# *VGP352 – Week 7*

▷ Agenda:
- Finish material from last week
- Nonphotorealistic Rendering
  - Cel shading
  - Gooch technical illustration

# *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.

17-November-2010

# *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

# *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 $\mathbf{n} \cdot \mathbf{l}$ *per vertex* and interpolate across surface, check value per fragment

   – Classically done using texture look-ups, but is faster using conditional assignments on shader hardware

# *Cel Shading*

▷ Shadow-and-highlight method:

- Calculate $\mathbf{n} \cdot \mathbf{l}$ per vertex
- Look-up in texture that is half shadow, almost half non-shadow, and a small amount "highlight"
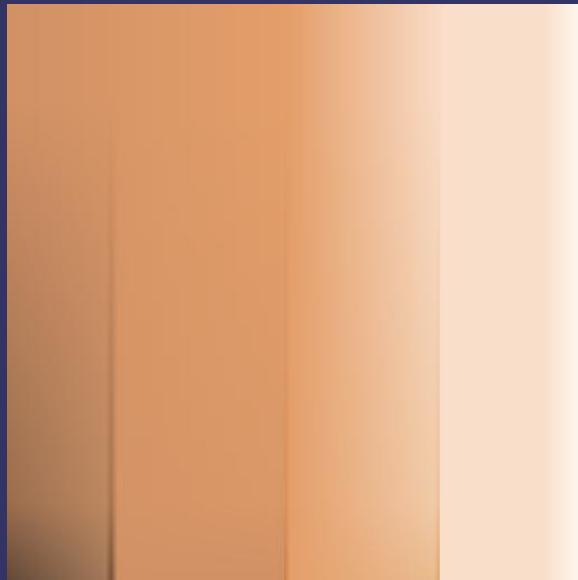    - Highlight is usually 1 texel



Image from [Lake 2000]

17-November-2010

# Cel Shading

⇨ X-toon method replaces 1D $\mathbf{n} \cdot \mathbf{l}$ look-up with a 2D $\{\mathbf{n} \cdot \mathbf{l}, (\mathbf{n} \cdot \mathbf{v})^s\}$ look-up

  – Provides a Fresnel-like "rim light" effect

Images from [Barla 2006]

# *Cel Shading*

▷ Team Fortress 2 takes this several steps further...

$$\mathbf{k}_d\Big[a(\mathbf{n})+\sum\nolimits_{\text{all lights}} \mathbf{c}_i\, w\Big(\big(\alpha(\mathbf{n}\cdot\mathbf{l}_i)+\beta\big)^y\Big)\Big]+$$

$$\sum\nolimits_{\text{all lights}}\Big[\mathbf{k}_s\mathbf{c}_i\max\Big(f_s(\mathbf{v}\cdot\mathbf{r})^{k_{\text{spec}}},f_r\mathbf{k}_r(\mathbf{v}\cdot\mathbf{r})^{k_{\text{rim}}}\Big)\Big]+(\mathbf{n}\cdot\mathbf{u})f_r\mathbf{k}_r\,a(\mathbf{v})$$

17-November-2010

# *Cel Shading*

▷ Team Fortress 2 takes this several steps further...

$$\mathbf{k}_d\Big[a(\mathbf{n})+\sum\nolimits_{\text{all lights}}\mathbf{c}_i\,w\big(\big(\alpha(\mathbf{n}\cdot\mathbf{l}_i)+\beta\big)^y\big)\Big]+$$

$$\sum\nolimits_{\text{all lights}}\Big[\mathbf{k}_s\mathbf{c}_i\max\big(f_s(\mathbf{v}\cdot\mathbf{r})^{k_{\text{spec}}},f_r\mathbf{k}_r(\mathbf{v}\cdot\mathbf{r})^{k_{\text{rim}}}\big)\Big]+(\mathbf{n}\cdot\mathbf{u})f_r\mathbf{k}_r a(\mathbf{v})$$

