

CSE 167:
Introduction to Computer Graphics
Lecture #15: Shader Effects

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2011

Announcements

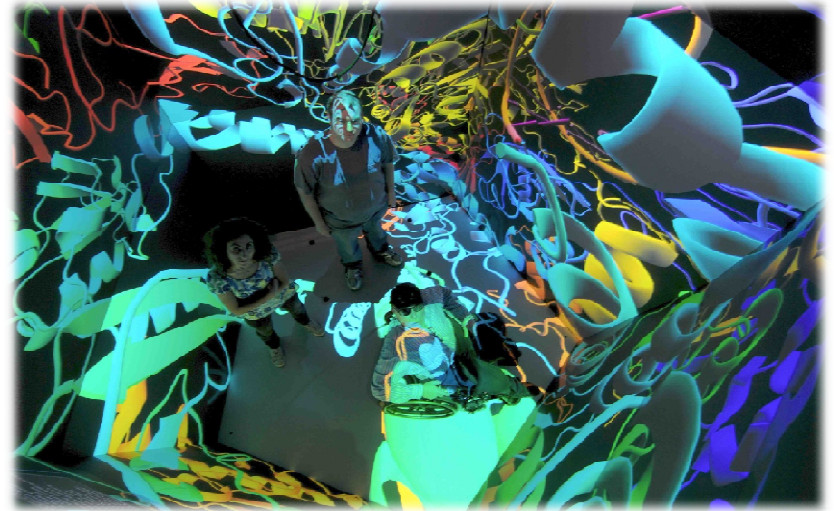
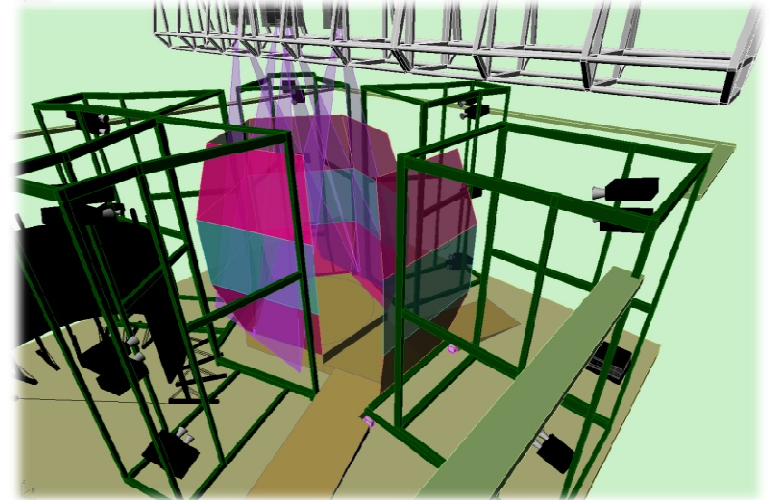
- ▶ Homework assignment #6 due Friday, Nov 18
- ▶ CAPE: on-line, email notification at beginning of week 9
 - ▶ Period: Monday 11/21 to Monday 12/5
 - ▶ <http://www.cape.ucsd.edu>
- ▶ Final project description is on-line

StarCAVE Tour

- ▶ Location: Atkinson Hall, 1st floor
- ▶ Time: Today after class, until 5pm
- ▶ Location:
 - ▶ Immersive Visualization Laboratory
 - ▶ 1st floor Atkinson Hall
 - ▶ Turn right at main entrance
 - ▶ Door on left, will be open

