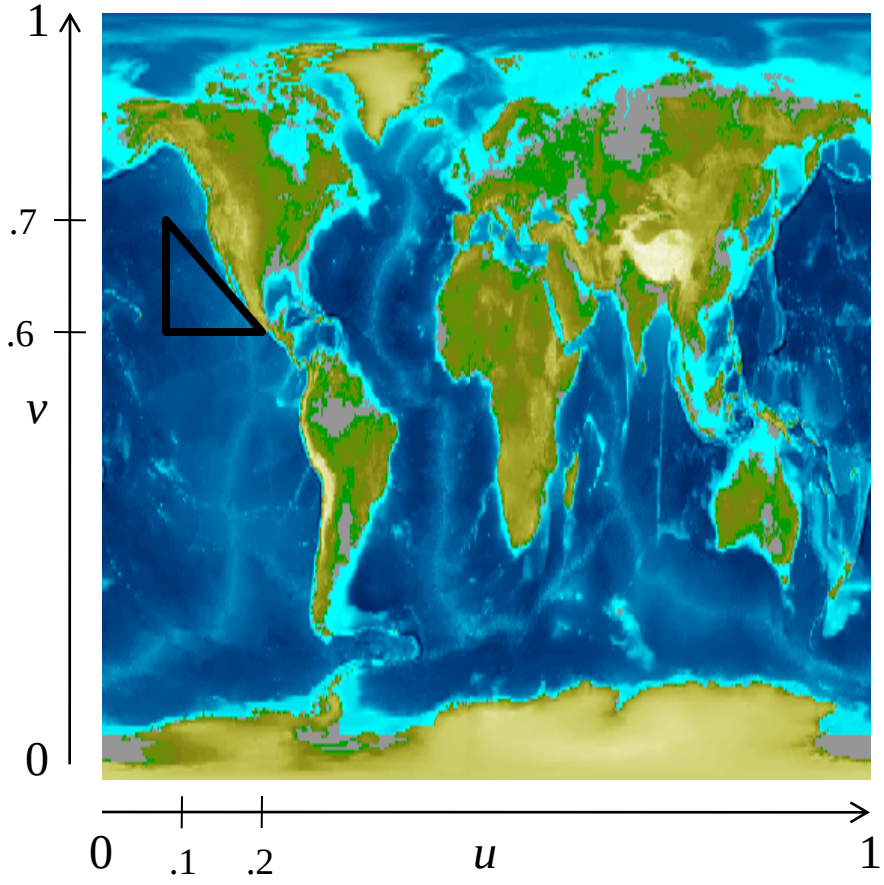
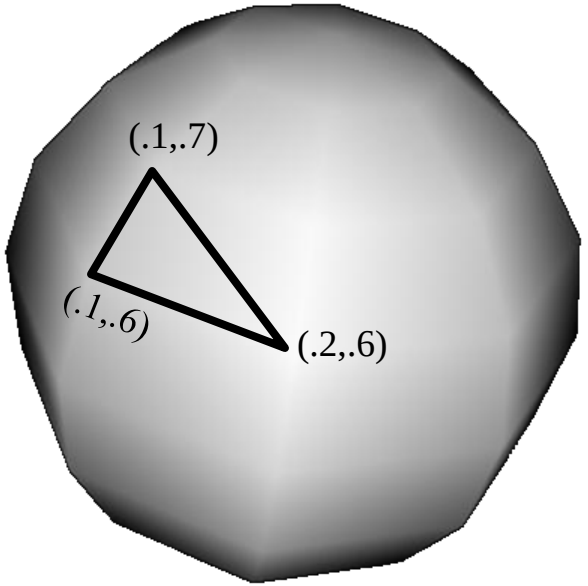
Texture Coordinates

