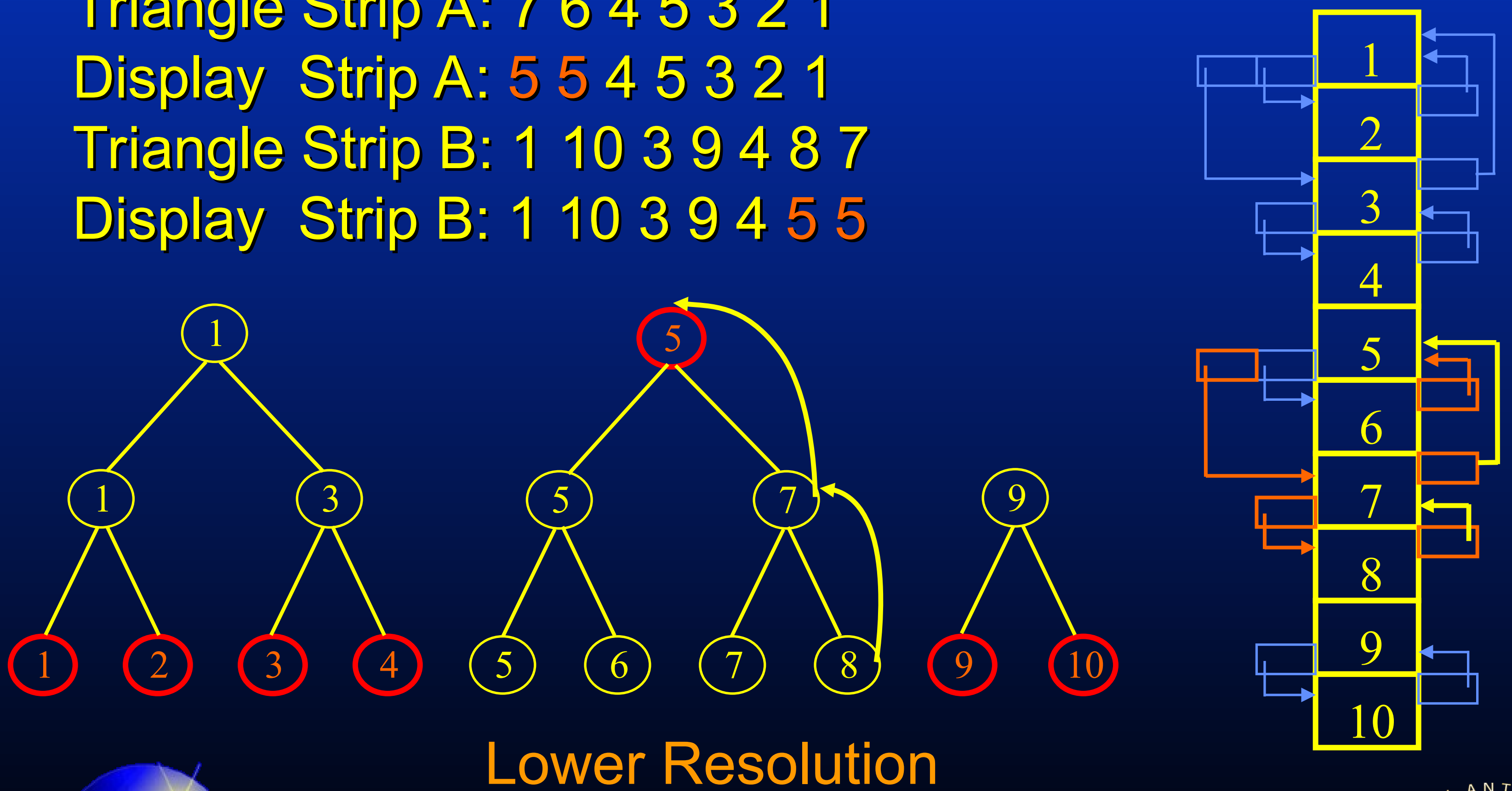


Lower Resolution



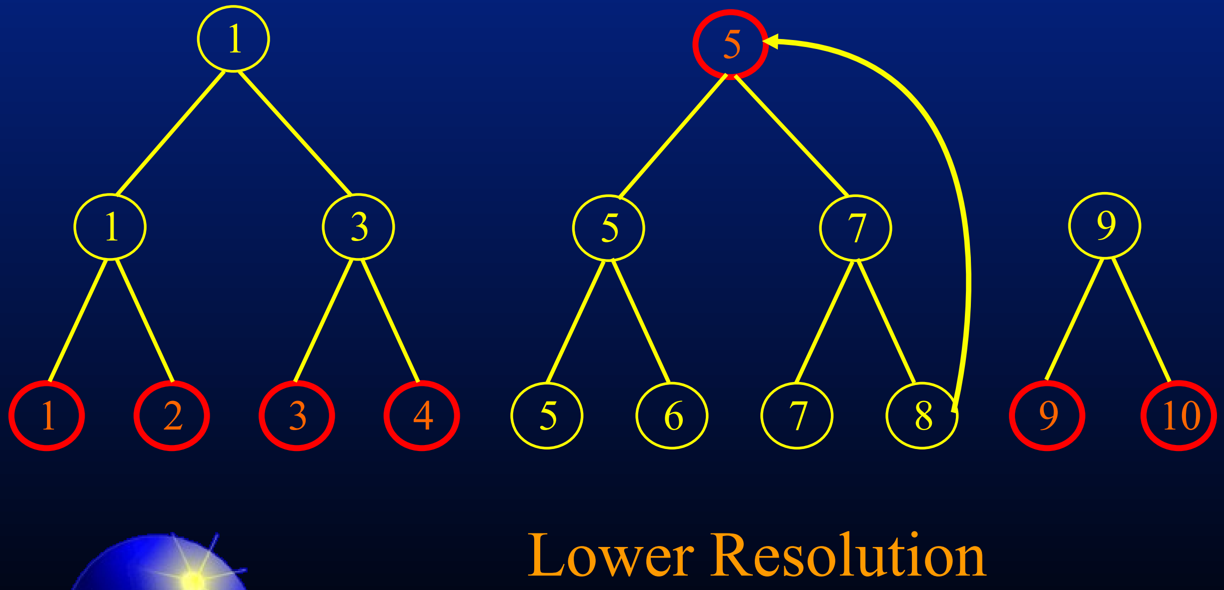
Skip Strip Example

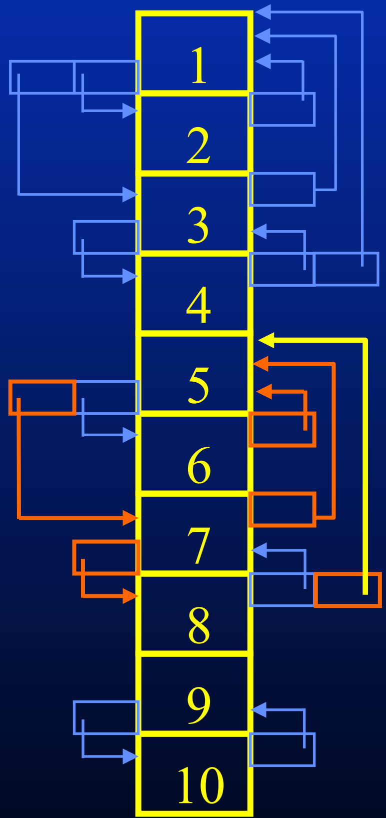
Triangle Strip A: 7 6 4 5 3 2 1
Display Strip A: 5 5 4 5 3 2 1
Triangle Strip B: 1 10 3 9 4 8 7
Display Strip B: 1 10 3 9 4 5 5



Optimized Skip Strip Example

Triangle Strip A: 7 6 4 5 3 2 1
Display Strip A: 5 5 4 5 3 2 1
Triangle Strip B: 1 10 3 9 4 8 7
Display Strip B: 1 10 3 9 4 5 5





SAN ANTONIO SIGGRAPH 2002

Real-Time Display

- Determine the display vertices
- Determine the display strips
  - Determine which strips have changed
  - Traverse the changed triangle strips
  - Follow Skip Strip pointers to get appropriate ancestors
  - Remove redundant vertices

