CSE 167:
Introduction to Computer Graphics
Lecture #14: Shader Effects

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2013

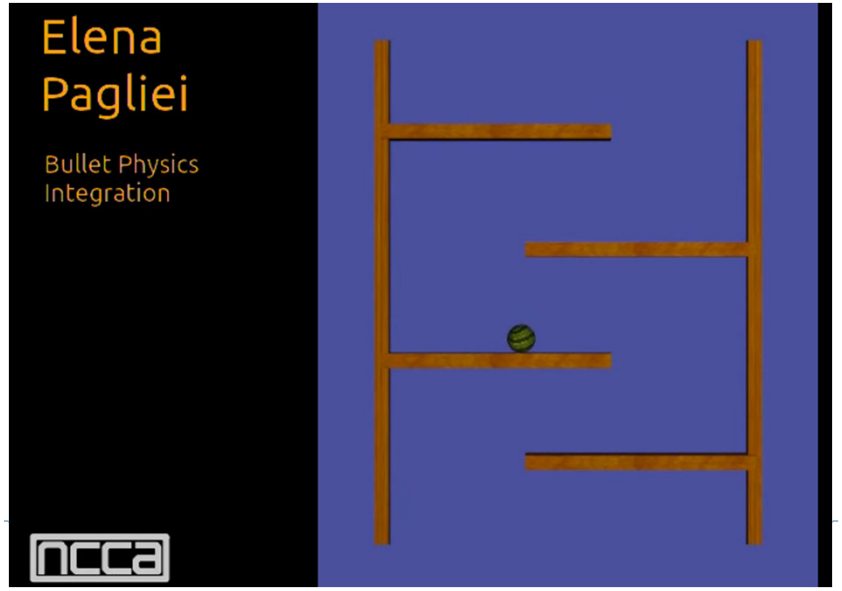
Announcements

- Homework assignment 6 due tomorrow
  - Note: image needs to stretch to height of surface of revolution and wrap entirely around it

- Monday: Midterm exam review
- Next Thursday: Midterm exam #2
- Homework assignment #7 (Final Project) will go on-line tomorrow, due December 12th
  - To be done in teams of 2 or 3
  - To be demonstrated during final exam slot from 3-6pm in room CSE 1202
  - I will provide my laptop with Windows 7, but laptop submissions are encouraged

Video

- Final Project motivation
- SDAGE PPP 1st Year Assignment
  - A selection of assignments from BSc Software Development for Animation and Games course held at NCCA, Bournemouth University, UK for the 2011/2012 academic year
  - http://vimeo.com/42326732