Light color
Surface specular color
Surface diffuse color

# *Cel Shading*

▷ Team Fortress 2 takes this several steps further...

$$\mathbf{k}_d \left[ \boxed{a(\mathbf{n})} + \sum\nolimits_{\text{all lights}} \mathbf{c}_i \, w \left( \left( \alpha(\mathbf{n} \cdot \mathbf{l}_i) + \beta \right)^y \right) \right] +$$

$$\sum\nolimits_{\text{all lights}} \left[ \mathbf{k}_s \mathbf{c}_i \max \left( f_s (\mathbf{v} \cdot \mathbf{r})^{k_{\text{spec}}}, f_r \mathbf{k}_r (\mathbf{v} \cdot \mathbf{r})^{k_{\text{rim}}} \right) \right] + (\mathbf{n} \cdot \mathbf{u}) f_r \mathbf{k}_r \boxed{a(\mathbf{v})}$$

Directional ambient term

17-November-2010

# *Cel Shading*

⇨ Team Fortress 2 takes this several steps further...

$$\mathbf{k}_d \left[ a(\mathbf{n}) + \sum\nolimits_{\text{all lights}} \mathbf{c}_i \boxed{\mathbf{w}\left(\left(\alpha(\mathbf{n} \cdot \mathbf{l}_i) + \beta\right)^y\right)} \right] +$$

$$\sum\nolimits_{\text{all lights}} \left[ \mathbf{k}_s \mathbf{c}_i \max\left(f_s(\mathbf{v} \cdot \mathbf{r})^{k_{\text{spec}}}, f_r \mathbf{k}_r (\mathbf{v} \cdot \mathbf{r})^{k_{\text{rim}}}\right) \right] + (\mathbf{n} \cdot \mathbf{u}) f_r \mathbf{k}_r a(\mathbf{v})$$

"Modified" Lambertian term

# *Cel Shading*

▷ Team Fortress 2 takes this several steps further...

$$\mathbf{k}_d \left[ \mathrm{a}(\mathbf{n}) + \sum\nolimits_{\text{all lights}} \mathbf{c}_i \, \mathrm{w}\left( \left( \alpha(\mathbf{n} \cdot \mathbf{l}_i) + \beta \right)^y \right) \right] +$$

$$\sum\nolimits_{\text{all lights}} \left[ \mathbf{k}_s \mathbf{c}_i \max\left( f_s(\mathbf{v} \cdot \mathbf{r})^{k_{\text{spec}}}, f_r \mathbf{k}_r(\mathbf{v} \cdot \mathbf{r})^{k_{\text{rim}}} \right) \right] + (\mathbf{n} \cdot \mathbf{u}) f_r \mathbf{k}_r \mathrm{a}(\mathbf{v})$$

Two specular terms!

# *Cel Shading*

⬦ Team Fortress 2 takes this several steps further...

$$\mathbf{k}_d \left[ \mathrm{a}(\mathbf{n}) + \sum\nolimits_{\text{all lights}} \mathbf{c}_i \, \mathrm{w} \left( \left( \alpha(\mathbf{n} \cdot \mathbf{l}_i) + \beta \right)^y \right) \right] +$$

$$\sum\nolimits_{\text{all lights}} \left[ \mathbf{k}_s \mathbf{c}_i \max \left( \boxed{f_s(\mathbf{v} \cdot \mathbf{r})^{k_{\text{spec}}}}, f_r \mathbf{k}_r(\mathbf{v} \cdot \mathbf{r})^{k_{\text{rim}}} \right) \right] + (\mathbf{n} \cdot \mathbf{u}) f_r \mathbf{k}_r \, \mathrm{a}(\mathbf{v})$$

Fairly standard specular term
  - $f_s$ – artist tuned Fresnel term
  - $k_{\text{spec}}$ – specular from texture or constant

