CSE 167:
Introduction to Computer Graphics
Lecture #13: Shader Effects

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2012

Announcements

- CAPE: on-line, email notification at beginning of week 9
  - Period: Monday 11/21 to Monday 12/5
  - http://www.cape.ucsd.edu

- Homework introduction for final project: Monday, Nov 19
  - Project description will go on-line this Friday
  - Final project will require you to work in teams of 2 or 3 people

- Homework assignment #6 due Friday, Nov 16
 - Introduction today after class
  - C^1 vs. G^1 continuity

Parametric Continuity

- C^0 continuity:
  - Curve segments are connected
- C^1 continuity:
  - C^0 & 1st-order derivatives agree
  - Curves have same tangents
  - Relevant for smooth shading
- C^2 continuity:
  - C^1 & 2nd-order derivatives agree
  - Curves have same tangents and curvature
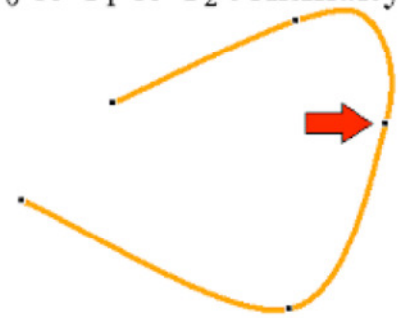 - Relevant for high quality reflections

C_0 continuity

C_0 & C_1 continuity
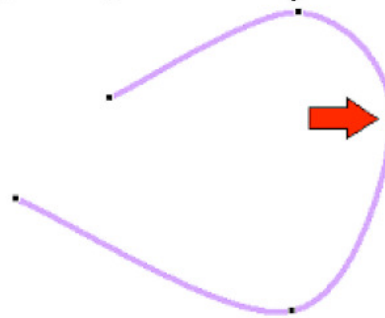
C_0 & C_1 & C_2 continuity

Geometric Continuity

▶ G^0:
  ▶ Curve segments are connected
  ▶ Same as C^0

▶ G^1:
  ▶ G^0 & 1st-order derivatives are proportional at joints
  ▶ Proportional = same direction but may have different magnitudes
  ▶ Weaker than C^1