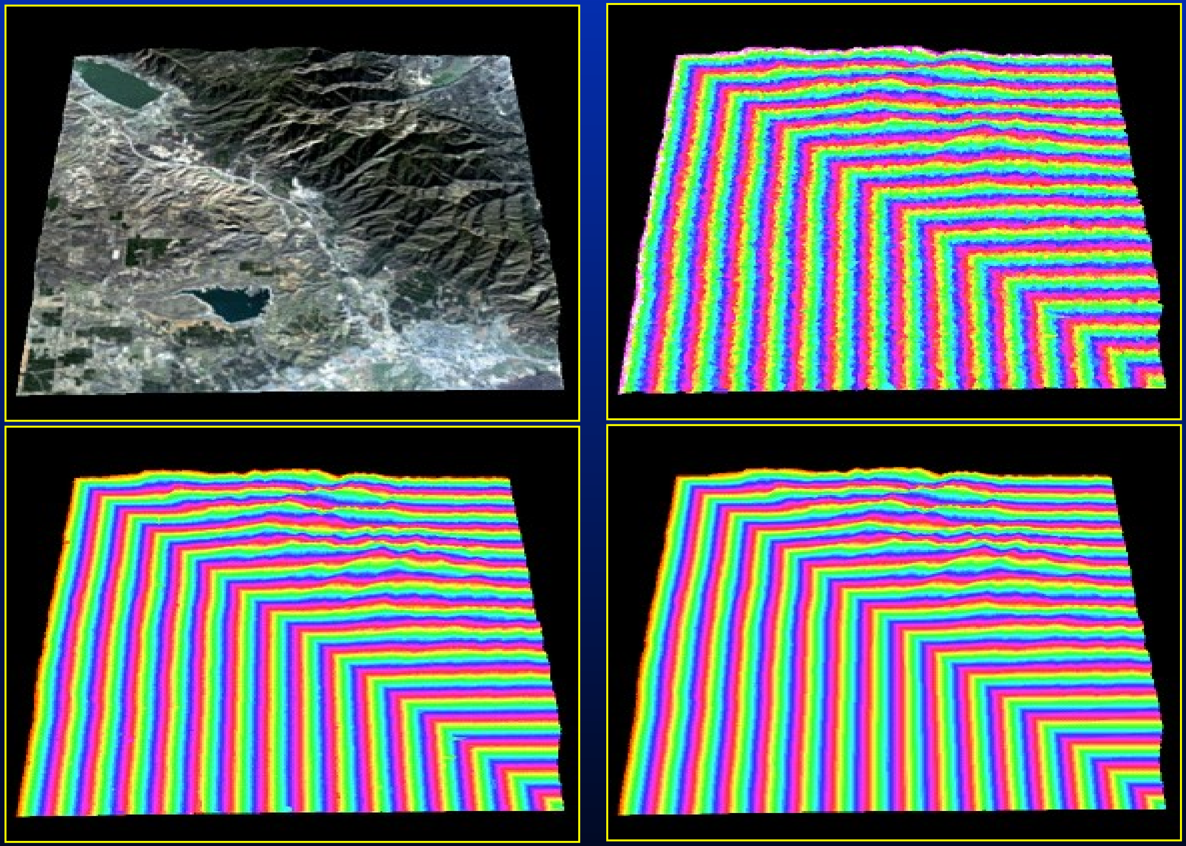
Efficient Skipping

- Reduce the traversed non-active vertices
  - Compress the traversed paths
  - Update the compressed paths in lazy fashion
  - Cache the active path index
- Efficiency
  - We use only *O(log log n)* jumps