CS418 Computer Graphics

John C. Hart

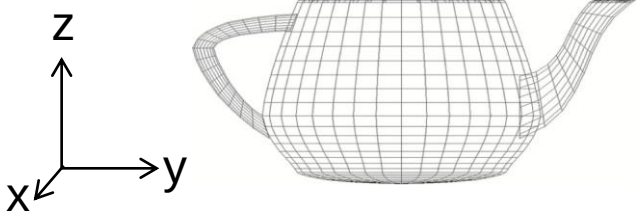
Texture Mapping



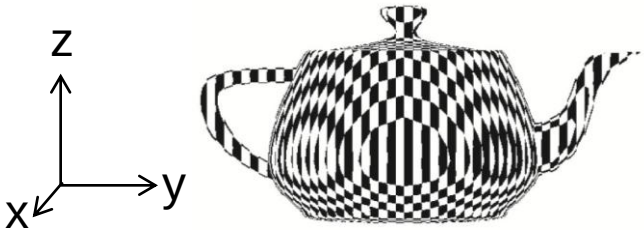


TexGen

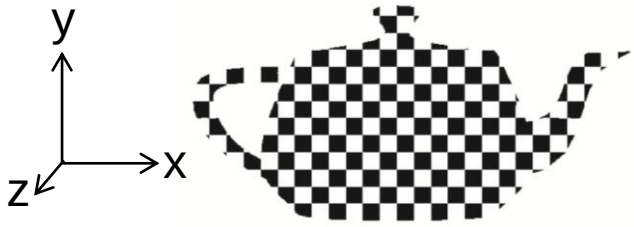
```
GLfloat xplane[] = {1,0,0,0};
GLfloat yplane[] = {0,1,0,0};
glTexGenfv(GL_S,GL_OBJECT_PLANE, xplane);
glTexGenfv(GL_T,GL_OBJECT_PLANE, yplane);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
```




```
glTexGeni(GL_S,
   GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
glTexGeni(GL_T,
   GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
```




```
glTexGeni(GL_S,
   GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
glTexGeni(GL_T,
   GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
```



Contouring

- Set texgen mode to object linear
- Define a plane
- Load a white texture with one black stripe

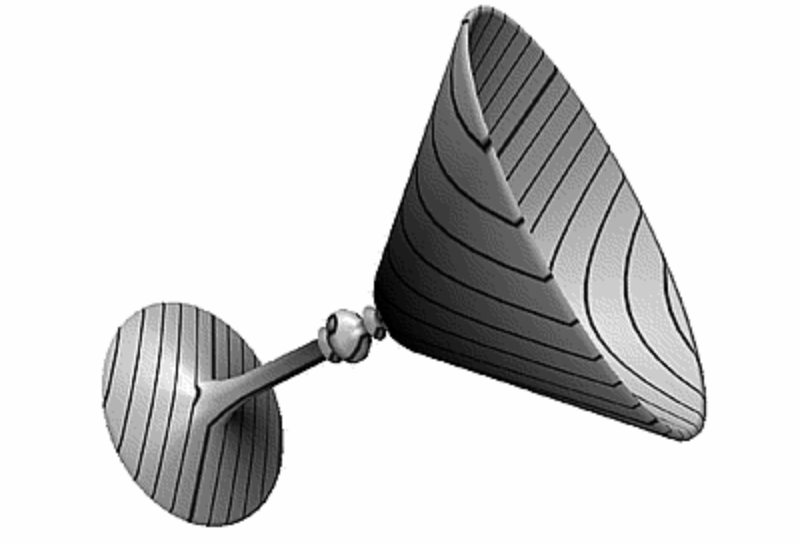
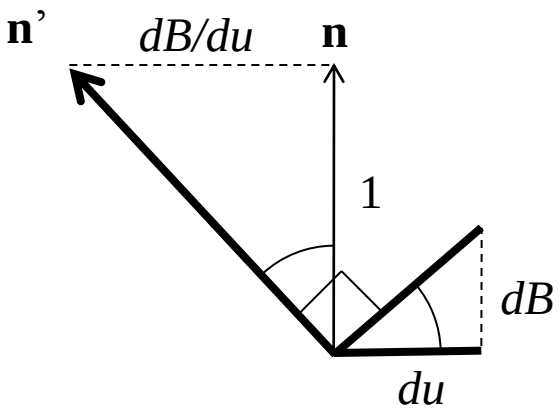
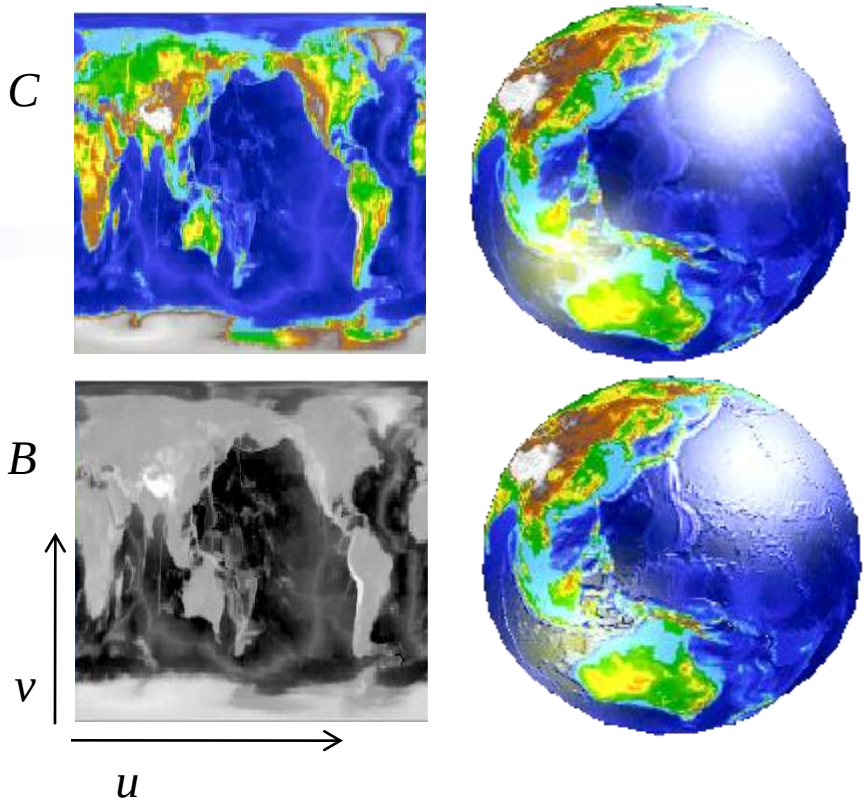


Figure 4. Showing distance from a plane with Contouring.