▶ G^2:
  ▶ G^1 & 2nd-order derivative proportional at joints

▶

Lecture Overview

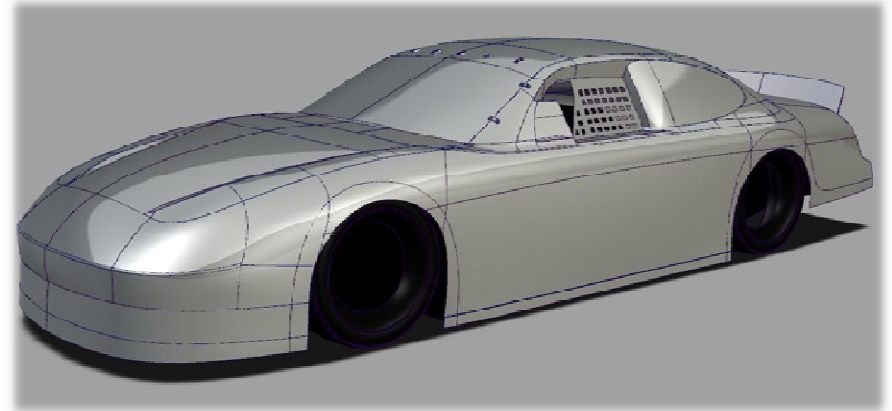▸ Advanced surface modeling

Advanced Shader Effects

▸ Environment mapping

▸ Toon shading

Problems with Bezier and NURBS Patches

- **NURBS surfaces are versatile**
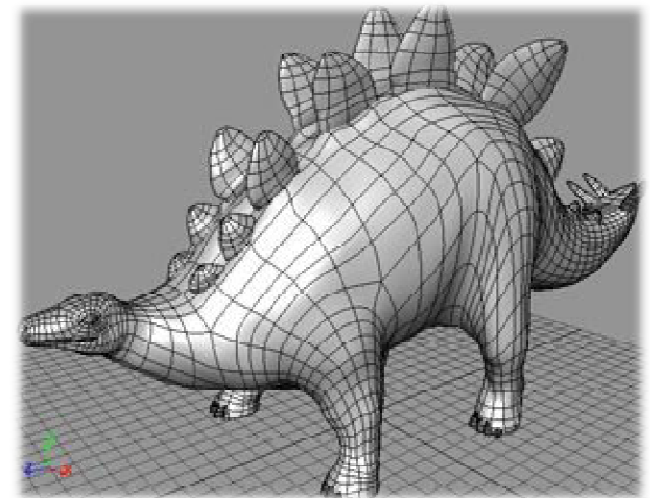 - Conic sections
  - Can blend, merge, trim…

- **But:**
 - Any surface will be made of quadrilateral patches (quadrilateral topology)

- **This makes it hard to**
  - Join or abut curved pieces
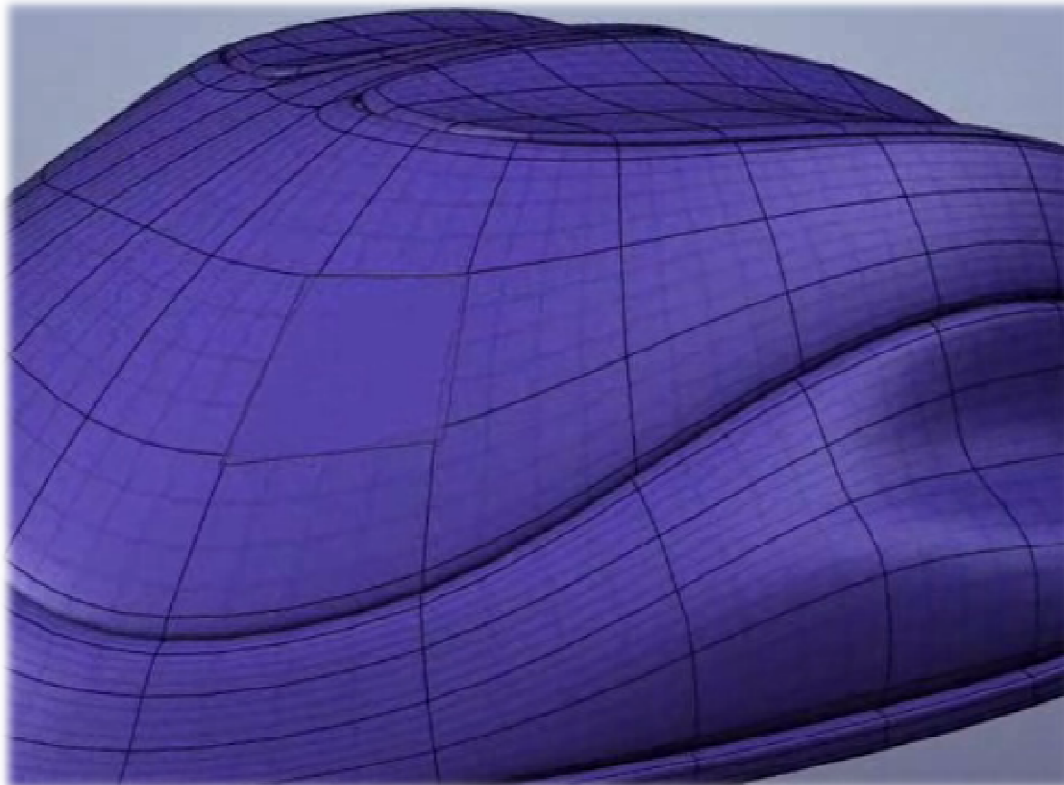 - Build surfaces with complex topology or structure