Results: Terrain

255K tris

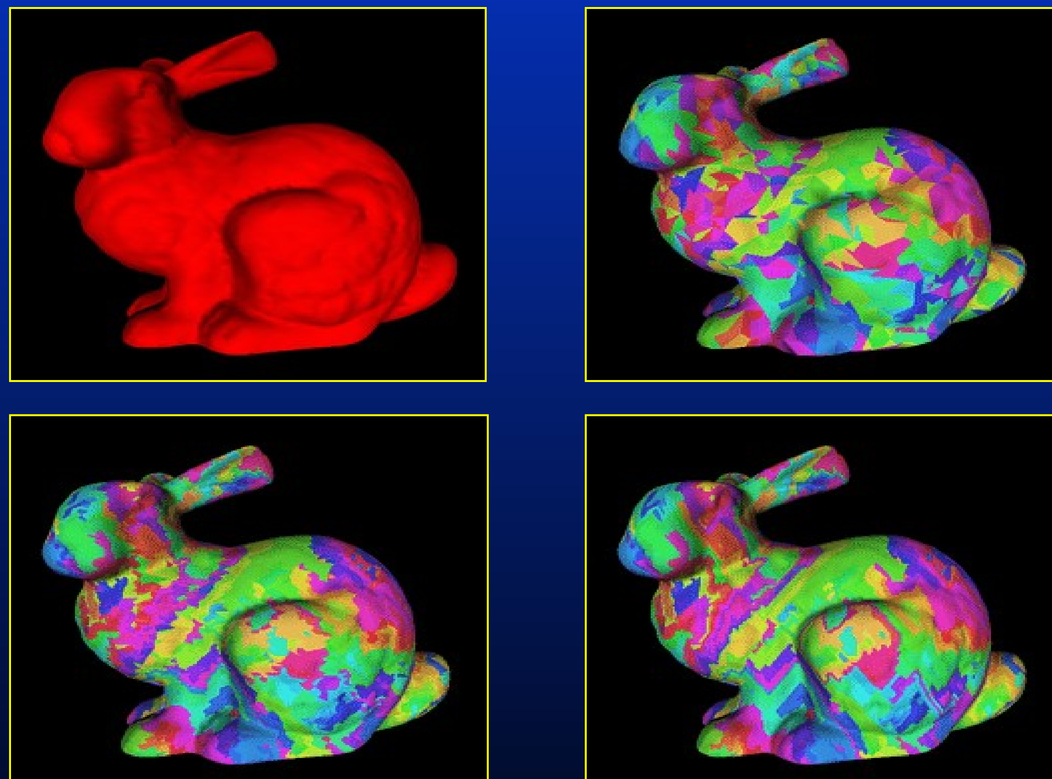
32K tris



255K tris

522K tris

Results: Bunny



30K tris

5K tris

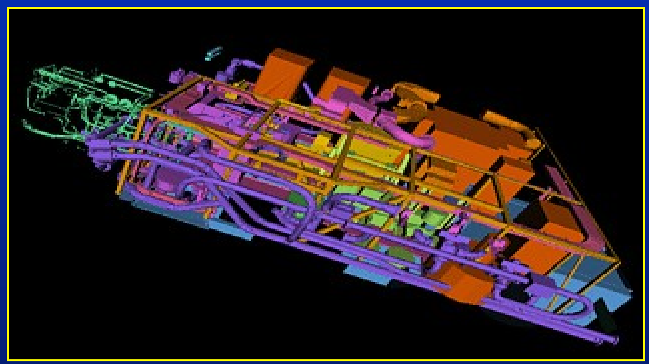
30K tris

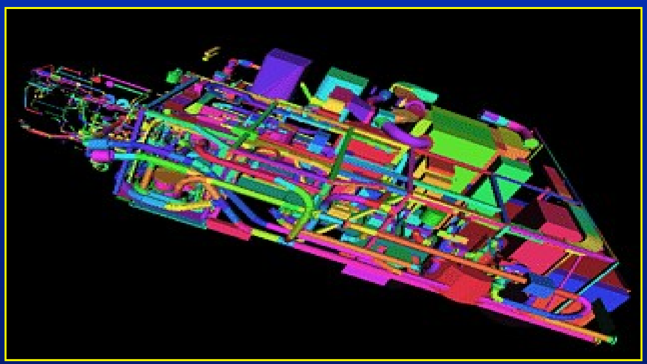
65K tris

El-Sana and Varshney, Visualization 99

Auxiliary Machine Room

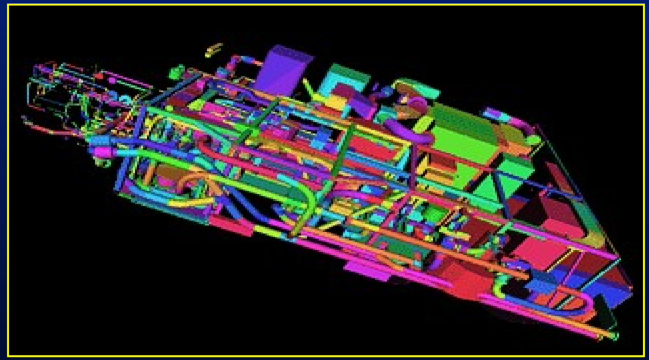
65K tris

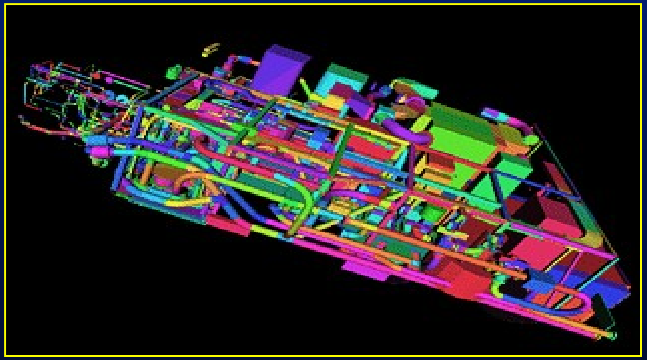




65K tris

170K tris





340K tris

El-Sana and Varshney, Visualization 99

Results

- Skip Strips make execution of split and merge operations more efficient
- Applicable to any hierarchical vertex scheme
- Four our four sample datasets:
  - Skip Strips provided ~ 1.5 to 2.0 X improvement over sending raw view-dependent triangles
  - Skip Strips provided ~ 1.6 to 1.7 X improvement over per-frame greedy triangle strip generation
- *But we have not tested this with video memory …*

Outline

- Geometry and Topology Simplifications