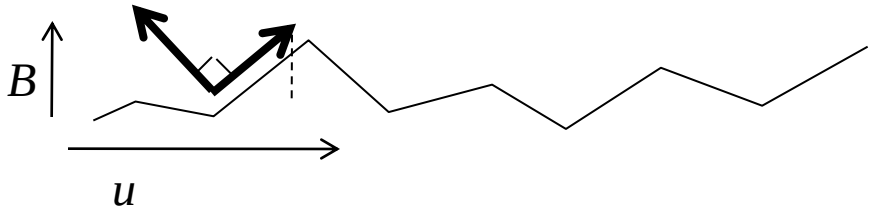
Texture Mapping as a Fundamental Drawing Primitive
Paul Haeberli and Mark Segal
Graphica Obcura, June 1993

Bump Mapping

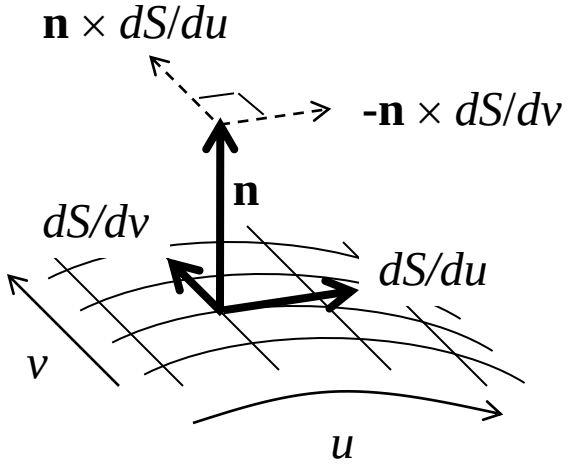




$$dB/du = (B(u+h) - B(u))/h$$



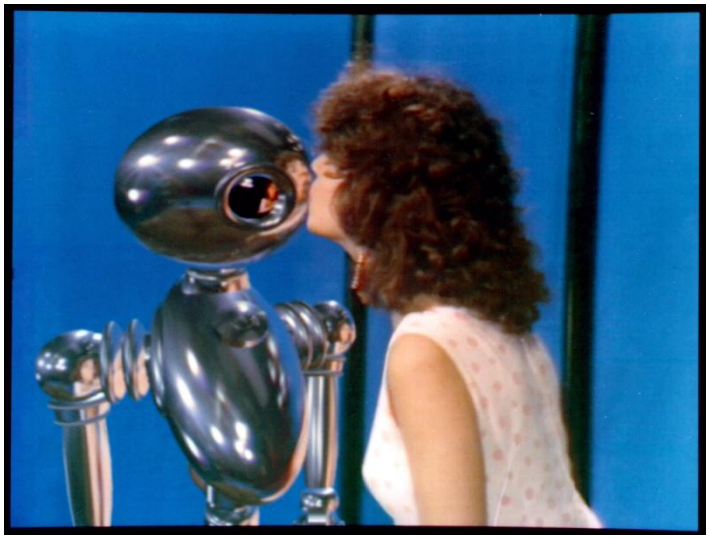
$$\mathbf{n}' = \mathbf{n} - (dB/du)\,\mathbf{n} \times dS/dv + (dB/dv)\,\mathbf{n} \times dS/du$$



Environment Mapping

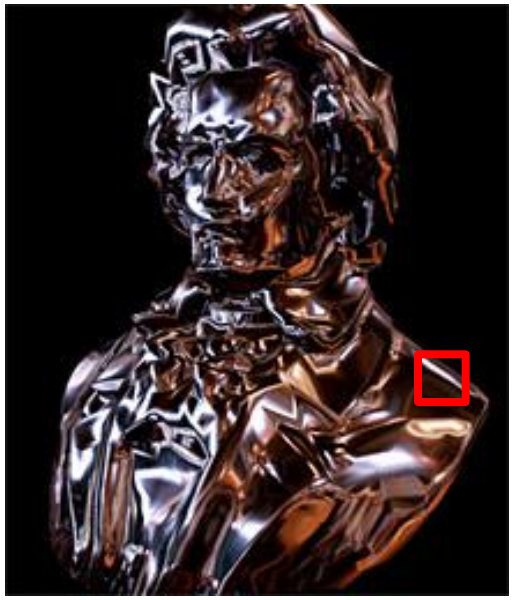
- Stores view from a point in all directions
- When rendering a fragment, compute reflection vector for eye (like the light reflection vector for specular reflection)
- Lookup prestored color and display that color as the reflection
- How can we store a precomputed view in all directions as a texture map?