Subdivision Surfaces

- Works by recursively subdividing mesh faces
  - Per-vertex annotation for weights, corners, creases
- Used in particular for character animation
  - One surface rather than collection of patches
  - Can deform geometry without creating cracks
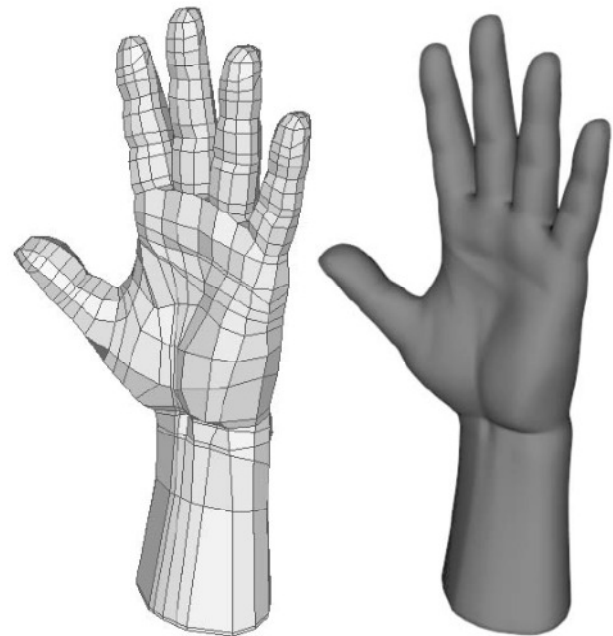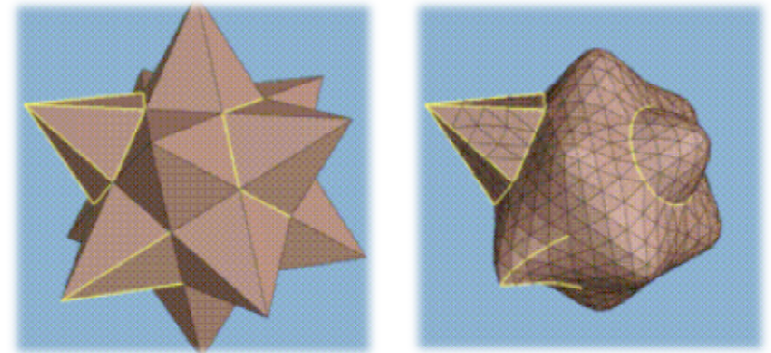
Subdivision Surfaces

▶ Video

  ▶ http://vimeo.com/2650080

Lecture Overview

- Advanced surface modeling

Advanced Shader Effects

- Environment mapping
- Toon shading

More Realistic Illumination

▸ In the real world:
  At each point in scene light arrives from all directions

  ▸ Not just from a few point light sources

  ▸ → Global Illumination is a solution, but computationally
    expensive

▸ Environment Maps

  ▸ Store "omni-directional" illumination as images

  ▸ Each pixel corresponds to light from a certain direction

Capturing Environment Maps

- "360 degrees" panoramic image
- Instead of 360 degrees panoramic image, take picture of mirror ball (light probe)

Light Probes by Paul Debevec
http://www.debevec.org/Probes/

Environment Maps as Light Sources

Simplifying Assumption

▶ Assume light captured by environment map is emitted from infinitely far away

▶ Environment map consists of directional light sources

  ▸ Value of environment map is defined for each **direction**, independent of position in scene

▶ Approach uses same environment map at each point in scene
  → Approximation!

Applications for Environment Maps

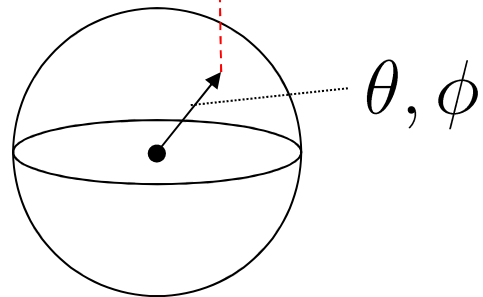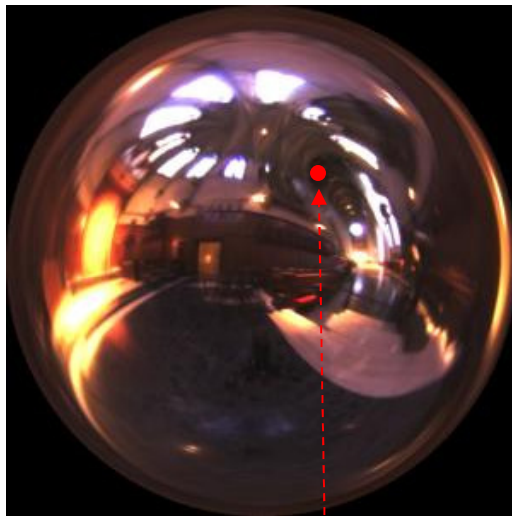▸ Use environment map as "light source"