Lecture Overview

Advanced Shader Effects

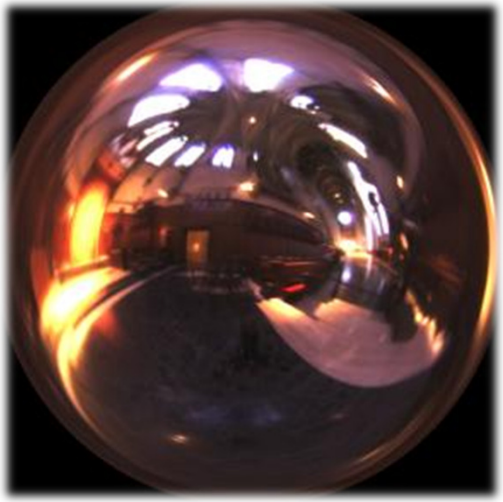
- Environment mapping
- Toon shading

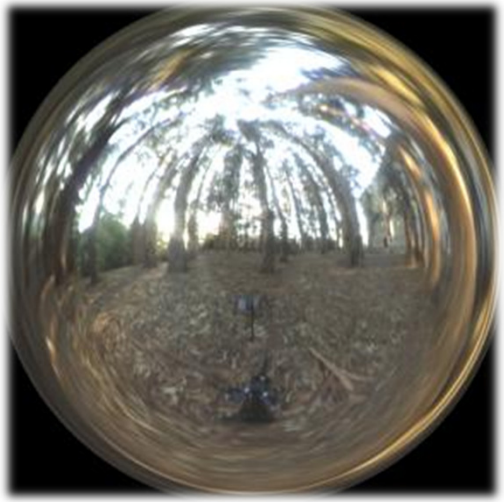
More Realistic Illumination

- 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

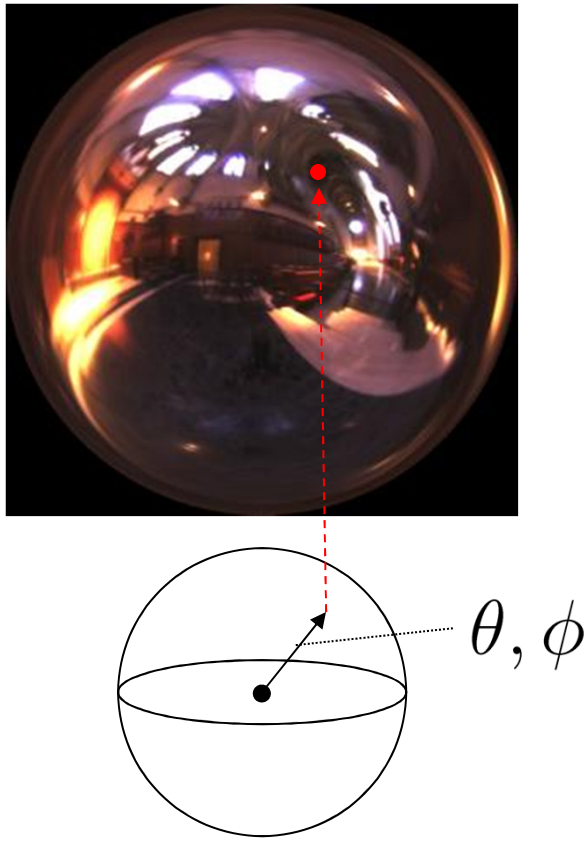
- Use environment map as "light source"

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

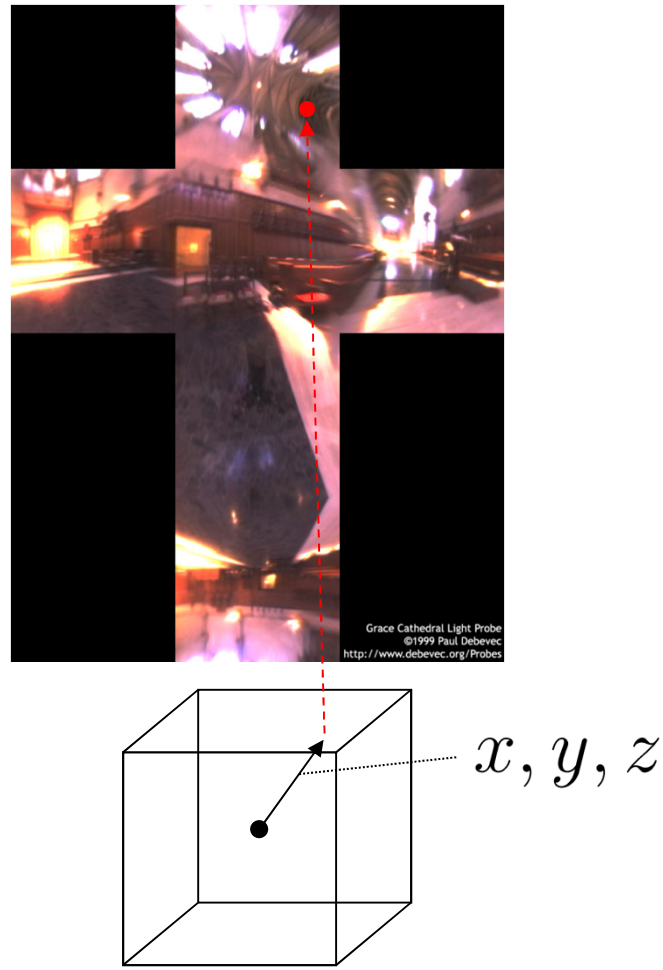
Reflection mapping [Terminator 2, 1991]