- See the Story of Reflection Mapping: http://debevec.org/ReflectionMapping/



Sphere Map Idea

- Photograph a reflective sphere (like a garden gazing ball) in whatever environment you like
- All possible normals you can see appear somewhere on the sphere
- Store image of reflective sphere as a texture map
- When rendering a point on a reflective object, find the point on the sphere with the same surface normal, and display the sphere's color on the object
- Need to convert surface normal to texture coordinates



Sphere Map Geometry

How to make a sphere map:

•For each point in disk:

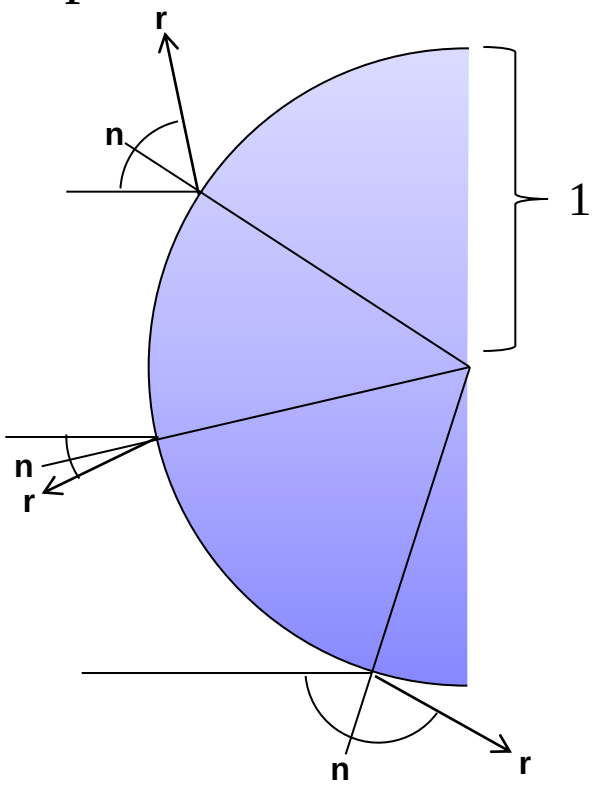
(x,y) s.t. $x^2 + y^2 \leq 1$

•**n** = (x,y,$(1-x^2-y^2)^{½}$)

•**r** = reflect((0,0,1),**n**)

