Interactive Visualization of Complex Scenes

Thomas Funkhouser
COS 526, Fall 2006

Interactive Visualization Goals

Realism
- Realistic enough to convey information
Frame rate:
- >10-60 frames per second
Latency
- <10-500 milliseconds response delay
Computation
- Fast preprocessing
- Fast startup
- Low storage
- etc.

Scene Complexity

Examples:
- Automobile: ~20,000 parts
- Boeing Airplane: ~2,000,000 parts
- Aircraft Carrier: ~20,000,000 parts
- Sculptures: ~200,000,000,000 samples
- Outdoor Environments

Prototype Models

- Cassini spacecraft
  - 349,281 faces
  - 127 objects

Architectural Models

- Soda Hall Model (7.6M polygons)
Coal-Fired Powerplant: 15 million triangles

Submarine Torpedo Room (850K polygons)

82 million triangles; 126,000 objects
Newport News Shipbuilding

Michelangelo sculptures: 100M Samples
Stanford Graphics Lab

Michelangelo sculptures: 127M Samples
Stanford Graphics Lab

Rendering Acceleration Techniques

Visibility Culling
- Backface culling, view-frustum culling, occlusion culling, ...

Detail Elision
- Levels of detail, multiresolution, ...

Images
- Textures, billboards, imposters, ...
Visibility Culling

Quickly eliminate large portions of the scene that will not be visible in the final image
- Not the exact visibility solution, but a quick and conservative test to reject primitives that are not visible
- Trivially reject stuff that is obviously not seen
- Use 2-buffer and clipping for the exact solution

Back-Face Culling

Do not draw polygons facing backwards with respect to camera

A polygon is backfacing if \( V \cdot N < 0 \)

Hierarchical Back-Face Culling

Avoid testing every face separately
- Cluster faces hierarchically
- Precompute range of normals for each cluster
- Check cluster before testing every face hierarchically

View-Frustum Culling

Don’t draw primitives outside the view frustum
- Organize primitives into clumps
- Before rendering the primitives in a clump, test their bounding volume against the view frustum
**View-Frustum Culling**

Hierarchical bounding volumes
- If a clump is entirely outside or entirely inside view frustum, no need to test its children

**Hierarchical View-Frustum Culling**

Hierarchical bounding volumes
- If a clump is entirely outside or entirely inside view frustum, no need to test its children

**Hierarchical View Frustum Culling**

Hierarchical bounding volumes
- If a clump is entirely outside or entirely inside view frustum, no need to test its children

**Uniform Grid Subdivision**

**Octree Subdivision**

Hierarchical View Frustum Culling

What shape should the bounding volumes be?
- Spheres and axis-aligned bounding boxes:
  - Simple to calculate/test
  - May be poor approximation
- Convex hulls:
  - More complex to calculate/test
  - Tighter approximation
BSP Subdivision

Occlusion Culling
Blue parts: occluders
Red parts: occludees

Occlusion Culling
Object-precision
- Cells and portals
- Shadow volumes
Image-precision
- OpenGL occlusion test
- Hierarchical Z-buffer
- Hierarchical occlusion maps

Cells and Portals
Subdivide space into cells.
Cluster polygons into objects.
Build an index of objects incident upon each cell.

Cells and Portals
Visible cells
- Cells used by sightlines from viewpoint
Visible Objects
- All objects visible upon lines of sight in 3D space
Source Cell
Visible Cells
Visible Objects
Observer
Occlusion Culling

Object-precision
- Cells and portals
- Shadow volumes

Image-precision
- OpenGL occlusion test
- Hierarchical Z-buffer
- Hierarchical occlusion maps

OpenGL Occlusion Test

Hardware returns how many z-buffer tests pass

Hierarchical Z-Buffer

Store z-buffer as pyramid and test depth hierarchically

Rendering Acceleration Techniques

Visibility Culling
- Backface culling, view-frustum culling, occlusion culling, ...

