Rapid Simplification of Multi-Attribute Meshes

Andrew Willmott
Maxis

HPG 2011
The Goal
Goal
Goal

~10 ms
Why? Real-time Domain

- Need to generate LODs for player-created models
- Must generate them while the game is running interactively
- Other demands for CPU, including generating the original high-res mesh and textures
Domain

- Specific example:
  - Start a level
  - Ask server for player creations
  - Expand descriptions into model geometry and textures
  - Generate LODs
  - Display world and player creations
<table>
<thead>
<tr>
<th>Goldwing Gunship</th>
<th>The great Harves</th>
<th>Harpy</th>
<th>Galactic Lung Dragon</th>
<th>Triocular Scorpio</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>MjrGlory</em></td>
<td><em>Kismet - Scar</em></td>
<td><em>CopperLou</em></td>
<td><em>AnUBISEL EasS</em></td>
<td><em>Parkaboy</em></td>
</tr>
<tr>
<td><img src="goldwing-gunship.png" alt="Image" /></td>
<td><img src="the-great-harves.png" alt="Image" /></td>
<td><img src="harpy.png" alt="Image" /></td>
<td><img src="galactic-lung-dragon.png" alt="Image" /></td>
<td><img src="triocular-scorpio.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pinguin Viscus Spik</th>
<th>Triocular Scorpio</th>
<th>gift for Ivonne</th>
<th>Night Rapton</th>
<th>Fineris</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>madnug</em></td>
<td><em>avaslash</em></td>
<td><em>RolandRyan</em></td>
<td><em>AnUBISEL EasS</em></td>
<td><em>RedDragon1958</em></td>
</tr>
<tr>
<td><img src="pinguin-viscus.png" alt="Image" /></td>
<td><img src="triocular-scorpio.png" alt="Image" /></td>
<td><img src="gift-for-ivonne.png" alt="Image" /></td>
<td><img src="night-rapton.png" alt="Image" /></td>
<td><img src="fineris.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inarml</th>
<th>Pink Unipeg</th>
<th>Shredder</th>
<th>Boba Fett</th>
<th>Triocular Scorpio</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Dustpuppy11</em></td>
<td><em>AnUBISEL EasS</em></td>
<td><em>FritoPie</em></td>
<td><em>FritoPie</em></td>
<td><em>Chopsholapoid</em></td>
</tr>
<tr>
<td><img src="inarml.png" alt="Image" /></td>
<td><img src="pink-unipeg.png" alt="Image" /></td>
<td><img src="shredder.png" alt="Image" /></td>
<td><img src="boba-fett.png" alt="Image" /></td>
<td><img src="triocular-scorpio.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Half Chicken-Half</th>
<th>Announcement!</th>
<th>Arctic Lion</th>
<th>Medusa</th>
<th>Systrous, Galactic</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>75Fire1Dragon</em></td>
<td><em>Chamaleon</em></td>
<td><em>flamestar</em></td>
<td><em>CopperLou</em></td>
<td><em>BlackSheep6059</em></td>
</tr>
<tr>
<td><img src="half-chicken-half.png" alt="Image" /></td>
<td><img src="announcement.png" alt="Image" /></td>
<td><img src="arctic-lion.png" alt="Image" /></td>
<td><img src="medusa.png" alt="Image" /></td>
<td><img src="systrous-galactic.png" alt="Image" /></td>
</tr>
<tr>
<td>Character</td>
<td>Name</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Queen Bee</td>
<td>Tars Tarkas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Einfin</td>
<td>Himmelslaub</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintenance Drok</td>
<td>Parkaby</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spring Fling Annoi</td>
<td>Tars Tarkas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ground Dragon</td>
<td>Cowboy Cleatus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baba Yaga's House</td>
<td>Starti2bartfast38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chinese Phoenix</td>
<td>AnUBISLEAs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unterwasserwelt</td>
<td>Chamaleon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Craby</td>
<td>sergio97</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Araya Armor Ripp</td>
<td>Mjr Glory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Breloom</td>
<td>Antarticas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIVE THE PATCH</td>
<td>Covest</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chinese Cloud Dra</td>
<td>AnUBISLEAs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emperor Auric'clii</td>
<td>Simago624</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turteltaub</td>
<td>Himmelslaub</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maker</td>
<td>Falcore</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fafnir</td>
<td>CopperLou</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blue Lagoon</td>
<td>Kevin92</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frost Demon Scor</td>
<td>doozerdude</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dlaleon</td>
<td>Shadowklaw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Worked Example