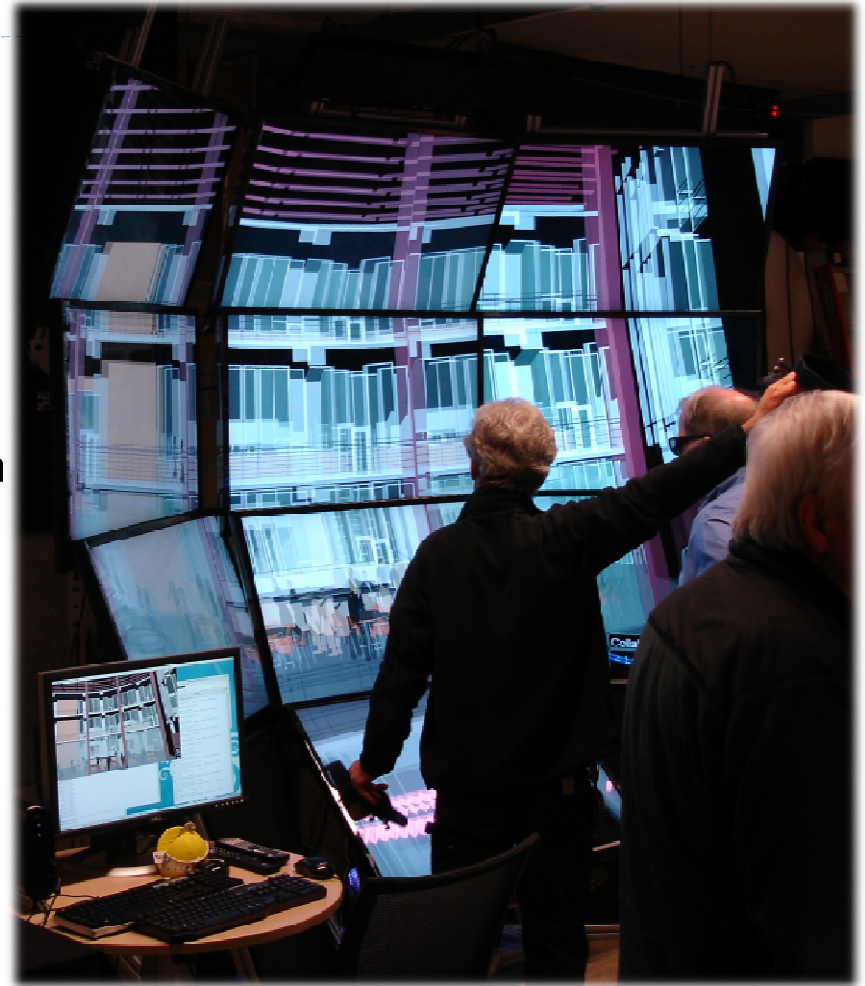
The StarCAVE

- ▶ 18 Dell XPS 710 PCs
- ▶ Dual Nvidia Quadro 5600 graphics cards
- ▶ CentOS 5.3 Linux
- ▶ 34 JVC HD2k projectors (1920x1080 pixels):
~34 megapixels per eye
- ▶ 360 degrees immersion
- ▶ Passive stereo, circular polarization
- ▶ 15 screens on 5 walls, ~8 x 4 foot each, plus
floor projection
- ▶ 4-camera optical tracking system



NexCAVE

- ▶ 10 42" JVC Xpol displays:
LCD panels with polarizing filters,
1920x1080 pixels
- ▶ 5 rendering PCs: Dell XPS 710
- ▶ Nvidia GeForce 480 GPUs
- ▶ 2-camera ART TrackPack tracking system