*Global illumination with
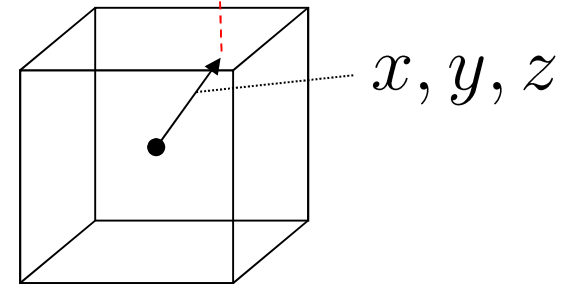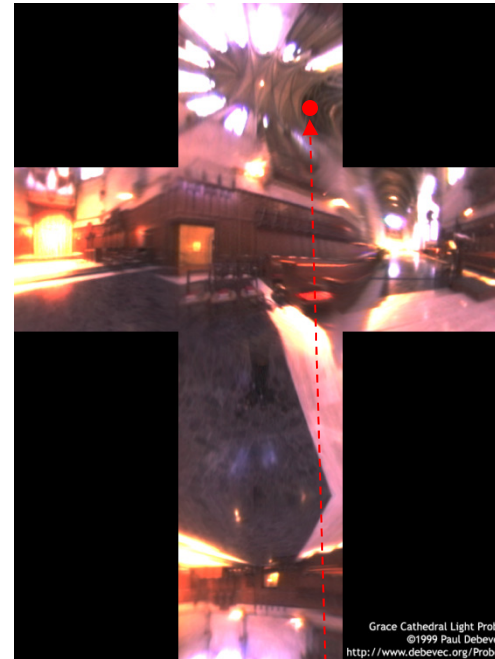pre-computed radiance transfer
[Sloan et al. 2002]*

*Reflection mapping
[Terminator 2, 1991]*

Cubic Environment Maps

- Store incident light on six faces of a cube instead of on sphere

θ, ϕ

Spherical map

x, y, z

Cube map

Grace Cathedral Light Probe
©1999 Paul Debevec
http://www.debevec.org/Probes

Cubic vs. Spherical Maps

▸ Advantages of cube maps:

 ▸ More even texel sample density causes less distortion, allowing for lower resolution maps

 ▸ Easier to dynamically generate cube maps for real-time simulated reflections

Bubble Demo

http://download.nvidia.com/downloads/nZone/demos/nvidia/Bubble.zip

Cubic Environment Maps

Cube map look-up

▶ Given: light direction (x,y,z)

▶ Largest coordinate component determines cube map face

▶ Dividing by magnitude of largest component yields coordinates within face

▶ In GLSL:

  ▶ Use (x,y,z) direction as texture coordinates to `samplerCube`

Reflection Mapping

▸ Simulates mirror reflection

▸ Computes reflection vector at each pixel

▸ Use reflection vector to look up cube map

▸ Rendering cube map itself is optional (application dependent)
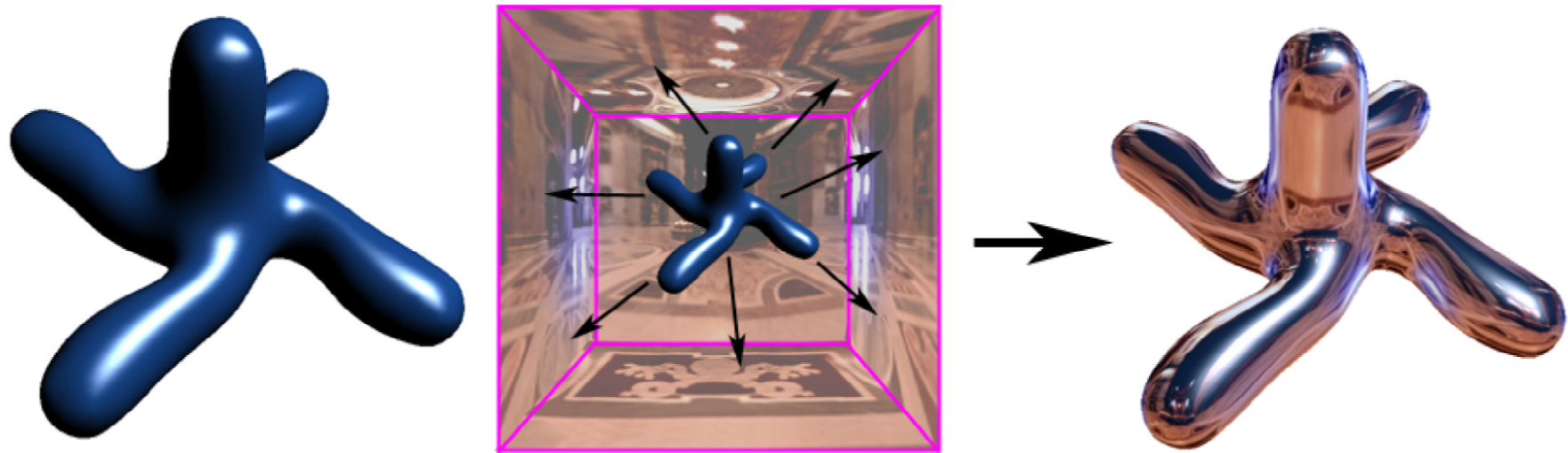
Reflection mapping

Reflection Mapping in GLSL

Application Setup

▶ Load and bind a cube environment map
```
glBindTexture(GL_TEXTURE_CUBE_MAP, …);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X,…);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X,…);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y,…);
…
glEnable(GL_TEXTURE_CUBE_MAP);
```

Reflection Mapping in GLSL

Vertex shader

- Compute viewing direction
- Reflection direction
  - Use `reflect` function
- Pass reflection direction to fragment shader

Fragment shader

- Look up cube map using interpolated reflection
  direction
  ```
  varying float3 refl;
  uniform samplerCube envMap;
  textureCube(envMap, refl);
  ```

Environment Maps as Light Sources

▸ Covered so far: shading of a specular surface

→ How do you compute shading of a diffuse surface?

Diffuse Irradiace Environment Map

▸ Given a scene with k directional lights, light directions $d_1..d_k$ and intensities $i_1..i_k$, illuminating a diffuse surface with normal n and color c

▸ Pixel intensity B is computed as: $B = c \sum_{j=1..k} \max(0, d_j \cdot n) i_j$

▸ Cost of computing B proportional to number of texels in environment map!

▸ → Precomputation of diffuse reflection

▸ Observations:

  ▸ All surfaces with normal direction n will return the same value for the sum

  ▸ The sum is dependent on just the lights in the scene and the surface normal

▸ Precompute sum for any normal n and store result in a second environment map, indexed by surface normal

▸ Second environment map is called *diffuse irradiance environment map*

▸ Allows to illuminate objects with arbitrarily complex lighting environments with single texture lookup

Diffuse Irradiace Environment Map

▶ **Two cubic environment maps:**

- ▶ Reflection map
- ▶ Diffuse map
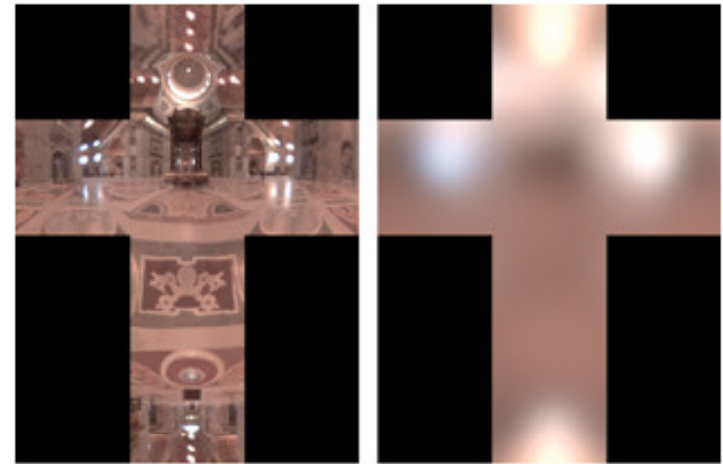
▶ **Diffuse shading vs. shading w/diffuse map**