- Implementing View-dependent LODs
- Variable-Precision Rendering

Defining Level of Detail

- *Number* of Primitives
- *Precision* of primitives
  - Colors (Heckbert *82*, Xiang *97*)
  - Normals (Deering *95*, Zhang & Hoff *97*)
  - Vertex coordinates (King & Rossignac *99*)

Variable-Precision Rendering

- Reduce the precision of graphics primitives
- Relate the number of bits of input precision for a given display accuracy
- Speedup 3D transformation and lighting by taking advantage of SIMD parallelism
- Explore spatio-temporal coherence

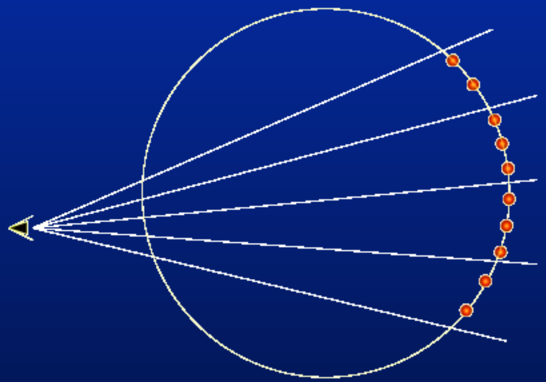
Related Work

Sugihara *89*
Milenkovic & Nackman *90*
Rossignac & Borrel *93*
Deering *95*
Fortune & Van Wyk *96*
Chow *97*
Luebke & Erikson *97*
Fortune *98*

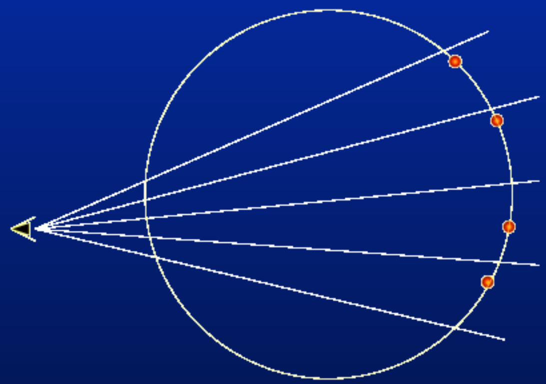
Taubin & Rossignac *98*
Taubin et al. *98*
Li & Kuo *98*
Cohen-Or et al. *99*
- Bajaj et al. *99*
- King & Rossignac *99*