Lecture Overview

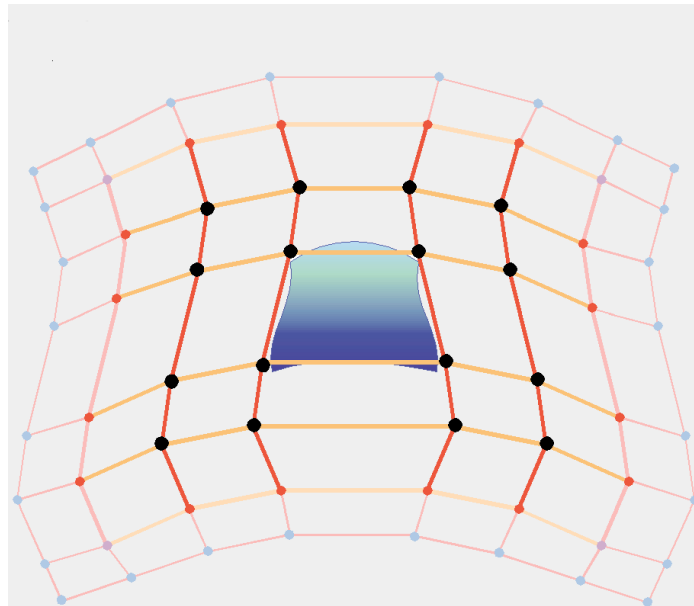
- ▶ **Advanced surface modeling**

Advanced Shader Effects

- ▶ Environment mapping
- ▶ Toon shading

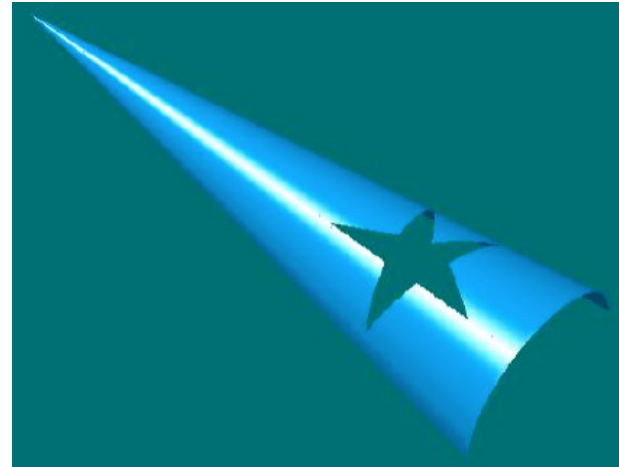
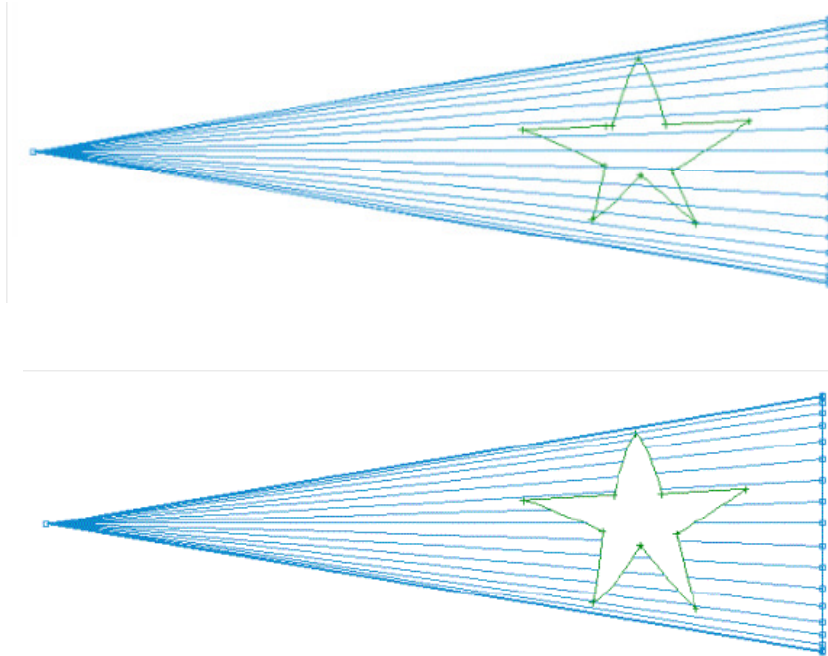
Advanced Surface Modeling

- ▶ B-spline/NURBS patches
- ▶ For the same reason as using B-spline/NURBS curves
 - ▶ More flexible (can model spheres)
 - ▶ Better continuity