• Generate 20 meshes x 3 LODs
• Say 10% of CPU per frame

• 10s per LOD -> 1 hour 40 minutes
• 10 seconds -> 16 ms per LOD
Other requirements

- Robustness
  - Player-created meshes, not artist-created
  - No time for input cleanup passes
- Static LODs
  - Need to LOD shaders and animation too
  - Generate lowest LODs first
- GPU-friendly simplification
Previous Work

• Rossignac & Borrel [1993]
• Hoppe, Garland & Heckbert [1996-8]
• Out of core: Lindstrom [2000]
  • Massive meshes without thrashing
  • Vertex clustering and quadrics
• DeCoro & Tatarchuk [2007]
  • Vertex Clustering on GPU
Why not QEM + Edge Collapse?

- First thing we tried
- Simply couldn’t get it fast enough
  - Sequence of serial operations
  - Poor memory access patterns
- Requires triangle connectivity
- Requires collection of manifold surfaces
Vertex Clustering

- Older, less sophisticated technique
- Very fast, very simple, very robust
- Quality not as good as edge-collapse-based algorithms
Vertex Clustering

- Enclose model with a uniform grid
- Cluster vertices inside cells
- Remap vertex indices according to cells
  - Store unique index in grid
  - OR use virtual grid: hash map lookup on cell i,j,k
Vertex Clustering

- For all vertices:
  - Classify by containing grid cell
  - Accumulate representative cell position

- For all triangles:
  - Update vertex indices according to cell
  - Discard if degenerate

- Compact mesh
Vertex Clustering Advantages

- Fast on modern architectures
- Doesn’t require edge connectivity
- Good memory coherency as Lindstrom demonstrated
- Two linear passes: vertices then indices
- Robust
- Will take absolutely any mesh you throw at it
LOD for GPU

- Lots of small triangles are bad
  - triangle setup
- Sliver triangles also bad
- Traditional simplification focuses on preserving detail
- Better: match triangle density to pixel density
- Vertex Clustering a good fit for this
Problem Solved?

3.2 Dual Quadric Metric

Given a set of vertices \( P \)

\[ P = \{(v_i, c_i)\}_{i=1}^{n} \]

This allows us to express the covariance matrix for the set as

\[ D = \sum_{i=1}^{n} (v_i - \bar{v})(v_i - \bar{v})^T \]

This formulation neglects the

\[ P \]

3.3 Quantizing the Mesh

Each vertex in the mesh generates a dual quadric that is added

\[ Q \]

To compute the sum of squared distances to a set of planes, we only

\[ D \]

Just as the quadric metric encodes distance from a point to a set of

\[ P \]

Dual quadric \([8, 10]\) measures distance from a plane to a set of points.

\[ Q \]

3.2 Dual Quadric Metric

Given a set of vertices

\[ P \]

This allows us to express the covariance matrix for the set as

\[ D = \sum_{i=1}^{n} (v_i - \bar{v})(v_i - \bar{v})^T \]

This formulation neglects the

\[ P \]

3.3 Quantizing the Mesh

Each vertex in the mesh generates a dual quadric that is added

\[ Q \]

To compute the sum of squared distances to a set of planes, we only

\[ D \]

Just as the quadric metric encodes distance from a point to a set of

\[ P \]
Problem Not Solved

- Position-based meshes only!
- No normal discontinuities
- Not textured
- Not animated
- No vertex-based material info
- Most game meshes feature all of the above
Attributes!
Animation!
So what happens?
So what happens?
Stuff happens
UV Chart Mixing
The Problem

- Attributes have discontinuities
- UV charts particularly bad
- Also normal/material splits (see paper)
- Can’t just ignore!
Edge Collapse
Edge Collapse Discontinuity
Discontinuity Preserved
Edge Collapse

- Edge Collapse deals with attributes natively:
  - Discontinuities are preserved
  - Or removed when interior to the collapsed triangles
- Simplification is a series of discontinuity-preserving collapses
Vertex Clustering with Attributes
Vertex Clustering with Attributes
Vertex Clustering with Attributes
Oversharing
Input Attributes
Just right
A Close-up
Insight: Boundary Edges
Issues

• How do we find the boundary edges?
• How do we use edges to link output vertices?
  • Without memory allocations
  • Efficiently
Finding Boundary Edges
Boundary Edge Chains
Grouping Edges

• Naive way
  • Treat as linked list
  • Insertion is $O(k)$, $k$ boundary edges
  • $O(k^2)$

• Observations
  • Insert $k$ edges, query $m$ edges, $m << k$
Union Find!