Bajaj et al. *2000*
Pajarola & Rossignac *2000*

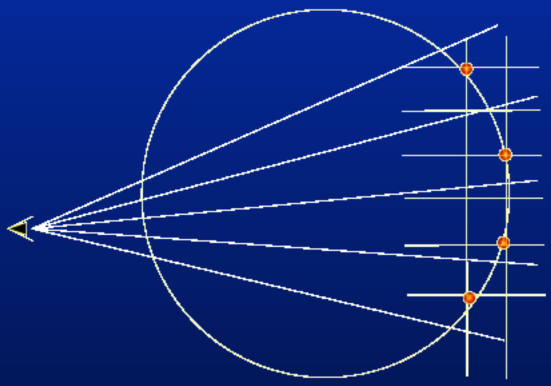
Variable-Precision vs. Multiresolution



Original



Multiresolution



Variable Precision

Assumptions

- Minimum-sized cube covering the object
  - x, y, z normalized to range [-1.0, 1.0]
- N-bit fixed-point representation of operands
- Rounding to the nearest integer
- Worst-case study

Error Analysis

- Representation error of a
  - Half bit, i.e., $\varepsilon_a \le \frac{1}{2}$
- Addition error of $(a+b)$

$$\varepsilon_{(a+b)} \le \varepsilon_a + \varepsilon_b \le \frac{1}{2} + \frac{1}{2} = 1, \text{ i.e., lose one bit of accuracy}$$

- Multiplication error of $(a \times b)$

$$\varepsilon_{(a \times b)} \le \left| a\varepsilon_b + b\varepsilon_a \right| \le \frac{1}{2} + \frac{1}{2} = 1, \text{ i.e., lose one bit of accuracy}$$

Error Analysis

- Division error of (a/b)

Generated error $\varepsilon_{(a/b)}^{gen} < 1$ due to truncation

Propagated error $\varepsilon_{(a/b)}^{prop} = \frac{\partial}{\partial a}\left(\frac{a}{b}\right)\varepsilon_a + \frac{\partial}{\partial b}\left(\frac{a}{b}\right)\varepsilon_b$

$$= \frac{\varepsilon_a}{b} + \frac{a}{b^2}\varepsilon_b \le \frac{1}{b} \qquad \left(\begin{array}{c}\varepsilon_a, \varepsilon_b < 1/2 \\ a < b\end{array}\right)$$

$$\varepsilon_{(a/b)} = \varepsilon_{(a/b)}^{gen} + \varepsilon_{(a/b)}^{prop} < 1 + \frac{\text{distance of far plane from eye}}{\text{distance of scene vertex to eye}}$$

Putting it all together

$$m = n + 3 + \left\lceil \log_2 \left(1 + \frac{\text{distance of far plane from eye}}{\text{distance of scene vertex to eye}} \right) \right\rceil$$

m is number of bits of input data

n is output accuracy after transformation

e.g., 1024×1024 window with pixel-level accuracy

object half-way across the view volume

$$n = 10$$

$$m = 10 + 3 + \lceil \log_2 (1 + 2) \rceil = 15$$

View-dependent Transformation

- Construct bounding volume hierarchy
- Find the projected size of the object
- Determine the nearest visible vertex accuracy

$$near\_bits = m - \left\lfloor \left| \log_2\left(\frac{\text{projected range}}{2}\right) \right| \right\rfloor$$

SAN ANTONIO SIGGRAPH 2002

View-dependent Transformation

Accuracy needed for each vertex:

$$vertex\_bits = near\_bits - \left\lfloor \log_2\left(\frac{\text{transformed W of the vertex}}{\text{distance of nearest vertex to eye}}\right) \right\rfloor$$

Compute by using bounding volume hierarchy

Spatio-temporal Coherence

- Spatial coherence
  - Using differences in neighboring vertices
  - $M\,x' = M\,(x + \Delta x) = M\,x + M\,\Delta x$
  - Top-down octree traversal
- Temporal Coherence
  - Frame-to-frame
  - $M'\,x = (M + \Delta M)\,x = M\,x + \Delta M\,x$
  - Can be combined with spatial coherence

Transformation Result

Floating Point
(32 bits/vertex coordinate)

Variable Precision
(7.9 bits/vertex coordinate)

Variable-Precision Lighting

$$
\begin{aligned}
Color = {} & emission_{mat} + ambient_{model} \times ambient_{mat} + \\
& \sum_{i=0}^{m-1} \left(\frac{1}{k_c + k_l d + k_q d^2}\right)_i \times (spotlight\_effect)_i \times \\
& (ambient_{light} \times ambient_{mat} + \\
& (\max\{\mathbf{L} \cdot \mathbf{N}, 0\}) \times diffuse_{light} \times diffuse_{mat} + \\
& (\max\{\mathbf{H} \cdot \mathbf{N}, 0\})^{shin} \times specular_{light} \times specular_{mat})_i
\end{aligned}
$$

