Beyond Meshes
Fall 2017
Extreme LOD

• What can a mesh turn into at its most distant LOD?
Billboards

- A billboard is extreme Level of Detail (LOD), reducing all the geometry to one or more textured polygons
  - Also considered a form of image-based rendering
- Questions in designing billboards:
  - How are they generated?
  - How are they oriented?
- Also called sprites, but a sprite normally stays aligned parallel to the image plane
How to Generate Billboards?

• By hand – a skilled artist does the work
  • Paints color and alpha
  • May generate a sequence of textures for animating
• Automatically:
  • Render a complex model and capture the image
  • Alpha detected by looking for background pixels in the image
  • Blend out alpha at the boundary for good anti-aliasing
How to Configure Billboards?

- The billboard polygons can be laid out in different ways
  - Single rectangle
  - Two rectangles at right angles
  - Several rectangles about a common axis
  - Several rectangles stacked
- Issues are:
  - What sorts of billboards are good for what sorts of objects?
  - How is the billboard oriented with respect to the viewer?
Single Polygon Billboards

- The billboard consists of a single textured polygon
- It must be pointed at the viewer, or it would disappear when viewed from the side
- Point Sprites:
  - Billboard rotated about a central point that faces the camera
- Axis Billboards:
  - Billboard aligned along an axis (arbitrary or axis-aligned)
Activity: Aligning a Billboard

- Billboard has a known forward vector $F$ that points out from the face
- Billboard has an “up” or axis vector $A$
- Camera has a view direction $V$

- Draw a billboard with its current orientation and show how we can realign $F$ to face $V$
Alignment About Axis

- **A** is billboard axis, **V** is viewer direction. From current forward **F** move to desired forward **D**

- Calculate **D**

- Compute angle $\gamma$ between **F** and **D**

- Significant shortcut if **A** is the z axis, and **F** points along the x axis

\[
D = A \times (V \times A)
\]

\[
\gamma = \cos^{-1}\left(\frac{F \cdot D}{\|F\|\|D\|}\right)
\]

\[
\gamma = \tan^{-1}\left(\frac{V_y}{V_x}\right)
\]
Multi-Polygon Billboards

- Use two polygons at right angles:
  - No alignment with viewer
  - What is this good for?
- Use more polygons for better appearance
- Rendering options: Blended or just depth buffered
View Dependent Billboards

- What if the object is not rotationally symmetric?
  - Appearance should change from different viewing angles
- This can be done with billboards:
  - Compute multiple textures corresponding to different views
  - Keep polygon fixed but vary texture according to viewer direction
  - Interpolate with texture blending between the two nearest views
  - Use 3D textures and hardware texture filtering to achieve good results
- Polygons are typically fixed, restricting the viewing angles
  - Use more polygons that each have a set of views associated with it
View Dependent Billboards

(Nvidia)
Impostor Example

- Another method uses slices from the original volume and blends them
Additional Optimizations

• How can we optimize geometry in scenes besides using LOD?
Pipeline Efficiency

- The rendering pipeline is (as the name suggests) a pipeline
  - Slowest pipeline operation determines throughput (frame rate)
  - For graphics, that could be memory bandwidth, transformations, clipping, rasterization, lighting, buffer fill etc
- Profiling tools tell you which part of your pipeline is slow
- One speedup is reducing the complexity of the geometry
  - Impacts every part of the pipeline up to the fragment stage
  - Assumption: You will touch roughly the same pixels, even with simpler geometry
Reducing Geometry

- Assume we are living in a polygon mesh world
- Several strategies exist, with varying degrees of difficulty, reductions in complexity, and quality trade-offs:
  - Reduce the amount of data sent per triangle, but keep the number of triangles the same
  - Reduce the number of triangles by ignoring things that you know the viewer can’t see – visibility culling
  - Reduce the number of triangles in view by reducing the quality (maybe) of the models – level of detail (LOD)
Compressing Meshes

- Base case: Three vertices per triangle with full vertex data (color, texture, normal etc)
- Much of this data is redundant:
  - Triangles share vertices
  - Vertices share colors and normals
  - Vertex data may be highly correlated
- Compression strategies seek to avoid sending redundant data
- Impacts memory bandwidth, but not too much else
  - A concern for transmitting models over a network
Compression Overview

- Use triangle strips to avoid sending vertex data more than once
- Use vertex arrays
  - Tell the API what vertices will be used
  - Specify triangles by indexing into the array
  - Reduces cost per vertex
  - Allows hardware to cache vertices
- Non-shared attributes, such as normal vectors, limit the effectiveness of some of these techniques
Mesh Compression

• Pipelined hardware typically accepts data in a stream, and has small buffers
  • Can’t do decompression that requires holding the entire mesh, or any large data structure
• Do decompression in software
• Typical strategies
  • Treat connectivity (which vertices go with which triangles) separately from vertex attributes (location, normal etc)
  • Build long strips or other implicit connectivity structures
  • Apply standard compression techniques to vertex attributes
Billboarding How-tos

- NeHe Productions &lt;http://nehe.gamedev.net/article/billboarding_how_to/18011/&gt;

- Lighthouse 3D &lt;http://www.lighthouse3d.com/opengl/billboarding/&gt;