Image source: http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter10.html

Lecture Overview

▶ Advanced surface modeling

Advanced Shader Effects

▶ Environment mapping

▶ Toon shading

Toon Shading

Source: Wikipedia

- A.k.a. Cel Shading ("Cel" is short for "celluloid" sheets, on which animation was hand-drawn)
- Gives any 3D model a cartoon-style look
- Emphasizes silhouettes
- Discrete steps for diffuse shading, highlights
- Non-photorealistic rendering method (NPR)
- Programmable shaders allow real-time performance

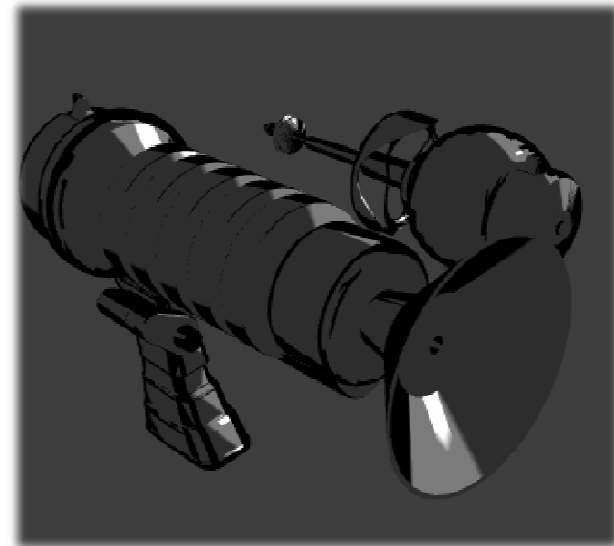© Michael Welter

Off-line toon shader
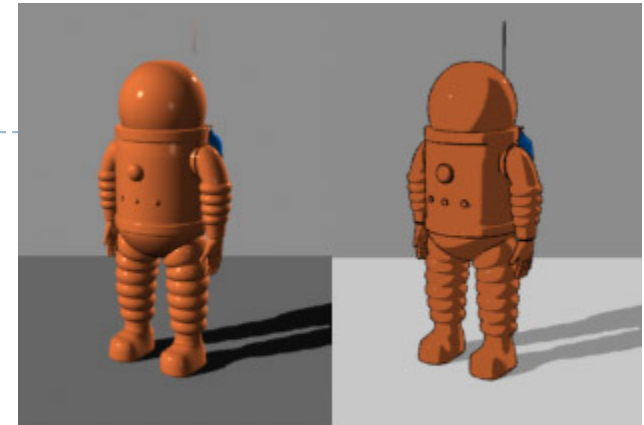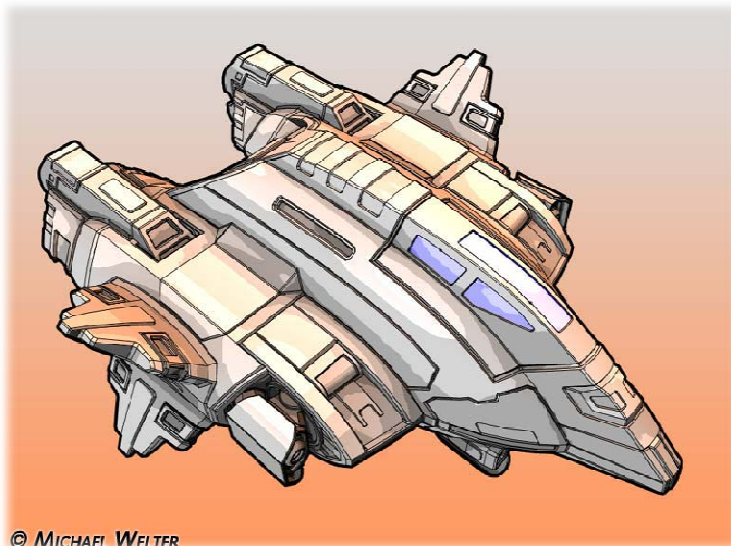
GLSL toon shader
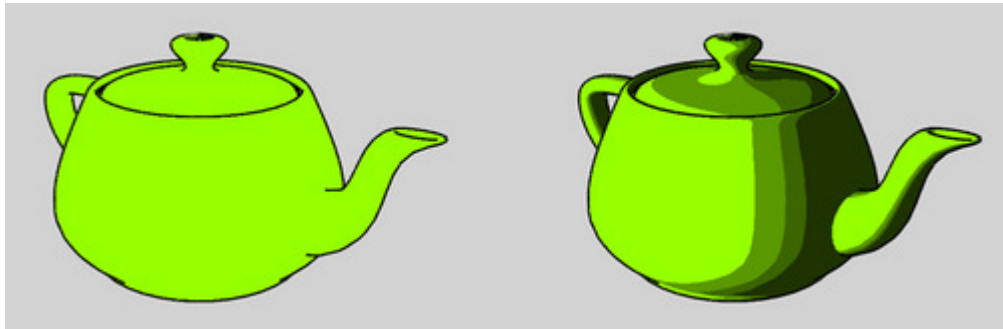
Approach

