3D for all!

Jarek Rossignac
GVU
Georgia Tech
Compressing and simplifying complex 3D polyhedra

Jarek Rossignac
GVU
Georgia Tech
What motivates you?

- Problem solving pleasures?
- Peer recognition?
- A hunger for knowledge?
- The need to impact society?
A vision for GVU

- Invent technologies that will make humans more effective in their professional, scholastic, and private activities
- Teach objectives, motivation, creativity, teamwork... the theory... and Java3D
- Work with industrials to understand where they are coming from and to help them decide where to go
How do we get there?

- Study humans and organizations
  - Activities/needs
  - Capabilities/limitations

- Explore technologies and inventions
  - R&D strategies and invention history
  - State-of-the art, possibilities, limitations
  - Hands-on experience/use what you create

- Understand commercialization
  - Market forces and acceptance issues
  - Engineering and testing for usability
  - Development and maintenance
Some Examples

- Multimedia capture of courses
- Mobile visual communication devices
- Intelligent appliances
- VR treatment of phobia
- Collaborative CAD
- Data mining
- Visualization
How big is 3D graphics?

- All graphic adaptors will be 3D enabled
- Next “must” after mouse, color, www
- Perspective = Detail + Background
- Intuitive
- Appealing
- Drives huge markets
What is it good for?

- 3D models of
  - terrain, underground cavities
  - homes, offices, buildings, cities
  - factories, assembly lines, robots
  - airplanes, cars, ships
  - consumer products
  - human organs, molecules
  - engineering simulation results
  - avatars, shopping malls, enemies
Why would I use it?

- Games, electronic commerce
- Design, PDM
- Design review, ergonomy
- Bids/part catalogs
- Communication/marketing
- Data understanding
- Intuitive navigation/selection
- Training/therapy
Why can’t I?

- Complex models
- Accessed through internet/phone
- Slow connections
- Limited rendering speed and storage
- Difficult user interface
What is a 3D graphic model?

- Representation of 3D geometry
- Surfaces and lighting model
- 3D images, textures
- Sampling of 4D light field
- Procedure + parameters for 3D graphics
What is a 3D graphic application?

 Representation
 - Geometry: vertex coordinates
 - Topology: triangle/vertex incidence
 - Photometry: colors, normals, textures

 Architecture of a 3D system

- design
- measure
- control
- manipulate
- 3D model
- display
- simulate
- archive
- transfer
What makes it effective?

- Fast access to 3D databases
- Internet connectivity
- Intuitive view and model manipulation
- Realtime feedback
The human needs?

- I want to access the data!
  - Compression
  - Progressive transmission

- I want to see what I am doing!
  - Simplification, graphic acceleration

- I want to be in control!
  - Virtual camera
Why focus on triangles?

- Optimized rendering systems
- Easily derived from polygons and surfaces
- Good measure of graphic complexity
  - Twice more triangles than vertices

add 1 vertex and 2 triangles
How many do you need?

- A sphere = 12x24x2 triangles
- Airplane = 50,000,000 triangles
- A city?
- A human body?
How much space do they take?

- **Geometry**: 3 coordinates per vertex
- **Topology**: 3 vertex indices per triangle
- **Photometry**: Normals, textures, colors
- **Total**: up to a 100 bytes per triangle
Is that a problem?

- A 100MT model takes:
  - A month to download over the phone
  - An hour to display

- Complexity will grow faster than bandwidth and graphic performance
How to cope with this complexity?

- Don’t bother with unimportant details
- **Compress**
  - Helps with transfer, storage, paging
- Eliminate redundant graphic operations
  - Visibility, back-face culling, triangle-strips
- Trade accuracy for performance
  - Lossy compression, simplification, images
How good is compression?

- **Lossless compression**
  - Preserves geometry and topology
  - Reduces # of bits for triangle/vertex incidence
  - Does not compress vertex representation

- **Current approach (Taubin & Rossignac)**
  - Cut surface into tree of corridors
  - Encode internal triangulation of corridors
  - 1.2 to 2.2 bits per Triangle
  - plus the vertex coordinates
Do I need full numeric accuracy?

- Models are approximations anyway
  - Imprecise measures or tolerated dimensions
  - Limited accuracy in geometric computation
  - Truncated coordinates to nearest float or double

- Use very short integers!
  - Floats are bad for geometry
    » Accuracy grows with distance to origin
    » Geometric models need uniform accuracy
  - Integers are perfect
    » Uniform accuracy throughout the model
    » More precise than floats for same number of bits
  - Don’t need 32 bits
    » 11 bits: 1/2mm for an engine block
Can this save more storage?