Sources of Illumination Errors

- Operands with different accuracy
- Square-root operation error
- Specular exponentiation error

Illumination Errors

Operands a and b with different bits of accuracy: n (for a) and n' (for b) and $n > n'$

- Addition error of $(a+b)$

$$\varepsilon_{(a+b)} \le \varepsilon_a + \varepsilon_b \le 2^{-(n+1)} + 2^{-(n'+1)} \approx 2^{-(n'+1)}$$

i.e. same accuracy as the less accurate operand

- Multiplication error of $(a \times b)$

$$\varepsilon_{(a\times b)} \le \left| a\varepsilon_b + b\varepsilon_a \right| \le 2^{-(n+1)} \times 1 + 2^{-(n'+1)} \times 1 \approx 2^{-(n'+1)}$$

Same as in the addition case

Illumination Errors

Square-root operation error

- Fixed-point operation ⇒ table lookup
- 2n bits operand
- Most significant n bits as index

Illumination Errors

Specular exponentiation error of (a^{shin})

- Operand a with n bits accuracy, $shin < 128$
- Error maximized by large a, ε_a, and $shin$

$$(a+\varepsilon_a)^{shin} \approx a^{shin} + a\varepsilon_a \times shin \quad (if\ \varepsilon_a << a)$$

$$\varepsilon_{(a^{shin})} \approx a\varepsilon_a \times shin < 1\times 2^{-(n+1)} \times 128 = 2^{-(n-6)}$$

i.e., lose 6 bits of accuracy at most.

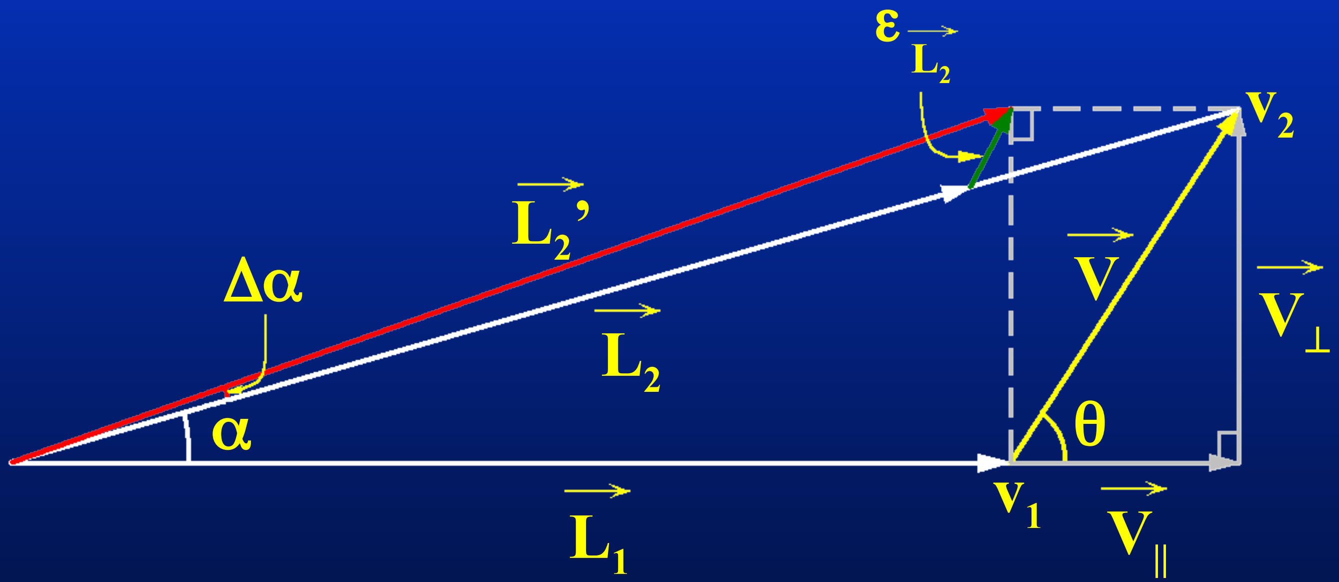
Illumination Errors

- Dot-product of vectors lose 2 bits accuracy
- Putting it all together
  - Specular (least accurate) decides the overall accuracy
  - lose 1 bit for normalization, 2 bits for dot product, 6 bits for exponentiation
  - Total loss of accuracy: 9 bits
  - So: $m = n+9$ *(n = output accuracy, m = input accuracy)*

GVIL

SAN ANTONIO SIGGRAPH 2002

Incremental Lighting



The error of using L_2' as an estimate of L_2 is in the order of $\left(\left\| \vec{V} \right\|^2 \Big/ 2 \left\| \vec{L_1} \right\|^2 \right)$

Implementation Notes

- Vertices processed in groups as a tradeoff between
  - L2 cache size
  - Expensive cost of resetting MMX register flag between changes in operand types
- Avoid error buildup
  - Matrix setup and composition per frame is full precision
  - Transformations are variable precision
  - Computation cost is negligible

