VGP352 – Week 7

- Agenda:
  - Quiz #3!
  - Assignment #3 due
  - Non-photorealistic rendering
    - Cel shading (cartoon rendering)
    - Silhouette edge rendering
    - Technical illustration
  - Assignment #4
  - Readings:
    - Present readings 1 and 2
    - Assign reading 3

29-January-2008

Non-photorealistic Rendering (NPR)

- From wikipedia:

  Non-photorealistic rendering (NPR) is an area of computer graphics that focuses on enabling a wide variety of expressive styles for digital art.

Non-photorealistic Rendering (NPR)

- From wikipedia:

  Non-photorealistic rendering (NPR) is an area of computer graphics that focuses on enabling a wide variety of expressive styles for digital art.

- In other words, NPR attempts to exaggerate or use alternate representations of imagery to convey or highlight a particular mood or message
  - Cel shading (a.k.a. “toon” rendering)
  - Painterly rendering
  - Technical illustrations
  - etc.

Cel Shading

- Several common cartoon image styles:
  - Character regions filled with solid, single-tone colors
  - Regions filled with two tones: light and dark
  - Regions filled with three tones: light, dark, and highlight

Cel Shading

- Several common cartoon image styles:
  - Character regions filled with solid, single-tone colors
  - Regions filled with two tones: light and dark
  - Regions filled with three tones: light, dark, and highlight
  - Each is easy to produce on a computer

Cel Shading

- Single tone coloring

29-January-2008

Cel Shading

- Single tone coloring
  - Solid coloring (flat shading) *without* lighting

Cel Shading

- Single tone coloring
  - Solid coloring (flat shading) *without* lighting
- Two-tone coloring

Cel Shading

- Single tone coloring
  - Solid coloring (flat shading) *without* lighting
- Two-tone coloring
  - Driven by surface lighting
  - If lighting is above some threshold, use the lighter color
  - Otherwise use the darker color

Cel Shading

- Single tone coloring
  - Solid coloring (flat shading) *without* lighting
- Two-tone coloring
  - Driven by surface lighting
  - If lighting is above some threshold, use the lighter color
  - Otherwise use the darker color
  - Calculate $N \bullet L$ *per vertex* and interpolate across surface, check value per fragment
    - Classically done using texture lookups, but is faster using conditional assignments on shader hardware

Cel Boundary Inking

- Anyone who has seen a cartoon or a comic book knows that certain boundaries are “inked”

Cel Boundary Inking

- Anyone who has seen a cartoon or a comic book knows that certain boundaries are “inked”
- Four main types of edges need inking:

Cel Boundary Inking

- Anyone who has seen a cartoon or a comic book knows that certain boundaries are “inked”
- Four main types of edges need inking:
  - Border edges – edges not shared by two polygons

Cel Boundary Inking

- Anyone who has seen a cartoon or a comic book knows that certain boundaries are “inked”
- Four main types of edges need inking:
  - Border edges – edges not shared by two polygons
  - Crease edges – edges where the angle between the two surfaces is too sharp
    - This angle is called the *dihedral angle*

Cel Boundary Inking

- Anyone who has seen a cartoon or a comic book knows that certain boundaries are “inked”
- Four main types of edges need inking:
  - Border edges – edges not shared by two polygons
  - Crease edges – edges where the angle between the two surfaces is too sharp
    - This angle is called the *dihedral angle*
  - Material edge – boundary between two different colors or materials

Cel Boundary Inking

- Anyone who has seen a cartoon or a comic book knows that certain boundaries are “inked”
- Four main types of edges need inking:
  - Border edges – edges not shared by two polygons
  - Crease edges – edges where the angle between the two surfaces is too sharp
    - This angle is called the *dihedral angle*
  - Material edge – boundary between two different colors or materials
  - Silhouette edges – edges where one border polygon faces towards the viewer and the other faces away