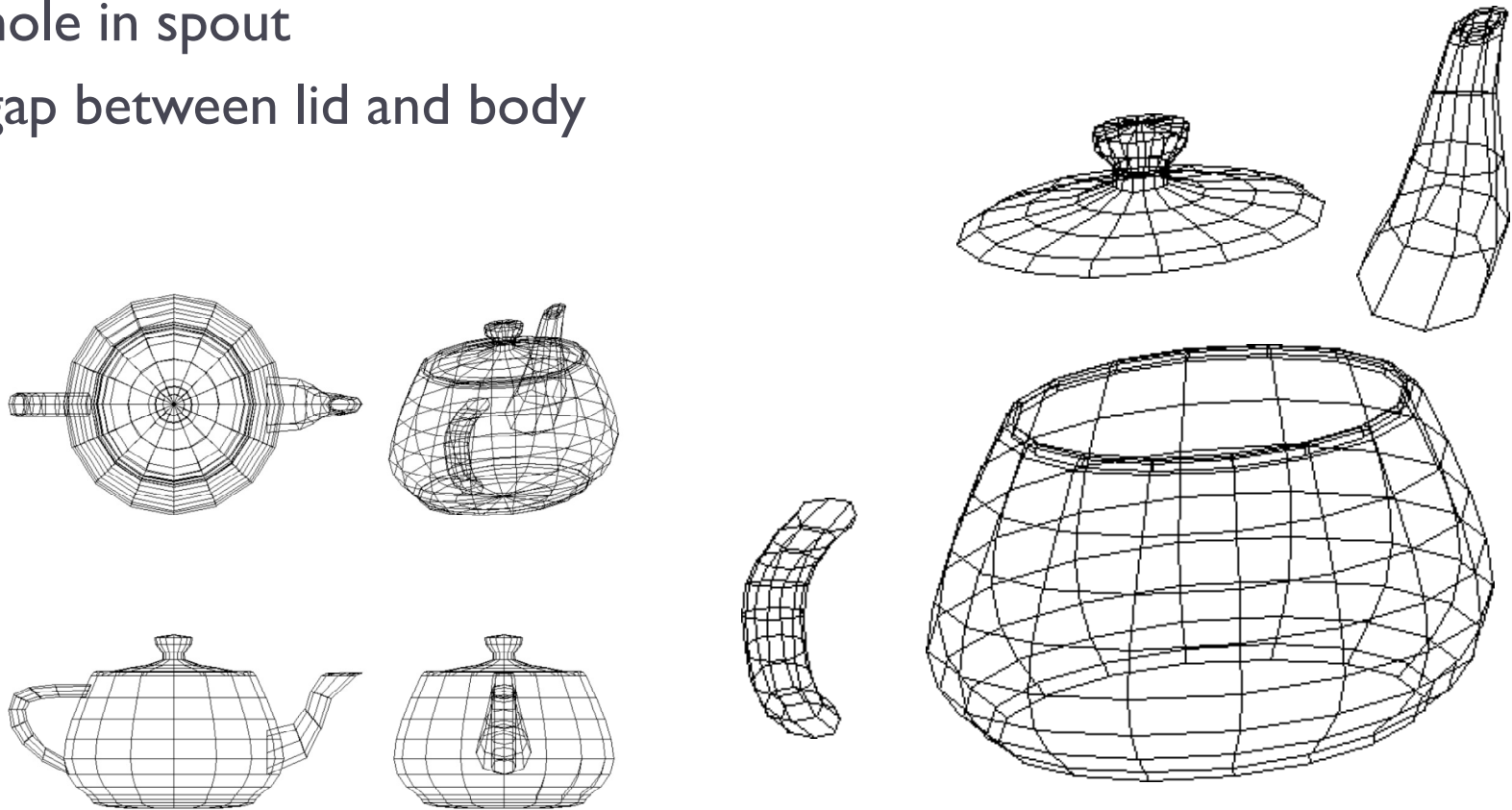
Advanced Surface Modeling

- ▶ Trim curves: cut away part of surface
 - ▶ Implement as part of tessellation/rendering



Modeling Headaches

- ▶ Original teapot is not “water tight”
 - ▶ spout & handle intersect with body
 - ▶ hole in spout
 - ▶ gap between lid and body