# *Cel Shading*

⇨ Team Fortress 2 takes this several steps further...

$$\mathbf{k}_d \left[ a(\mathbf{n}) + \sum_{\text{all lights}} \mathbf{c}_i w\left( \left( \alpha(\mathbf{n}\cdot\mathbf{l}_i) + \beta \right)^y \right) \right] +$$

$$\sum_{\text{all lights}} \left[ \mathbf{k}_s \mathbf{c}_i \max\left( f_s(\mathbf{v}\cdot\mathbf{r})^{k_{\text{spec}}}, \boxed{f_r \mathbf{k}_r(\mathbf{v}\cdot\mathbf{r})^{k_{\text{rim}}}} \right) \right] + (\mathbf{n}\cdot\mathbf{u}) f_r \mathbf{k}_r a(\mathbf{v})$$

Rim specular term

– $f_r$ – rim Fresnel $(1-(\mathbf{n}\cdot\mathbf{v}))^4$

– $k_{\text{rim}}$ – constant rim exponent

# *Cel Shading*

⇨ Team Fortress 2 takes this several steps further...

$$\mathbf{k}_d \left[ a(\mathbf{n}) + \sum\nolimits_{\text{all lights}} \mathbf{c}_i \, w \left( \left( \alpha(\mathbf{n} \cdot \mathbf{l}_i) + \beta \right)^y \right) \right] +$$

$$\sum\nolimits_{\text{all lights}} \left[ \mathbf{k}_s \mathbf{c}_i \max \left( f_s (\mathbf{v} \cdot \mathbf{r})^{k_{\text{spec}}}, f_r \mathbf{k}_r (\mathbf{v} \cdot \mathbf{r})^{k_{\text{rim}}} \right) \right] + (\mathbf{n} \cdot \mathbf{u}) f_r \mathbf{k}_r \, a(\mathbf{v})$$

Extra term to make rim highlights come from above

— $\mathbf{u}$ – up vector

# References

Lake, A., Marshall, C., Harris, M., and Blackstein, M. 2000. Stylized rendering techniques for scalable real-time 3D animation. In *Proceedings of the 1st international Symposium on Non-Photorealistic Animation and Rendering* (Annecy, France, June 05 - 07, 2000). NPAR '00. ACM, New York, NY, 13-20. http://www.cs.utah.edu/npr/papers/Lake_NPAR00.pdf

Barla, P., Thollot, J., and Markosian, L. 2006. X-toon: an extended toon shader. In *Proceedings of the 4th international Symposium on Non-Photorealistic Animation and Rendering* (Annecy, France, June 05 - 07, 2006). NPAR '06. ACM, New York, NY, 127-132. http://artis.imag.fr/Publications/2006/BTM06a/x-toon.pdf

Mitchell, J. L., Francke, M., and Eng, D. 2007. Illustrative rendering in Team Fortress 2. In *ACM SIGGRAPH 2007 Courses* (San Diego, California, August 05 - 09, 2007). SIGGRAPH '07. ACM, New York, NY, 19-32.  http://www.valvesoftware.com/publications.html

# *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

17-November-2010

# *Cel Boundary Inking*

▷ Most boundary types are calculated during authoring or as a pre-processing step

- Border edges – edges are added by the artist, by the authoring tool, or are detected in a pre-processing step