Cel Boundary Inking

- Most boundary types are calculated during authoring or as a preprocessing step
  - Border edges – edges are added by the artist, by the authoring tool, or are detected in a preprocessing step
  - Crease edges – dihedral angle is calculated during preprocessing. If $N_{surface1} \bullet N_{surface2} < \cos(60^\circ)$, the edge is a crease
  - Material edge – handled the same as border edges

Silhouette Edge Rendering

- Silhouette edges are view-dependent and must be calculated at run-time
  - Conceptually similar to drawing fins in shells-and-fins fur rendering
- Several broad classes of implementations:
  - Surface angle
  - Added geometry
  - Image processing
  - Explicit edge detection

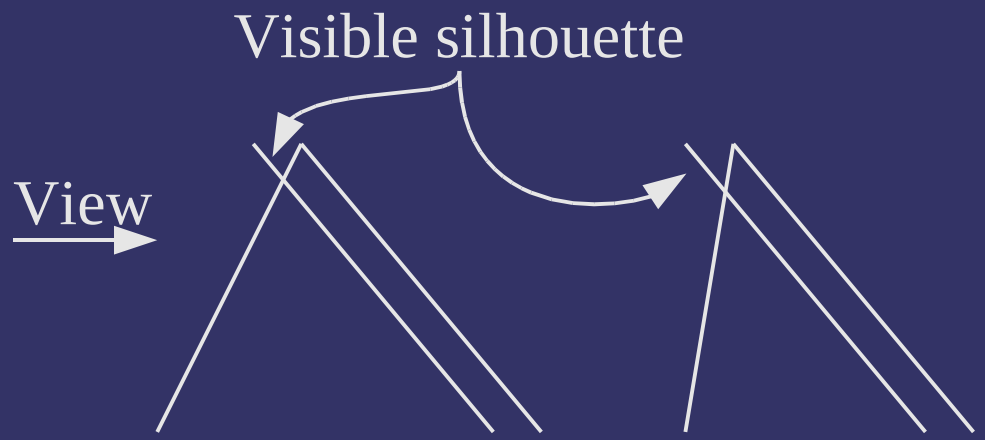
Silhouette Edge Rendering

- Surface angle test is similar to two-tone cel shading
  - Examine angle between V and N
  - If angle is near 90°, use silhouette color
- Pros / cons:
  - *Really* easy to implement
  - Doesn't work on all models
    - Generally fails on models with large flat surfaces
    - Only worked on about 25% of the models in the game *Cel Damage*[1]

[1] *Real-Time Rendering*, p. 295

Silhouette Edge Rendering

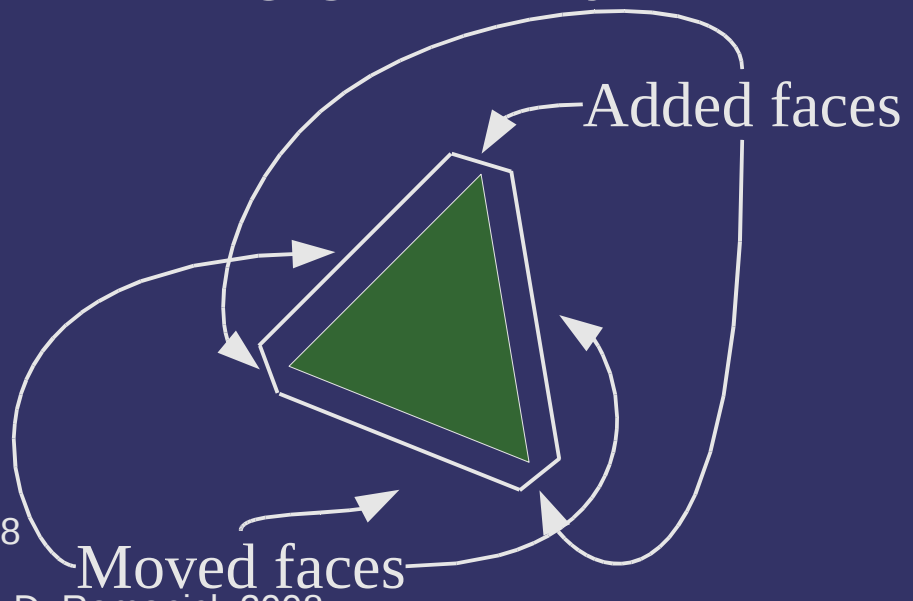
- Back-face biasing:
  - Render back-facing geometry by moving it towards the camera by some small delta