- Start with regular 3D model
- Apply two rendering tricks:
  - Silhouette edges
    - Emphasize pixels with normals perpendicular to viewing direction.
  - Discretized shading
    - Conventional (smooth) lighting values calculated for each pixel, then mapped to a small number of discrete shades.
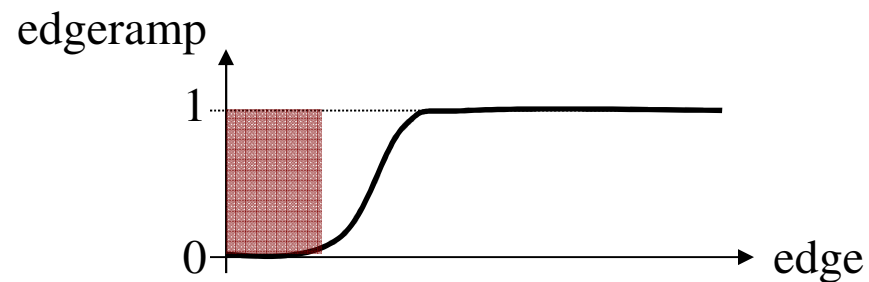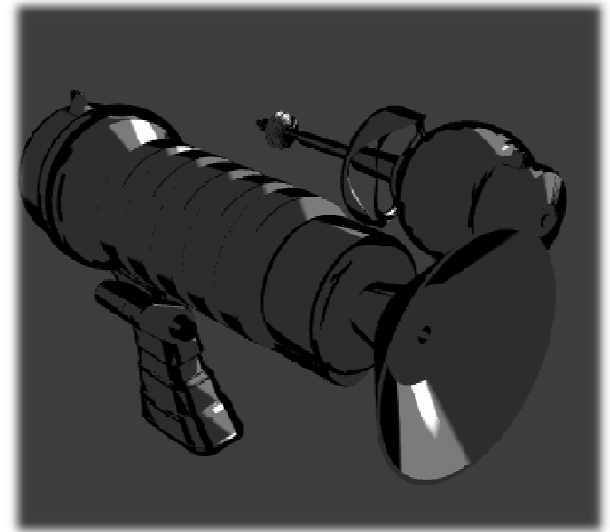

Source: Wikipedia

Silhouette Edges

▸ **Silhouette edge detection**

    ▸ Compute dot product of viewing direction **v** and normal **n**

$$\text{edge} = \max(0, \mathbf{n} \cdot \mathbf{v})$$

    ▸ Use 1D texture to define edge ramp

uniform sample1D edgeramp; e=texture1D(edgeramp,edge);

Discretized Shading

▸ **Compute diffuse and specular shading**
$$\text{diffuse} = \mathbf{n} \cdot \mathbf{L} \qquad \text{specular} = (\mathbf{n} \cdot \mathbf{h})^s$$

▸ **Use 1D textures** diffuseramp, specularramp **to map diffuse and specular shading to colors**

▸ **Final color:**

```
uniform sampler1D diffuseramp;
uniform sampler1D specularramp;
c = e * (texture1D(diffuse,diffuseramp) +

texture1D(specular,specularramp));
```

Toon Shading Demo

http://www.bonzaisoftware.com/npr.html