Cubic Environment Maps

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



Spherical map

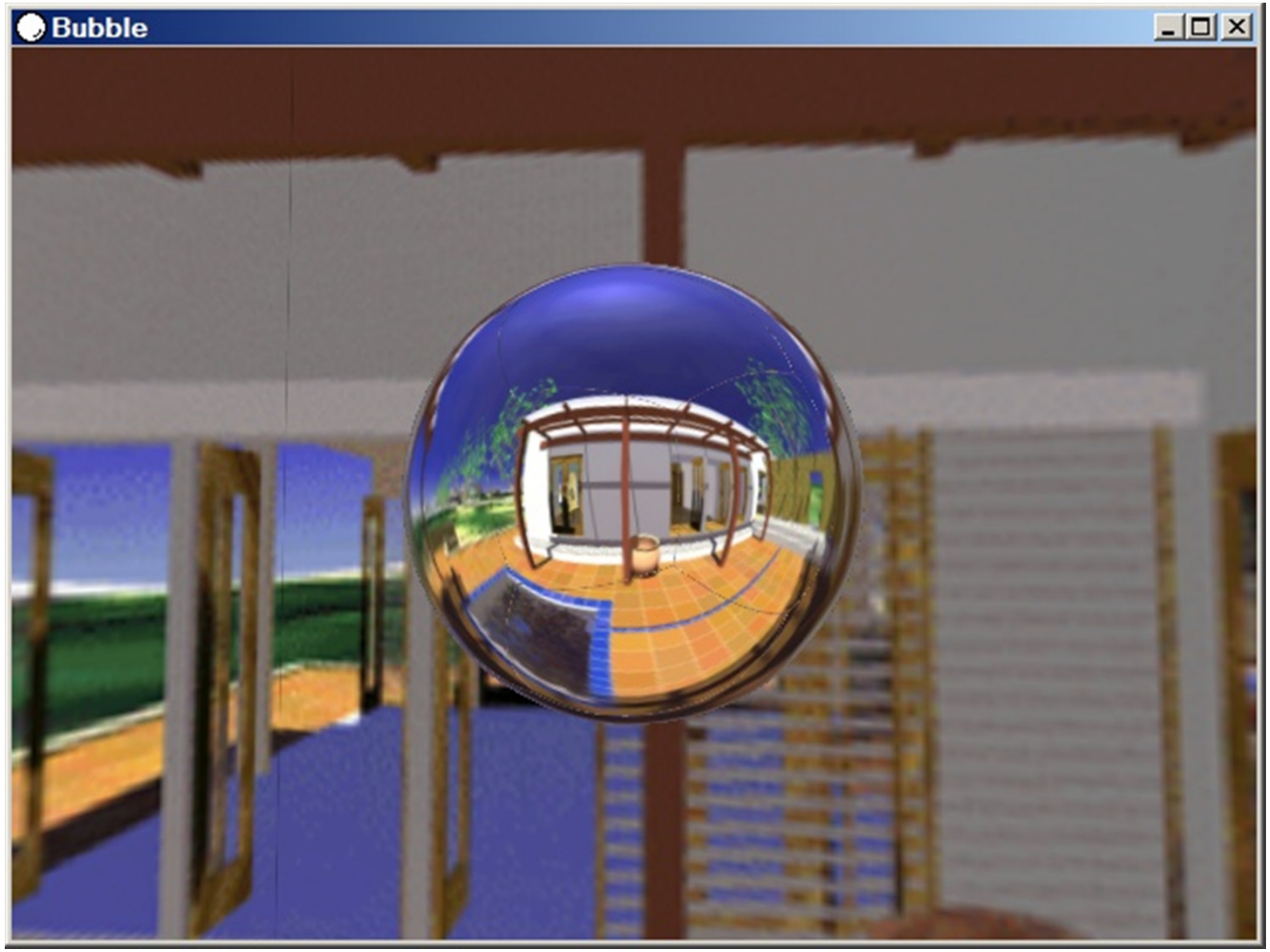


Cube map

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

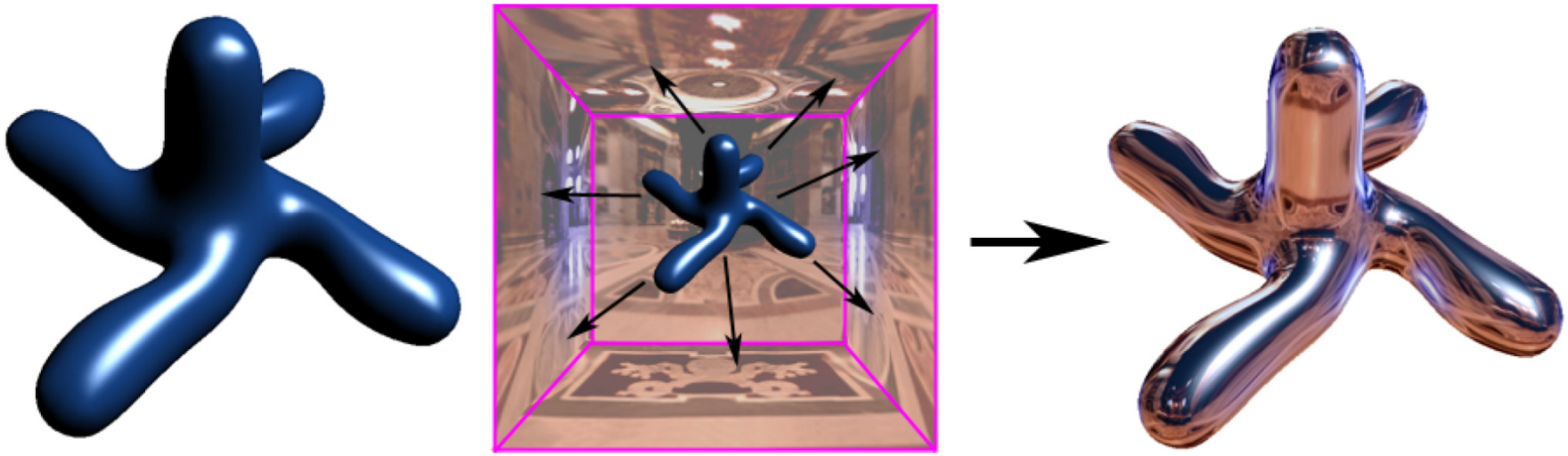
Cubic Environment Maps

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

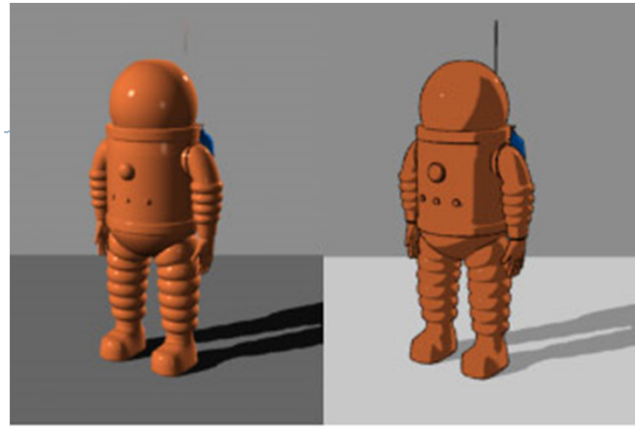
