Advanced Computer Graphics
CS 563: Non-Photorealistic Rendering

Frederik Clinckemaillie

Computer Science Dept.
Worcester Polytechnic Institute (WPI)
Uses of Non-Photorealistic Rendering

- Different styles of Non-Photorealistic Rendering (NPR) are used for different reasons
  - Displaying information
  - Creating emotion
  - Imitate art style
Toon Shading

- Cartoon-like rendering effects
- Silhouette Edges

Simple Shading:
- Solid shading
- Two-tone shading
- Specular/Diffuse three-tone shading
Cel Shading

- Solid Shading is trivial
- Two-Tone Shading
  - Calculate the diffuse shading dot product \((n \cdot l)\) for each vertex
    - If \(n \cdot l < 0\), surface is facing away
  - Use values on a 1-D texture map
  - Implemented on CPU or as Vertex Shader
  - Tone Mapping
Additional Work in Cel Shading

- Card and Mitchell [2002]
  - Determined how to perform algorithm on GPU effectively
- Barla et al. [2001]
  - Added view-dependent effects by using 2-D maps
- Rusinkiewicz et al. [2006]
  - Proposed Alternate shading model with high-contrast by adjusting effective light position
Cel Shading (cont.)

- Specular/Diffuse three-tone shading
  - Calculate the specular shading dot product ($L \cdot R$) for each vertex
  - Map to 1-D texture map
Silhouette Edge Rendering

- Adds silhouettes to emphasize edges
- Different categories of algorithms:
  - Surface Angle
  - Procedural Geometry
  - Image Processing
  - Vector Edge Detection
  - Hybrid
Types of Edges:

- **Boundary edge**
  - One not shared by two polygons

- **Crease edge:**
  - Angle is greater than 60 degrees

- **Material edge**
  - One where polygons are different materials

- **Silhouette edge**
  - Two neighboring triangles face different directions as compared to eye vector
Surface Angle Silhouetting

- Using the dot product of the viewpoint and the surface normal
- Results are mapped to circle map
- Results close to 0 are near Silhouette edge
- Does not work for with Surfaces with hard edges
- No vertex interpolation
Procedural Geometry Silhouetting

- Render front faces normally, render the edges of the backfaces behind front faces
- Z-bias method
  - Move backfaces closer so that edges become in front
  - Edges thickness is not constant
Procedural Geometry Silhouetting

- Backface triangles are fattened.
- Slope of triangle and distance from viewer determine fattening.
- Corners are cut off to avoid elongated corners.
Results

Figure 11.8. Silhouettes rendered with back-facing edge drawing with thick lines, z-bias, and fattened triangle algorithms. The backface edge technique gives poor joins between lines and nonuniform lines due to biasing problems on small features. The z-bias technique gives nonuniform edge width because of the dependence on the angles of the frontfaces. (Images courtesy of Raskar and Cohen [1046].)
Halo Method

- Move the backfaces outwards by shifting their vertices along the shared vertex normals
- Creates a halo behind front faces
Limitations

- Fattening cannot work on curved surfaces
- Shell techniques work on curved surfaces only if the surface representation can be displaced along surface normals
- Z-bias works with all curved surfaces
- Little control over edge appearance
- Semitransparent surfaces are difficult to render
- Edges look poor without antialiasing
Silhouetting by Image Processing

- Entirely based on data stored in buffers
- No modifications on the geometry of the scene
- Look for silhouette edges by searching for z-buffer discontinuities
Silhouette Edge Rendering

- **Process:**
  - Use Vertex shaders to render the world space normals and z-depth to a texture
  - Normals are colors, z-depths are alpha channels
  - Implement a Sobel edge detection filter

- **Benefits**
  - Handles all primitives
  - Meshes do not need to be connected or consistent

- **Flaws**
  - Edges might not be found if discontinuities are small
Silhouette Edge Detection

- Other methods have weaknesses:
  - Often need two passes to render the silhouette
  - No control on edge rendering style

- Detect silhouette edges and render them directly
  - Obtain more fine control on how lines are rendered
  - Edges become independent of the model

- An edge is defined by:
  \[(n_0 \cdot v > 0) \neq (n_1 \cdot v > 0)\]
Finding Silhouette Edges

- Standard: Loop through list of edges
- Improving efficiency:
  - Removing triangles inside planar polygons
  - Storing dot products for faces
  - In static scenes, preserve silhouettes until view moves a certain distance
Hybrid Silhouetting

- Northrup and Markosian
- Find a list of silhouette edges.
- Render all triangles and silhouette edges, assigning each a different ID number(color)
- ID buffer is read back and visible silhouette edges are determined from it
- Visible segments are then checked for overlaps and linked together
- Stylized strokes are added on path
Figure 11.12. An image produced using Northrup and Markosian’s hybrid technique, whereby silhouette edges are found, built into chains, and rendered as strokes. (Image courtesy of Lee Markosian.)
References