•Determine color in **r** direction

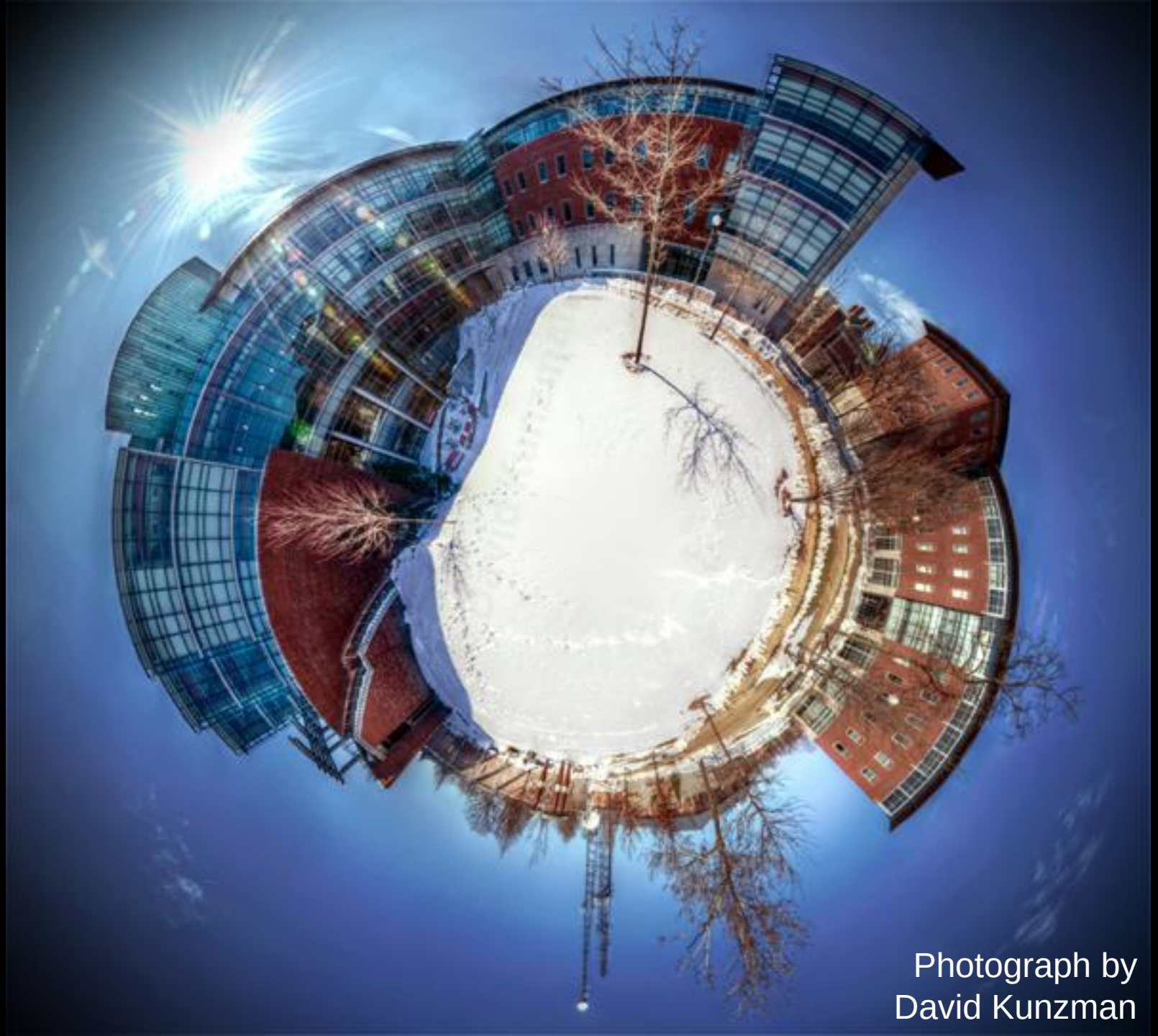
- Ray tracing
- Images
- Photograph



OpenGL Sphere Map

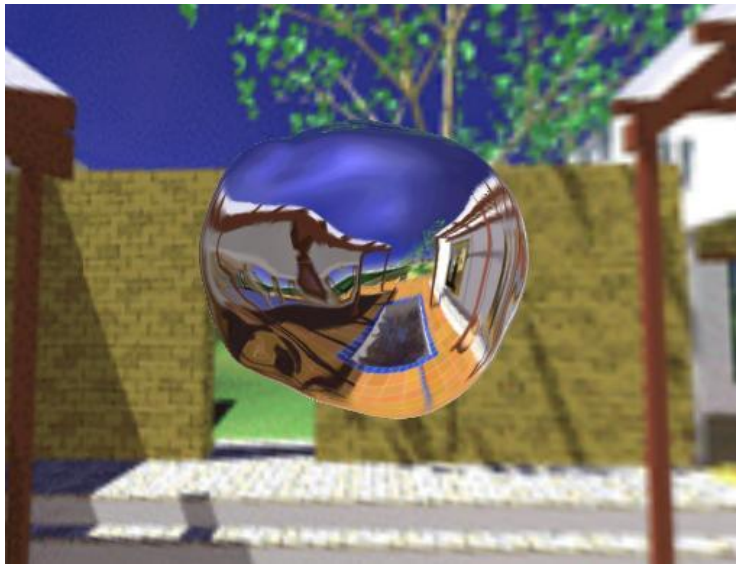
```
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);

glTexGeni(GL_S,
   GL_TEXTURE_GEN_MODE,GL_SPHERE_MAP);
glTexGeni(GL_T,
   GL_TEXTURE_GEN_MODE,GL_SPHERE_MAP);
```