  - Amount to bias back-face depends on both slope of back-face and slope of front-face

Silhouette Edge Rendering

- Edge expansion:
  - Move each face out by some distance along the plane's normal
    - *Not* the vertex normal!
    - Adjust the distance according to the desired silhouette thickness
  - Create new geometry to fill in the gaps
  - Render back-facing geometry



Silhouette Edge Rendering

- Shell expansion:
  - Similar to edge expansion
  - Render shell as object geometry expanded along vertex normals
    - Normals must be identical for vertices shared by two polygons
    - Otherwise degenerate edge polygons must be added
    - Render only back-faces of shell

Silhouette Edge Rendering

- Image processing:
  - Render surface normal and depth a texture
    - Store normal in RGB and most significant portion of depth in alpha
  - Process texture with separable edge detection filter
    - Card and Mitchell recommend using the Sobel edge detection filter
    - Store each pass in a texture
    - Composite both textures together over scene

Silhouette Edge Rendering

- Explicit edge detection:
  - Draw each edge of the object as a line
  - At each vertex, store the normals of the two adjoining polygons
  - If one normal points towards the viewer and the other away, draw the line as a silhouette
  - If the two normals point significantly away from each other, draw the line as a crease

Break

29-January-2008

Gooch-style Technical Illustration

- Many similar ideas to cel shading
  - Use alternate shading
  - Highlight creases
  - Highlight silhouettes

Gooch-style Technical Illustration

- Shade objects from warm to cool instead of light to dark
  - Still conveys information about the curvature of the object
  - Maintains visibility of details in areas that would be dark or difficult to light

Gooch-style Technical Illustration

- Shade objects from warm to cool instead of light to dark
  - Still conveys information about the curvature of the object
  - Maintains visibility of details in areas that would be dark or difficult to light
- Shade in similar manner to cel shading
  - Calculate $N \bullet L$ per vertex
  - Use interpolated value per fragment to look up in a 1D blue-green to yellow-orange gradient texture

Gooch-style Technical Illustration

- Draw crease edges in white
  - This helps provide information about the model's orientation

Gooch-style Technical Illustration

- Draw crease edges in white
  - This helps provide information about the model's orientation
- Draw silhouette edges in black
  - If an edge is *both* a crease and a silhouette, it should be drawn as a silhouette

Gooch-style Technical Illustration

- Draw crease edges in white
  - This helps provide information about the model's orientation
- Draw silhouette edges in black
  - If an edge is *both* a crease and a silhouette, it should be drawn as a silhouette
- Silhouette and crease edges are handled differently, so the image processing method of inking probably can't be used
  - Using the explicit edge detection method allows silhouettes and creases to be drawn in a single pass

References

Gooch, B., Sloan, P. J., Gooch, A., Shirley, P., and Riesenfeld, R. 1999. Interactive technical illustration. In *Proceedings of the 1999 Symposium on interactive 3D Graphics* (Atlanta, Georgia, United States, April 26 - 29, 1999). I3D '99. ACM, New York, NY, 31-38. http://www.cs.utah.edu/~bgooch/ITI/

Next week...

- Procedural textures
  - Noise
  - Simple noise based textures
  - Wang tiles

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of IBM or the Art Institute of Portland.

OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.

Khronos and OpenGL ES are trademarks of the Khronos Group.

Other company, product, and service names may be trademarks or service marks of others.

29-January-2008