Lecture Overview

Advanced Shader Effects

- Environment mapping
- Toon shading

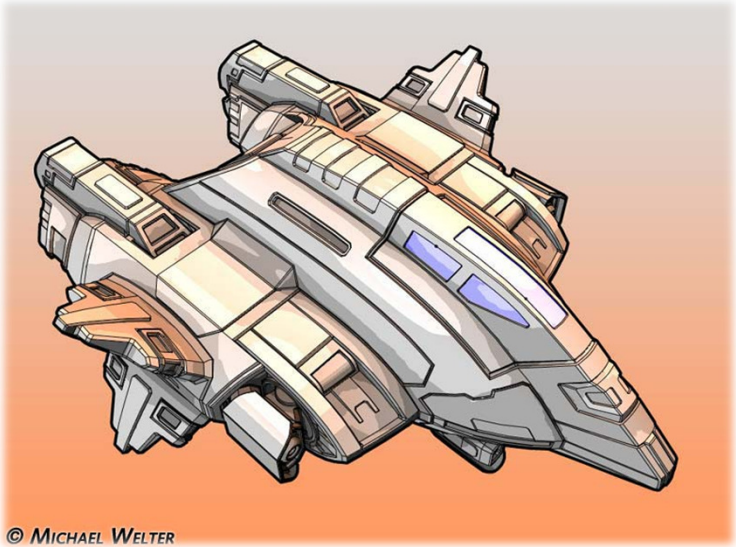
Toon Shading

- 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

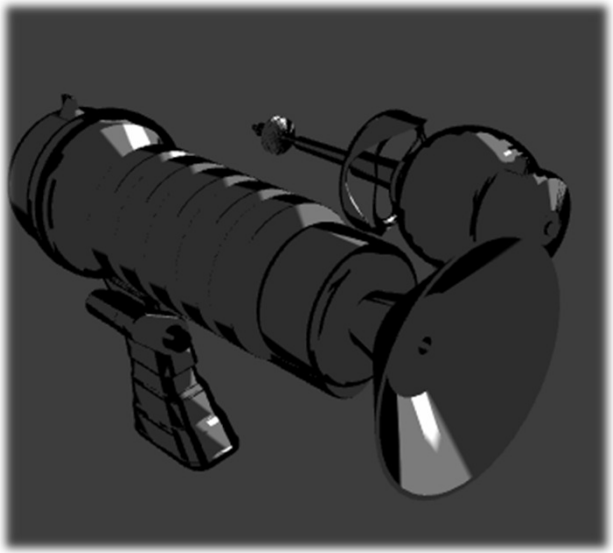


plastic shader toon shader

Source: Wikipedia



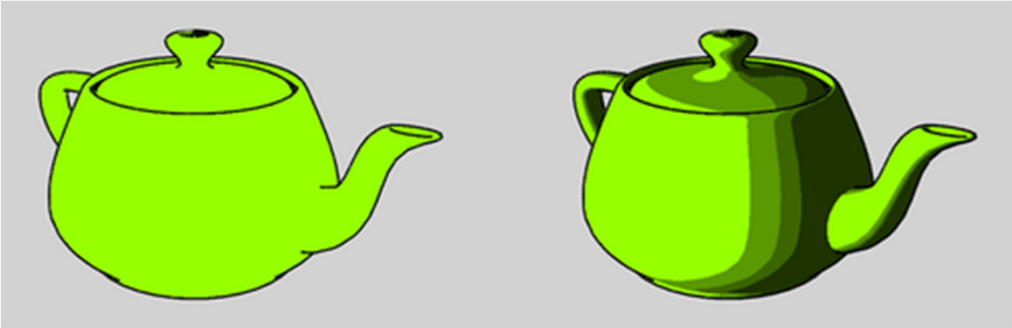
Off-line toon shader



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.