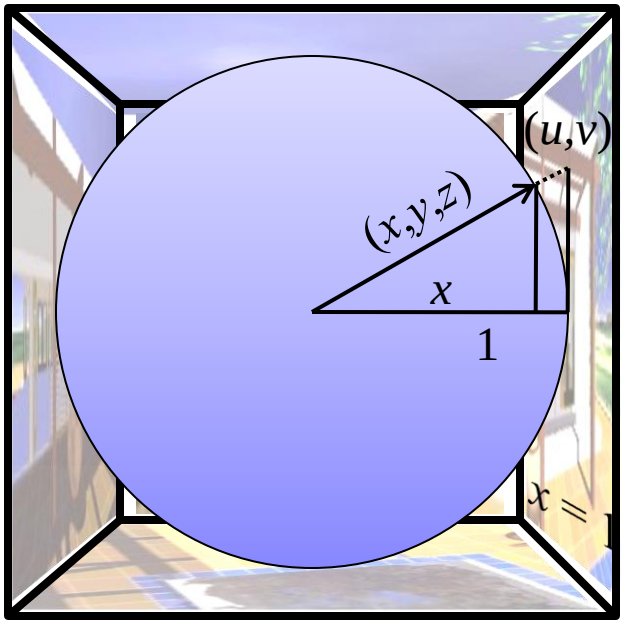


Photograph by
David Kunzman

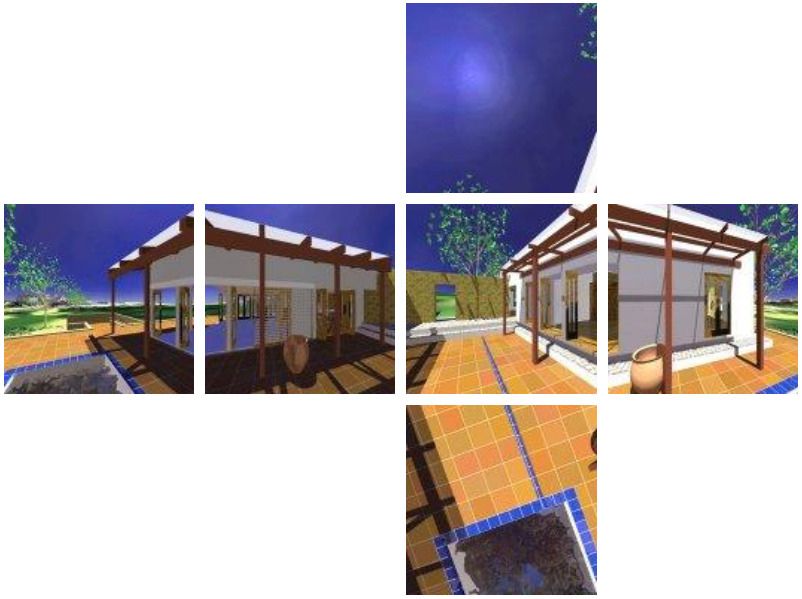
OpenGL Cube Map



- Sphere map creates distortion near the edges of the sphere
- Can use six different texture images to form a cube map

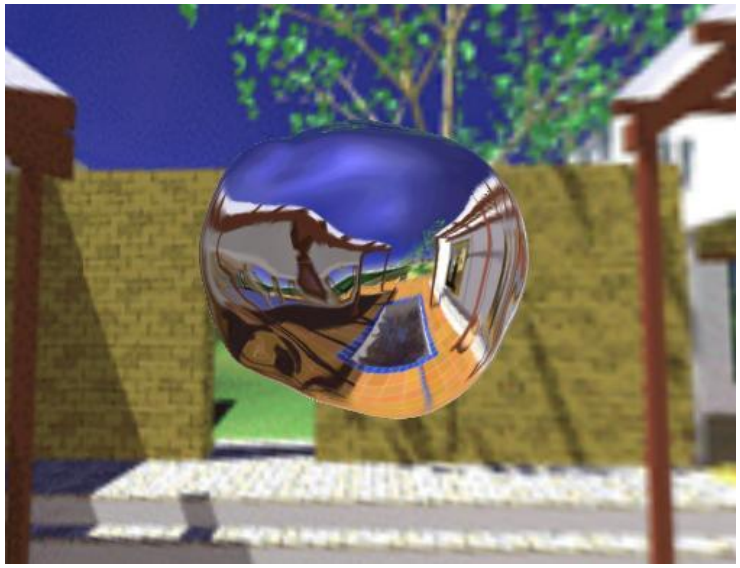


$u = y/x$
$v = z/x$



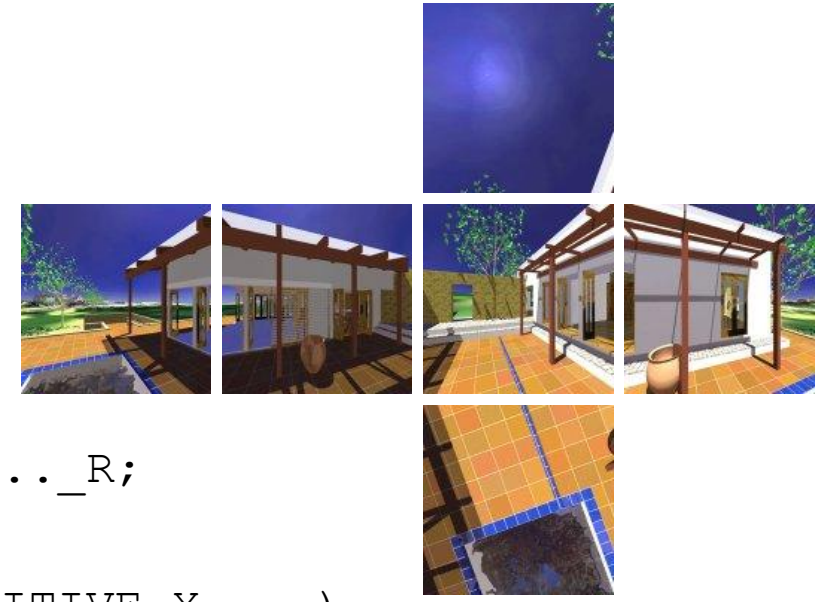
http://developer.nvidia.com/object/cube_map_ogl_tutorial.html

OpenGL Cube Map



- Sphere map creates distortion near the edges of the sphere
- Can use six different texture images to form a cube map