Modeling Headaches

- ▶ **NURBS surfaces are versatile**

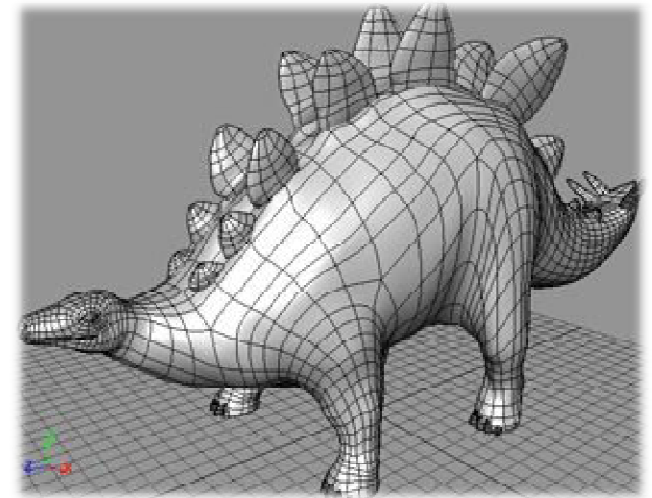
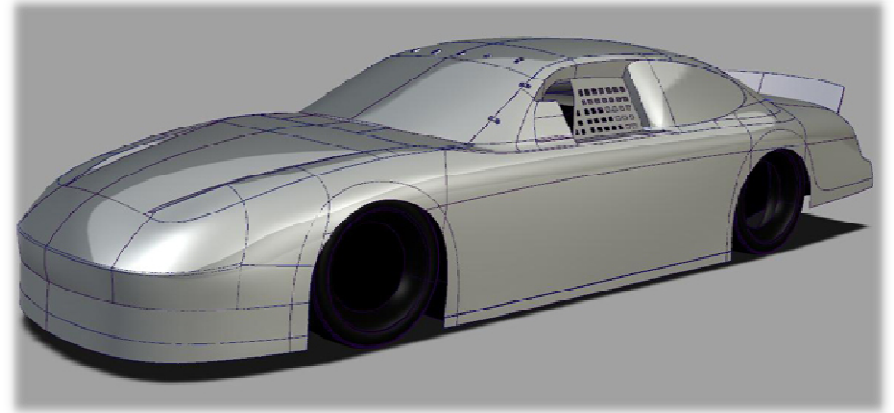
- ▶ Conic sections
- ▶ Can blend, merge, trim...

- ▶ **But:**

- ▶ Any surface will be made of quadrilateral patches (quadrilateral topology)

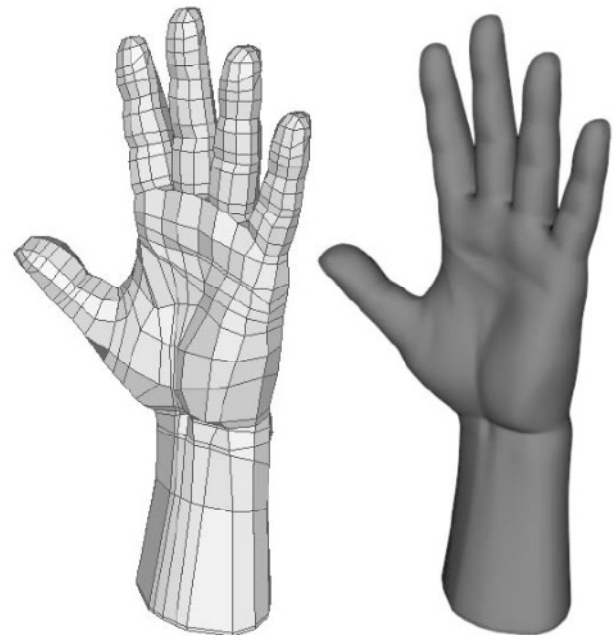
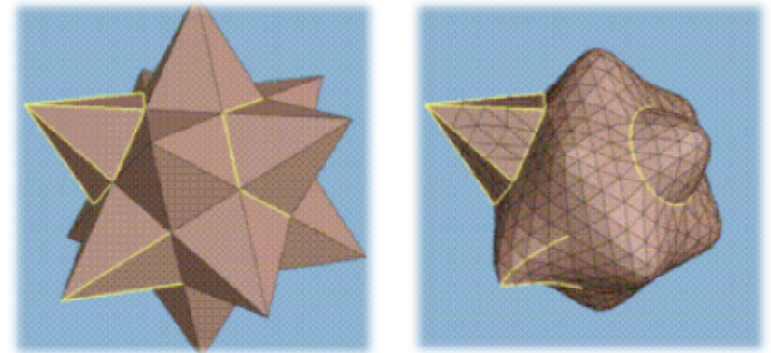
- ▶ **This makes it hard to**

- ▶ Join or abut curved pieces
- ▶ Build surfaces with complex topology or structure



Subdivision Surfaces

- ▶ Arbitrary mesh of control points, not quadrilateral topology
 - ▶ No global u, v parameters
- ▶ Can make surfaces with arbitrary topology or connectivity
- ▶ Work 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