- **Lossy compression**
  - shorter representations of vertex coordinates
  - more frequent repetitions of values
  - efficient entropy coding schemes

- **Current approach (Taubin & Rossignac)**
  - Quantize vertex coordinates to 8-12 bits
  - Predict location of next vertex
  - Code difference (short corrective vector)
  - Use variable length coding (entropy)
  - Compresses down to about 6 b/T
Do I need all these triangles?

- Can’t see most of them
  - quickly discard invisible ones
- Many details are smaller than a pixel
  - remove them or use an impostor
- Surface tesselations are not optimal
  - Need accuracy at silhouettes
  - Could use coarser meshes elsewhere
How do I take advantage of this?

- Precompute approximations of features:
  - surface subsets, objects, groups
- Use approximations to accelerate graphics
  - when resulting visual error acceptable
- Current approaches
  - Coalesce clusters of vertices
    » How to form the clusters’ hierarchy?
    » Where to place the result of coalescing a cluster?
  - Remove degenerate triangles
  - Dynamically select LOD based on view
    » How to measure error?
    » How to allocate rendering time?
    » How to avoid artifacts?
Do I need training to use 3D?

- Why should you? Your kids got none.
- Few designers/many users
  - Scientists, doctors
  - Non technical professionals
  - Home shoppers
  - Game players
How do I control the view?

- You don’t want a HMD
- You can’t afford a CAVE
- The track ball will break your... enthusiasm
- The mouse is not intuitive
- Use a virtual camera!
What is a virtual camera?

- 3D model
- (Large) screen
- Table with drawing or global view
- Move small camera over the table
- See in 3D what you are pointing at
Outline of the presentation

- Geometric and topological compression
  - Taubin & Rossignac
- Grid-based vertex coalescence
  - Rossignac & Borrel
- Edge-collapse vertex coalescence
  - Ronfard & Rossignac
- Real icon for a virtual camera
  - Rossignac & Wolf
How do I measure the error?

- **Geometric 3D deviation**
  - View-independent
  - Allowed by model tolerance

- **Image space error**
  - View-dependent
    - Silhouettes, color/shading variation
  - Bounded by projection of 3D error

- **Color error**
  - View dependent
  - Error on color reflected by surface point
  - Depends on surface orientation
  - Not important for most applications
Hausdorff error

- $H(A, B) = \max(d(a, B), d(b, A))$
  - for all $a$ in $A$ and $b$ in $B$

- Radius of largest ball
  - that has its center on one surface and
  - that is disjoint from the other surface

- Expensive to compute
  - Need not happen at vertex or edge
3D Compression

- Geometric Compression through Topological Surgery
  - Gabriel Taubin and Jarek Rossignac
  - IBM Research Report RC 20340,
  - Revised 7/1/97
Mesh representations and storage

Independent triangles: 36 B/T
- 9x4 B/T

Vertex and mesh tables: 18 B/T
- 3x4B/V + 3x4B/T

Triangle strips: 13.4 B/T
- 1.1x3x4B/T + 1B/S + 1b/T
Constructing and encoding corridors

Given the boundary and the starting point, we need 1 bit/triangle to encode a strip.

Connecting vertices in a spiral cut defines the boundary of a corridor (both sides).

Store vertices in their order along the spiral.
Store one L/R bit per triangle (except first).
Wait a minute!

- The corridor may have warts!
- The spiral may cut itself or split!
- The corridor may bifurcate!
Compression details

- Compress trees using runs
  - Cut = Vertex spanning tree (zipper)
  - Corridors = Dual binary tree

- Encode interior triangulation of corridor
  - 1 bit per triangle
Lossy geometric compression

- Quantize all vertex coordinates
  - Desired precision
  - Consistent with modeling tolerance
- Use tree ancestors to predict next vertex
  - Optimal linear combination of ancestors
  - Store corrective vectors (error)
- Use variable length coordinates
  - Coordinates of correction are small integers
  - Lots of values repeated for complex models
  - Use short codes to frequently used values
  - Need between 3 and 10 B/T
    » depending on tolerance
Implementation

- VRML-2 compression/decompression
  - G. Taubin, P. Horn, F. Lazarus (IBM)
  - Compressed models: average of 1 B/T
    » Depends on complexity, smoothness, tolerance
    » Example: 160K triangle model
      » Tolerance: 0.0005 of model size (11 bit quantization)
      » Incidence: 1.2 b/T
      » Geometry: 5.4 b/T
  - Decompression speed: 60K T/sec
Other compression techniques