GVIL

SAN ANTONIO SIGGRAPH 2002

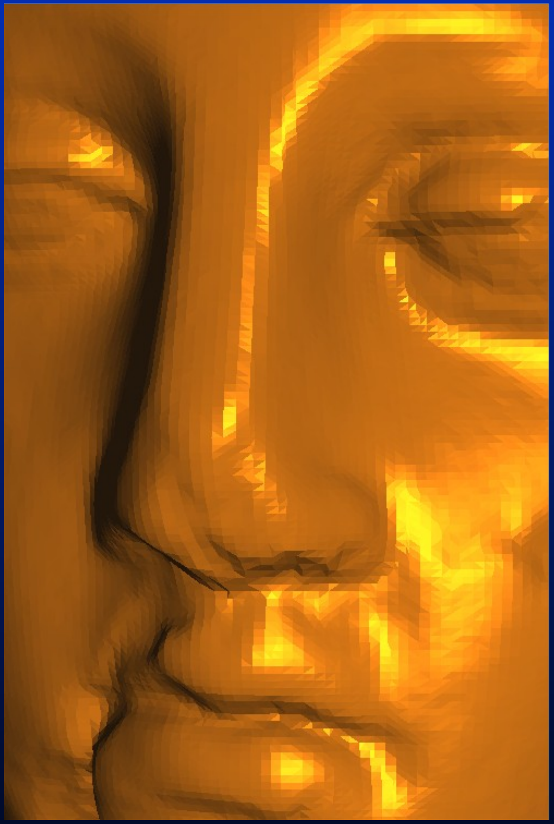
Results: Venus

Floating Point

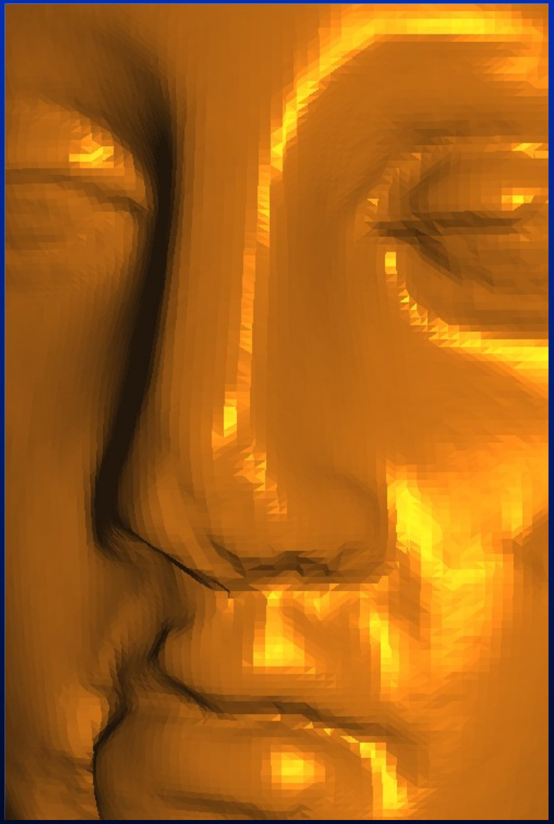
Variable Precision

Hao and Varshney, ACM I3D 2001

Results: Venus

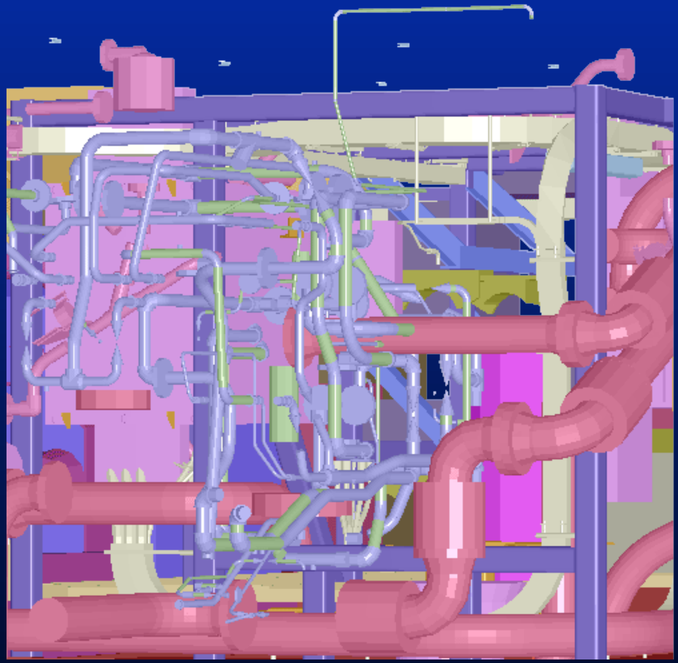


Floating Point

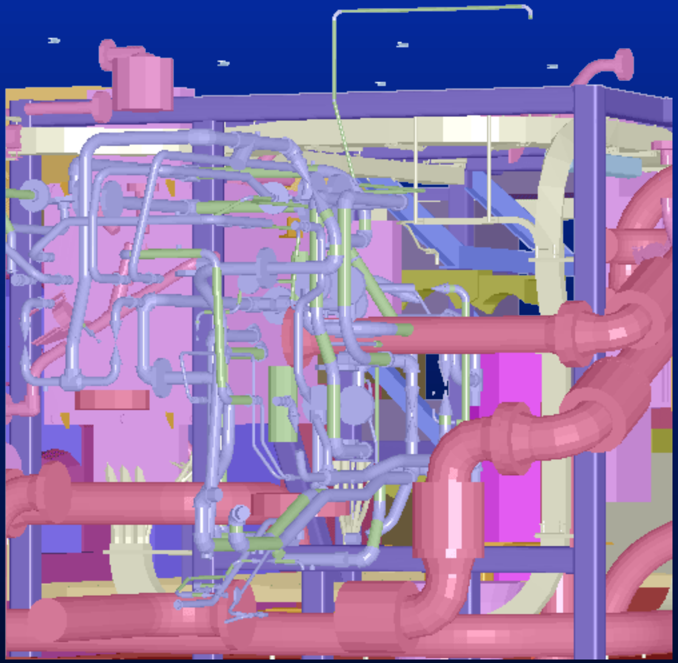


Variable Precision

Results
Auxiliary Machine Room



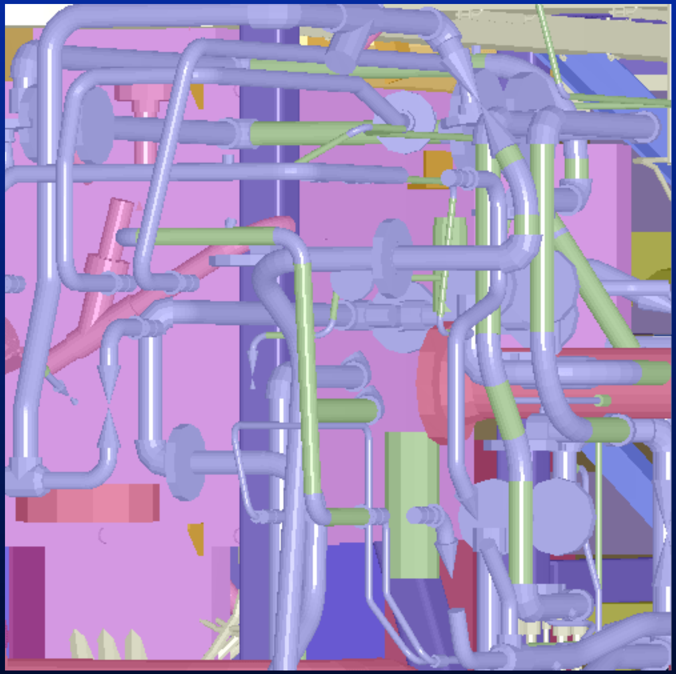
Floating Point



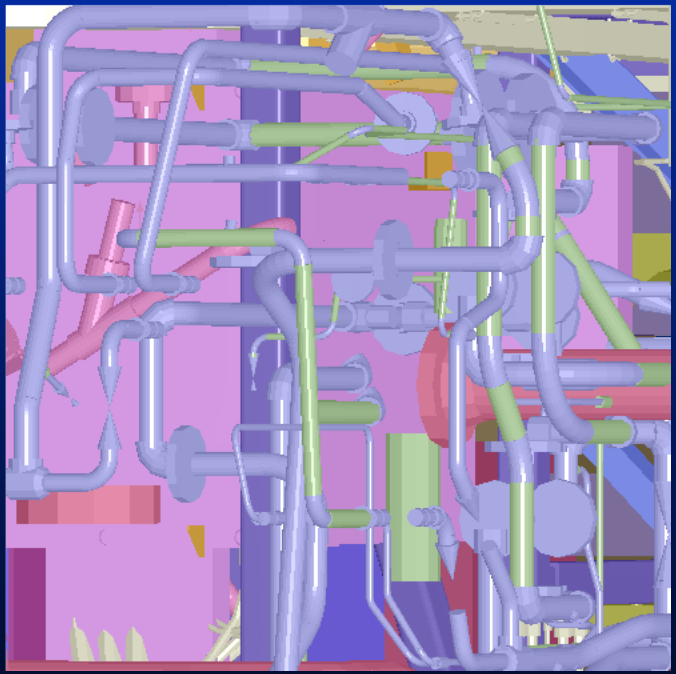
Variable Precision

SAN ANTONIO SIGGRAPH 2002

Results
Auxiliary Machine Room



Floating Point
Close-up



Variable Precision
Close-up

Conclusions

- More efficient transformation and lighting
- Complementary to multiresolution approaches
- For the datasets we tested
  - Using PII 400MHz PC with 128M RAM
  - Voodoo3 3500 graphics card and Glide API
  - Provides a factor of 4 or more speedup

Software

- http://www.cs.umd.edu/gvil/vpr.html
- Download free for non-commercial use

Conclusions

- Discrete and View-dependent LODs for simplifications of geometry and topology
- Implicit Dependencies for localizing data accesses
- Skip Strips: Updating triangle strips with view-dependent LODs
- Variable-Precision transformations and lighting

GVIL

SAN ANTONIO SIGGRAPH 2002

Acknowledgements

National Science Foundation

Dept of Defense

Honda R & D, North America

General Dynamics

UMIACS, University of Maryland

Stanford Graphics Lab

Cyberware, Inc.