- Crease edges – dihedral angle is calculated during pre-processing.  If $\mathbf{n}_{surface1} \cdot \mathbf{n}_{surface2}$ < cos(60°), 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 $\mathbf{v}$ and $\mathbf{n}$
- 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
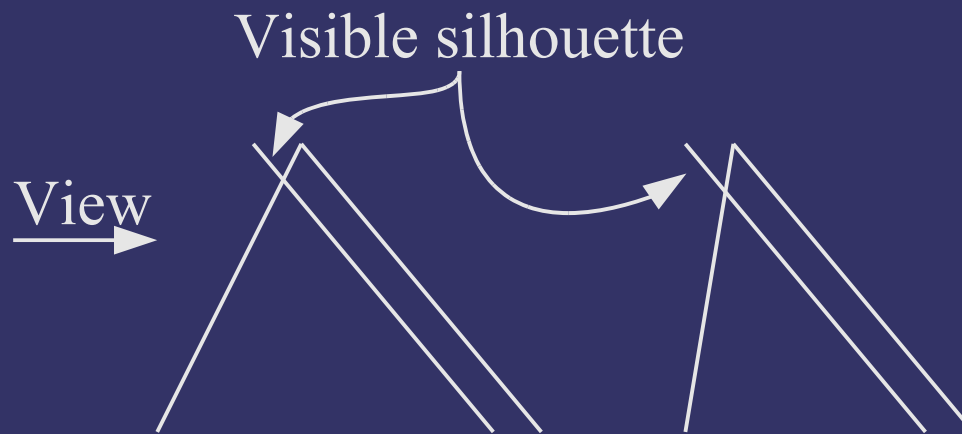  - Only worked on about 25% of the models in the game *Cel Damage*[1]

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

17-November-2010

# *Silhouette Edge Rendering*

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



Visible silhouette

View

- 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

Added faces

Moved faces
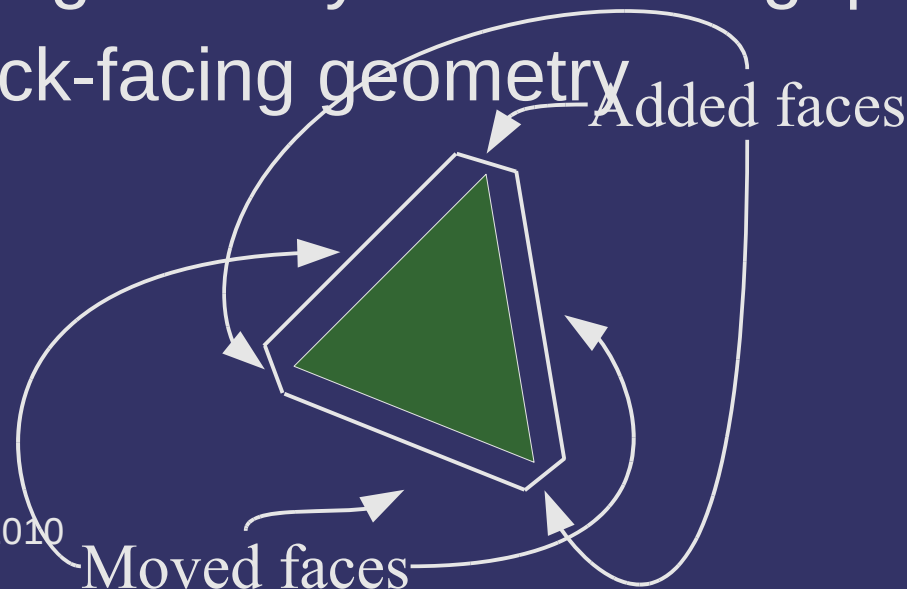
# *Silhouette Edge Rendering*

▷ 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

17-November-2010

# *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

# 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 $\mathbf{n} \cdot \mathbf{l}$ 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

17-November-2010

# *References*

Bruce Gooch and Peter-Pike J. Sloan and Amy Gooch and Peter S. Shirley and Rich Riesenfeld, "Interactive Technical Illustration." In 1999 ACM Symposium on Interactive 3D Graphics , pages 31--38. 1999. http://www.thegooch.org/Publications.php

17-November-2010

# *Next week...*

▷ Texture rectangles

▷ Post-processing

   – Full-screen post-processing overview

   – Filter kernels

      – Separable filters

   – Special effects

      – Water ripple

      – Depth of field

# *Legal Statement*

This work represents the view of the authors and does not necessarily represent the view of Intel 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.