- Amortised $O(1)$ insertion and query
- setLinks stores back pointers
Partial Path Compression

- Don’t do full path compression
  - Doesn’t help! In fact hurts
  - Extra memory accesses not paid for by results
- Do compress input vertices
  - Memory we have to access anyway.
  - Does result in minor gains
Building the sets

```plaintext
rv0 = dv0 = ea_n[ev[e0]]
rv1 = dv1 = ea_n[ev[e1]]
level = 0

while (setLinks[rv0]) >= 0)
    rv0 = setLinks[rv0]
    level++

while (setLinks[rv1]) >= 0)
    rv1 = setLinks[rv1]
    level--

if (rv0 != rv1)
    if (level < 0)
        setLinks[rv0] = rv1
        setLinks[dv0] = rv1
    else
        setLinks[rv1] = rv0
        setLinks[dv1] = rv0
```
Using the sets

```plaintext
foreach (iv in 3 Nf)
    i  = ev[iv]
    dv = ea_n[i]
    rv = dv

while (setLinks[rv] >= 0)
    rv = setLinks[rv]

if (setLinks[rv] == -1)
    setLinks[rv] = -2 - dv;

if (dv != rv)
    ea_n[i] = -2 - next
    setLinks[dv] = next
```
Results
Results

• Done!

• But can do more to improve Vertex Clustering quality
Shape Preservation

• A consequence of vertex clustering:
  • Any feature smaller than the cell size in at least one dimension will disappear completely

• Not always desirable!
  • Limbs
  • Poles, fences
Disappearing Trunk
Disappearing Trunk
Shape Preservation
Thin Features
After Collapse
Insight: Normal Clustering
Cluster Strategy

- Quantize normal 8 ways
- Trivial: assemble x/y/z sign bits
- Cell label now `<cell>_<qnorm>`
Shape Preservation
Trunks Preserved
Trunks Preserved
Bone Preservation

• Simplifying animated models leads to problems
  • Base pose is not representative of all animated poses
  • May collapse parts of the mesh together that are animated independently
Webbing
Fixing Unwanted Collapses

- Use same approach as normal clustering
- Append major bone index to the vertex label
  - Prevents any triangle spanning two bones from being removed
- Avoids cross-limb collapses
- Label: `<cell>_<qnorm>_<bone>`
- Fast to look up with sorted weights
Simplification Control

- Information from game can help:
  - Know which parts of the mesh are animated
  - Know which parts are detail and can be heavily simplified
  - Use to affect simplification factor (cell size) and what extensions to use

- See paper
Label Size

- We’ve been merrily extending the vertex label, does that hurt us?
  - `<cell>_<qnorm>_<bone>_<tag>`

- Previously: xyz x 32 bits, hash to output index

- Now: 3 x 24 bits + normal (3 bits) + bone (8 bits) + tag (5 bits)

- No change to cluster index lookup!
Results
Results

- **Vertex Clustering**
- **Attribute Handling**
- **Attribute + Shape**
- **Attribute + Shape + Bones**

![Graph showing time taken vs simplification factor for different configurations.](image-url)
Summary

• Vertex Clustering adapted for production quality meshes
• High speed
  • Memory friendly, faster for lower LODs
  • Job-friendly, mostly Compute-friendly
• Robust!
  • No restrictions on input mesh
Testing!

• There are 160 million\(^1\) player-created models published on [http://www.spore.com/sporepedia](http://www.spore.com/sporepedia)

• Our system has generated 3-4 LODs for all of them with no issues.

\(^1\) 165,568,111 @ 9am
Acknowledgements

• Ocean Quigley
• Maxis
  • Core Engine Team
• Lucy Bradshaw

• Questions?
Parallelism

- Label assignment is embarrassingly parallel
- Compaction of triangle list = stream compaction
- Boundary edges work at the cell level
- Ideally suited for SPU
QuantiseVertices:
   foreach (i in Nv)
       Generate cell label
       Record replacement index ep[i]
       Accumulate p into representative point p_label

RemoveDegenerateTriangles:
   foreach (i in Nf)
       if (p[ep[ev[3i]]] = p[ep[ev[3i + 1]]] = p[ep[ev[3i + 2]]])
           Discard triangle

Compact:
   Share all vertices with identical element references
   Remove all unindexed data
Normal Discontinuities