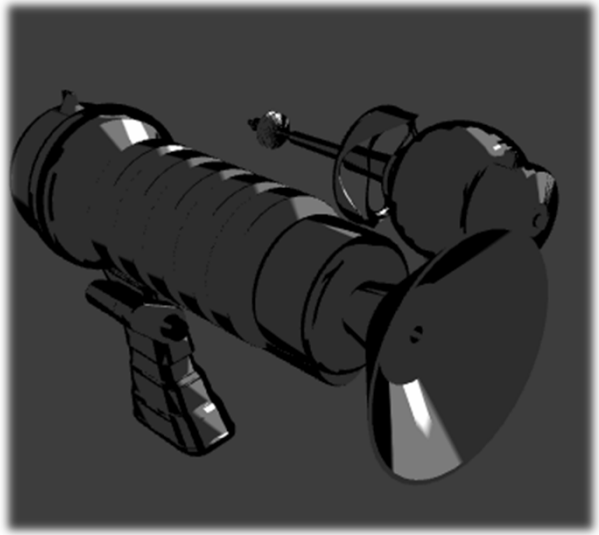


Source: Wikipedia

Silhouette Edges

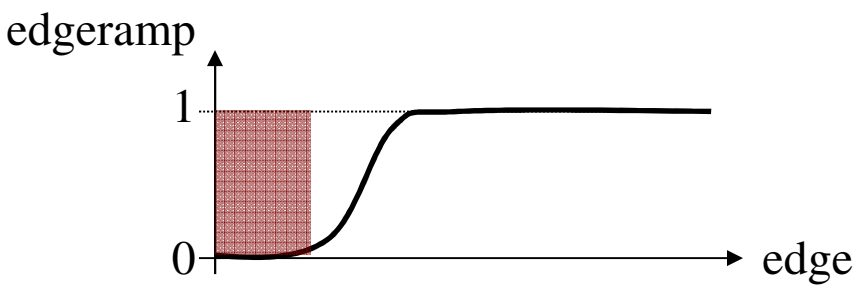
- Silhouette edge detection
  - Compute dot product of viewing direction $\mathbf{v}$ and normal $\mathbf{n}$

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



  - 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