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE,
   GL_REFLECTION_MAP);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE,
   GL_REFLECTION_MAP);
glTexGeni(GL_R, GL_TEXTURE_GEN_MODE,
   GL_REFLECTION_MAP);
glEnable(GL_TEXTURE_GEN_S); ..._T; ..._R;
glEnable(GL_TEXTURE_CUBE_MAP);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X,...)
```



http://developer.nvidia.com/object/cube_map_ogl_tutorial.html

Texture Shading

- OpenGL lighting equation

$$I = k_a L_a + k_d L_d (N \cdot L) + k_s L_s (V \cdot R)^n$$
$$I = k_a L_a + k_d L_d (N \cdot L) + k_s L_s (V \cdot R)^n$$
$$I = k_a L_a + k_d L_d (N \cdot L) + k_s L_s (V \cdot R)^n$$

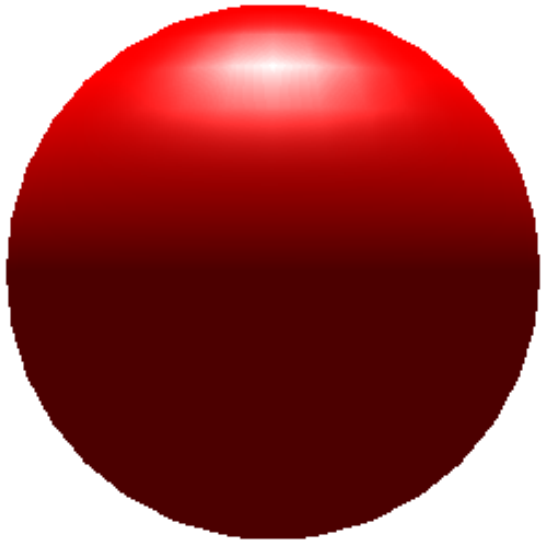
- Set texcoords of vertices to:

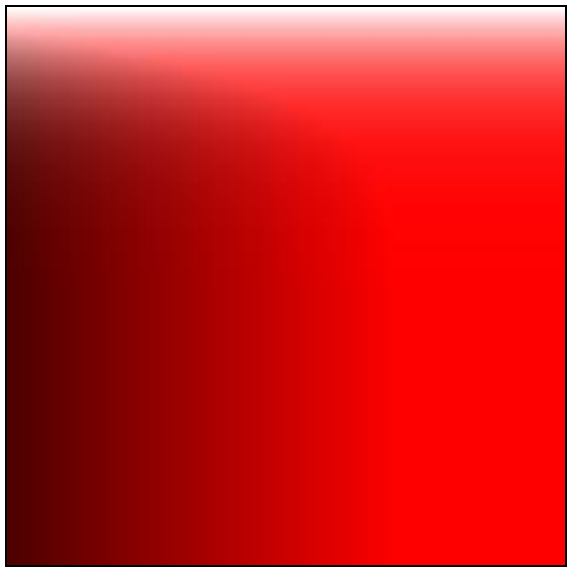
$$s = N \cdot L, \quad t = V \cdot R$$

- Create a texture by evaluating OpenGL lighting equation for each texel

$$C(s,t) = k_a L_a + k_d L_d s + k_s L_s t^n$$
$$\text{for all } 0 \le s,t \le 1$$

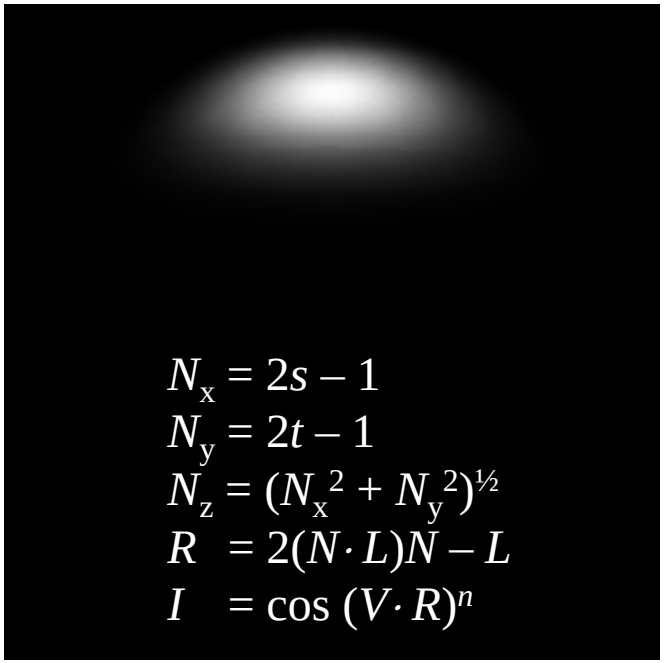
- Disable OpenGL lighting!
- Still just Gouraud color interpolation
- Must recompute texture coords when view changes





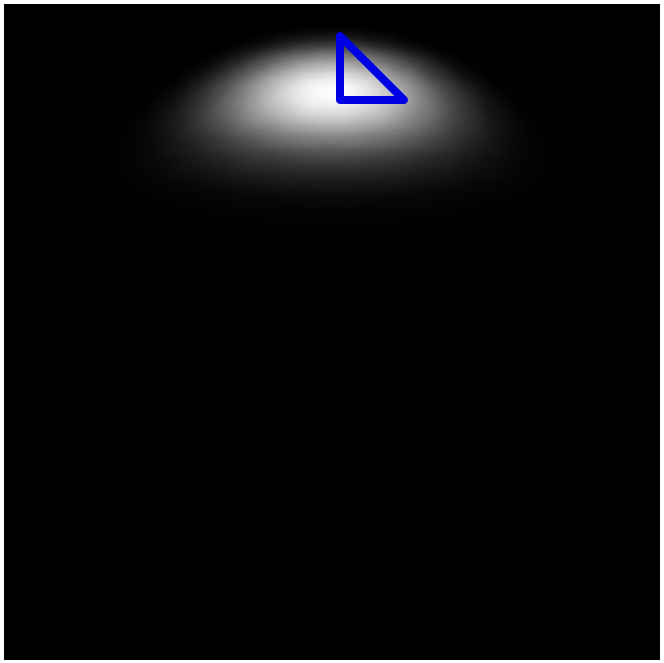
Phong Map