Lecture Overview

- ▶ Advanced surface modeling

Advanced Shader Effects

- ▶ **Environment mapping**
- ▶ Toon shading

More Realistic Illumination

- ▶ In real world:
 - At each point in scene light arrives from all directions
 - ▶ Not just from 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
precomputed radiance transfer

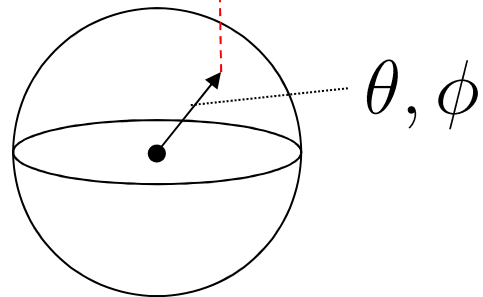
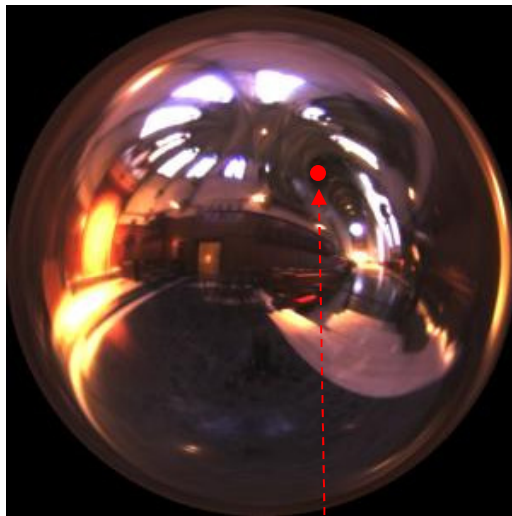
[Sloan et al. 2002]



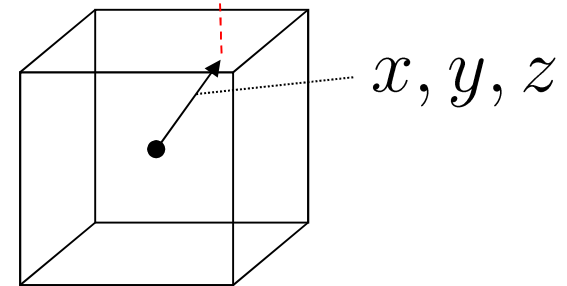
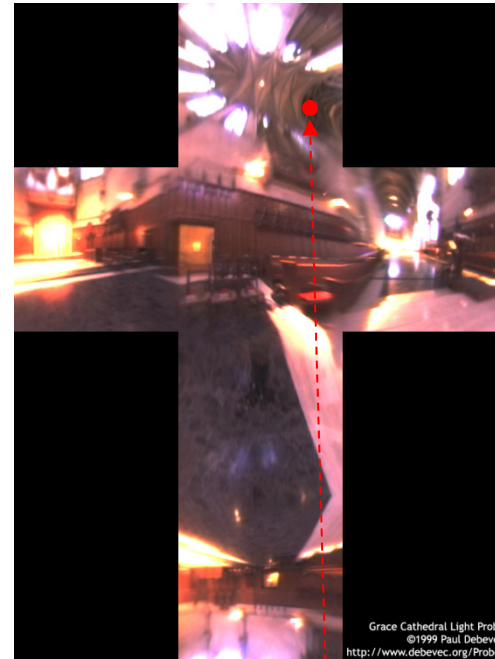
Reflection mapping
[Terminator 2, 1991]