Detail Elision
- Levels of detail, multiresolution, ...

Images
- Textures, billboards, imposters, ...

Levels of Detail

courtesy of Division and Viewpoint

Multiresolution
**QSplat**

Key observation: a single bounding sphere hierarchy can be used for
- Hierarchical frustum and backface culling
- Level of detail control
- Splat rendering [Westover 89]

**Creating the Data Structure**

Start with a triangle mesh produced by aligning and integrating scans [Curless 96]

**Creating the Data Structure**

Place a sphere at each node, large enough to touch neighboring spheres

**Creating the Data Structure**

Build up hierarchy

**QSplat Node Structure**

Position and radius encoded relative to parent node
- Hierarchical coding vs. delta coding along a path for vertex positions
QSplat Node Structure

Uncompressed

Hierarchical Coding

Normal quantized to grid on faces of a cube

Each node contains bounding cone of children’s normals
Hierarchical backface culling [Kumar 96]
**QSplat Node Structure**

- **Position and Radius**: 13 bits
- **Tree Structure**: 3 bits
- **Normal**: 14 bits
- **Width of Cone of Normals**: 2 bits
- **Color (Optional)**: 16 bits

**Per-vertex color is quantized 5-6-5 (R-G-B)**

**QSplat Rendering Algorithm**

1. Traverse hierarchy recursively
2. If (node not visible)
3. Skip this branch
4. Else if (leaf node)
5. Draw a splat
6. Else if (size on screen < threshold)
7. Draw a splat
8. Else
9. Traverse children

**Frame Rate Control**

- Feedback-driven frame rate control
  - During motion: adjust recursion threshold based on time to render previous frame
  - On mouse up: redraw with smaller thresholds
  - Consequence: frame rate may vary
- Alternative:
  - Predictive control of detail [Funkhouser 93]

**Results**

- Interactive (9 frames/sec)
- High quality (6 sec)

**Demo – St. Matthew**

- 3D scan of 2.7 meter statue at 0.25 mm
- 102,868,637 points
- File size: 644 MB
- Preprocessing time: 1 hour
**Tradeoffs of Splatting**

For rendering large 3D models, what are the tradeoffs of:

<table>
<thead>
<tr>
<th>Polygons</th>
<th>Q5plat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good for large, flat or subtly curved regions</td>
<td>Good for models with detail everywhere</td>
</tr>
<tr>
<td>Highly-efficient rasterization with 3D graphics hardware</td>
<td>Higher per-pixel cost, but less slowdown in absence of 3D hardware</td>
</tr>
<tr>
<td>Decimation or creating LOD data structures is often expensive</td>
<td>Fast preprocessing</td>
</tr>
</tbody>
</table>

**Rendering Acceleration Techniques**

- Visibility Culling
  - Backface culling, view-frustum culling, occlusion culling, ...
- Detail Elision
  - Levels of detail, multiresolution, ...
- Images
  - Textures, billboards, imposters, ...

**Imposters**

**Algorithm**
- Select subset of model
- Create image of the subset
- Cull subset and replace with image

**Why?**
- Image displayed in (approx.) constant time
- Image reused for several frames

**Simple Example**
**Issues**

Imposter placement
- What geometry should be replaced by images
- How should images be integrated into scene

Imposter representation
- What viewpoint(s) should be captured in image?
- How render from arbitrary viewpoints?

**Imposter Placement**

**Cells and Portals**

**Portal Images**

**Creating Portal Images**

Ideal portal image would be one sampled from the current eye position

Display one of a large number of pre-computed images (~120)
Creating Portal Images

or...

Warp one of a much smaller number of reference images

Summary

Visibility Culling
- Backface culling, view-frustum culling, occlusion culling, ...

Detail Elision
- Levels of detail, multiresolution, ...

Images
- Textures, billboards, imposters, ...

Recurring Themes:
- Trivial reject checks
- Hierarchical processing