- Deering (Siggraph’95)
  - Extend triangle strip
    - Maintain buffer of 16 vertices for reuse
    - Op-code for building next triangle: 4-6 b/T
    - Vertex quantization and variable length coding
      - Local coordinate system
  - Good for graphics:
    - In-line fast decoding and small local storage

- Hoppe (Siggraph’96)
  - Store vertex-split operations
    - Index to vertex
    - Index to 2 adjacent edges
    - Displacement(s) from common vertex to new pair
      - Short vectors (variable length coding)
Model simplification

- Multi-resolution 3D approximations for rendering complex scenes
  - Jarek Rossignac and Paul Borrel
  - Geometric Modeling in Computer Graphics’93

- Full-range approximations of triangulated polyhedra
  - Remi Ronfard and Jarek Rossignac
  - Eurographics’96 (Computer Graphics Forum)
Multi-resolution models

- Show distant features at lower resolution
- Split the model into small features
  - Use design entities (solids or groups)
  - Split large and complex solids
- Precompute several levels of detail per feature
  - Logarithmic reduction of triangle count
  - Guaranty error bound
- Select appropriate resolution for each feature
  - Perspective projection of error estimate
  - Position on screen, velocity
- Merge groups of small distant features
  - Preserve overall volume and color?
Vertex merge (Rossignac&Borrel)

- Cluster vertices using grid
  - Compute truncated coordinates for each vertex
  - Use them as a cluster identifier

- Coalesce vertex clusters
  - Compute location of representative vertex
  - Associate each vertex with its new location

- Remove degenerate triangles
  - Those with 2 or 3 vertices in the same cluster

- Build display lists for each LOD
  - New/old normals and textures
  - Triangle strips
Pros and cons for Rossignac&Borrel

♦ **Advantages**
  – Much faster than any other method
  – Very robust (no restriction on input data)
  – Simple code (implement in hours)
  – Guaranteed error bound (cell diagonal)
  – Reduces topological complexity (holes...)
  – Very effective for complicated details
  – Does not create holes of gaps in surface

♦ **Drawbacks**
  – Suboptimal for simplifying smooth flat surfaces
    » A vertex cannot travel more than the cell size
  – Simplified surface may self intersect
  – Hard to achieve precise triangle count
Extensions and variations

- Low&Tan, Interactive 3D Graphics’97
  - Sort vertices by importance in each cell
  - Attract vertices from neighboring cells

- Luebke&Erikson, Siggraph’97
  - Octree or other vertex clustering technique
  - Hierarchy of clusters
  - View defines which clusters are active
  - List of active triangles
  - Temporal coherence for adaptive LOD
  - Bad for T-strips and display lists
  - Use only for large, complex solids

- Popovic&Hoppe, Siggraph’97
  - Combine PM with Rossignac-Borrel’s proximity
Edge collapse (Ronfard&Rossignac)

- Coalescence vertices
  - that are further apart than the tolerance
  - as long as they slide close to the surface

- Efficient error estimate for a cluster
  - Max distance between representative vertex and all the planes supporting the incident triangles upon the vertices of the cluster

- Approach
  - Evaluate error associated with edge collapses
  - Maintain sorted list of candidate edges
  - Keep collapsing edges until reach
    » tolerance or
    » desired triangle count
Pros/cons for Ronfard & Rossignac

◆ Advantages
  – Better simplification ratios for same tolerance
  – Can control error or triangle count
  – Can choose whether to preserve topology or not

◆ Drawbacks
  – Requires maintaining incidence graph
  – Imposes topological restrictions on data
  – Slower than grid-based coalescence
  – Hard to achieve precise triangle count
Extensions and variations

- Garland & Heckbert, Siggraph’97
  - Use least square distance to supporting planes
  - Propagate only 4x4 matrix instead of all planes
  - Not a bound on the error!

- Hoppe, Siggraph’96
  - Store sequence of edge collapse operations
  - Use it’s inverse as a progressive mesh (PM)
  - Each split is defined by
    » 2 adjacent edges in the previous model
    » The relative position of the 2 new vertices
Conclusion

- Increasingly complex triangular 3D models
- Compress, simplify within tolerance
- Bit-efficient coding: 1 Byte/triangle
- Simplification for fast graphics:
  - Grid-based vertex coalescence
    » Fast, robust, effective, suboptimal
  - Edge-collapse
    » Slower, more precise, better results
- Need extractable encoding for
  - view-dependent multi-res transmission