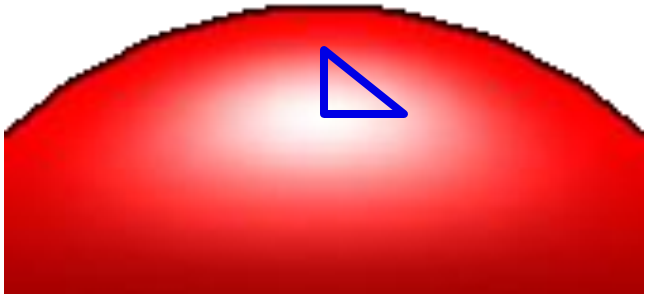
- The sphere map is an environment map stored as the reflection of a scene on sphere
- Not only stores scene, stores the *reflection* of the scene
  - Let the scene be a single point light source
  - Render a Phong specular highlight on a sphere
  - Use rendered sphere as an environment spheremap
- Texcoord interpolation samples highlight through face interior
- Texture is view/light dependent

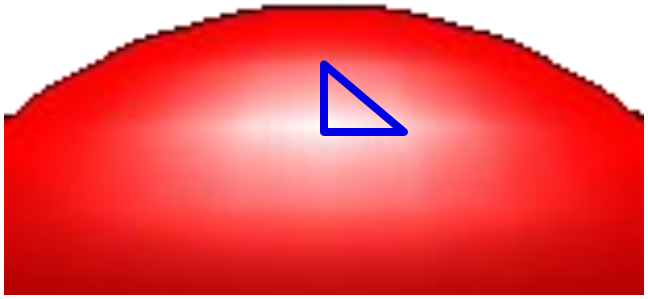


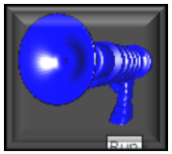
Phong Map

- The sphere map is an environment map stored as the reflection of a scene on sphere
- Not only stores scene, stores the *reflection* of the scene
  - Let the scene be a single point light source
  - Render a Phong specular highlight on a sphere
  - Use rendered sphere as an environment spheremap
- Texcoord interpolation samples highlight through face interior

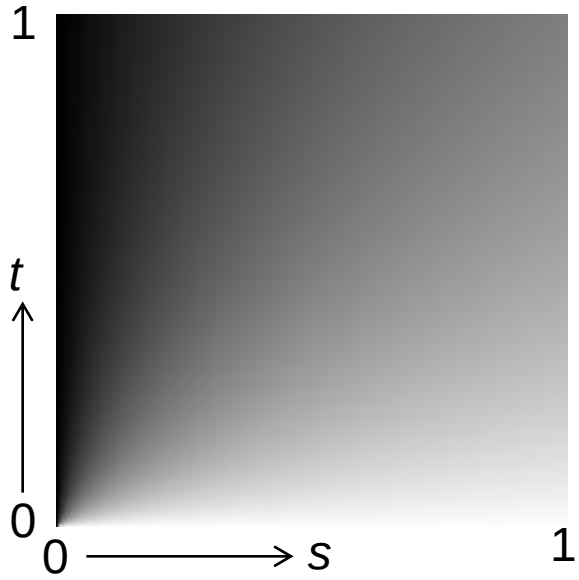








A Skin Texture Shader



- Skin appears softer than Lambertian reflectance because of subsurface scattering

- Seeliger lighting model

$$I = (N \cdot L) / (N \cdot L + N \cdot V)$$

- Implement as a texture shader

$$s = N \cdot L$$

$$t = N \cdot V$$

$$C = s/(s+t)$$