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>

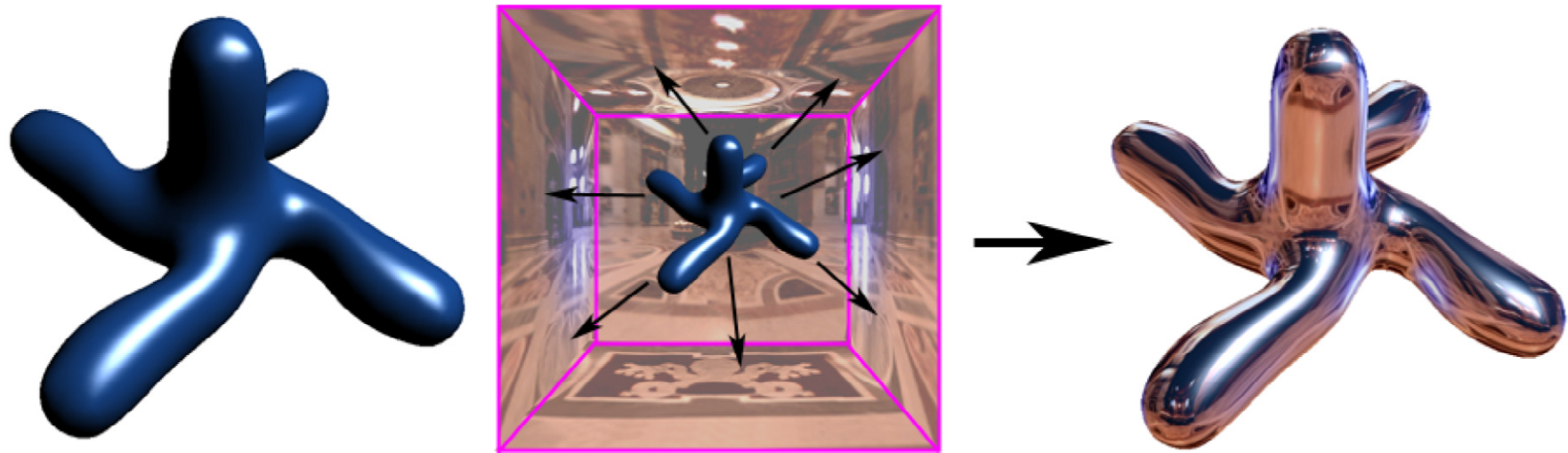
Cubic Environment Maps

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?

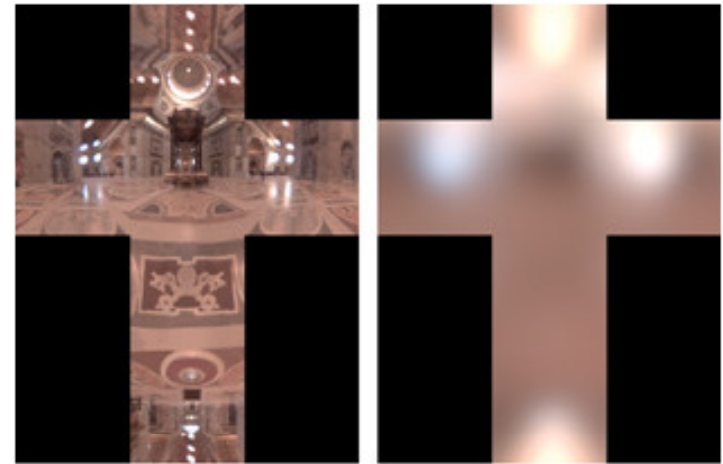
Diffuse Irradiance Environment Map

- ▶ 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 Irradiance Environment Map

- ▶ Two cubic environment maps:

- ▶ reflection map
 - ▶ diffuse map



- ▶ Diffuse shading vs. shading w/diffuse map



Source: http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter10.html

Lecture Overview

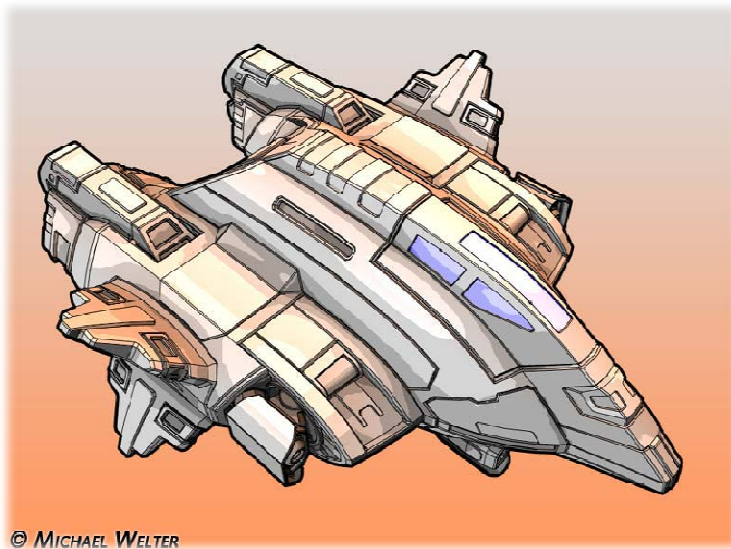
- ▶ Advanced surface modeling

Advanced Shader Effects

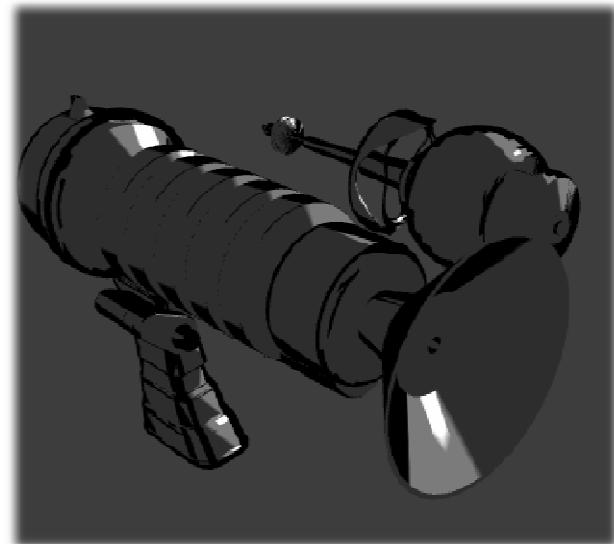
- ▶ Environment mapping
- ▶ **Toon shading**

Toon Shading

- ▶ A.k.a. Cel Shading
- ▶ Simple cartoon-style shader
- ▶ Emphasizes silhouettes
- ▶ Discrete steps for diffuse shading, highlights
- ▶ Non-photorealistic rendering method (NPR)



Off-line toon shader



GLSL toon shader

Toon Shading Demo



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

Toon Shading

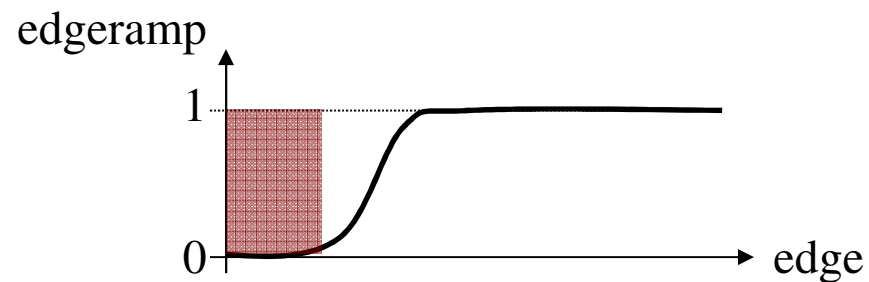
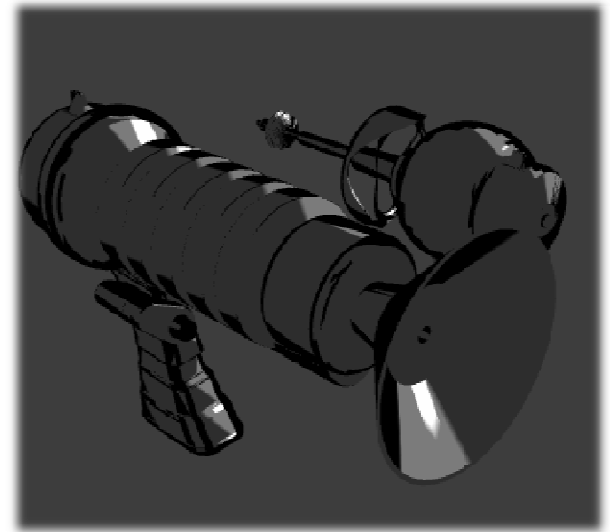
- ▶ Silhouette edge detection

- ▶ 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 sampler1D edgeramp; e=texture1D(edgeramp,edge);`



Toon Shading

- ▶ Compute diffuse and specular shading

$$\text{diffuse} = \mathbf{n} \cdot \mathbf{L} \quad \text{specular} = (\mathbf{n} \cdot \mathbf{h})^s$$

- ▶ 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));
```

Next Lecture

- ▶ Ambient Occlusion
- ▶ Shadow mapping