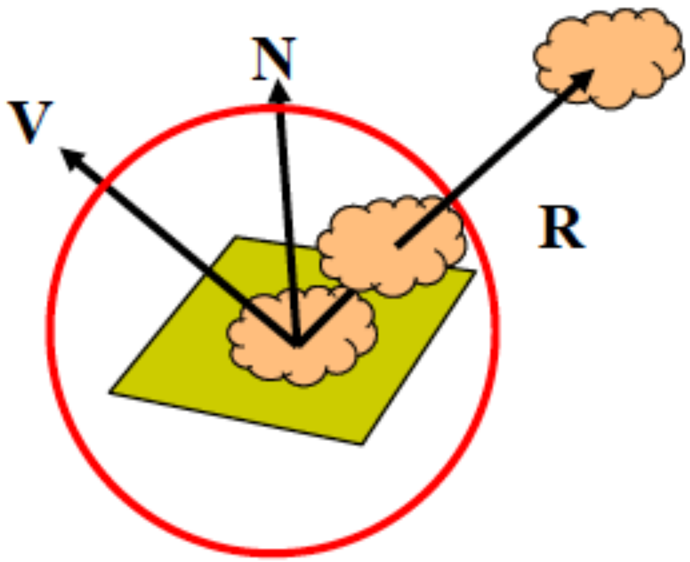
CS 418: Interactive Computer Graphics

Environment Mapping

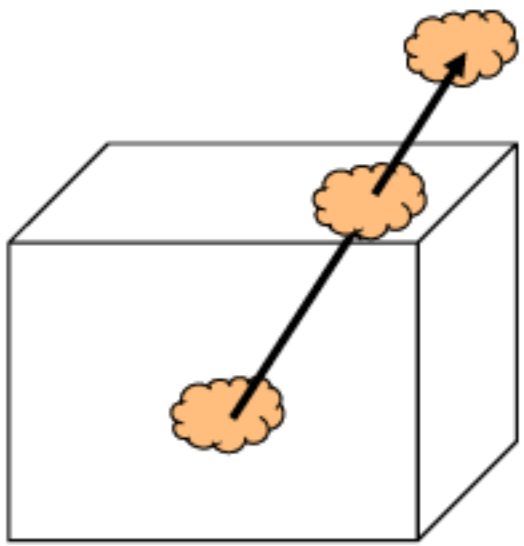
Eric Shaffer

Types of Environment Maps

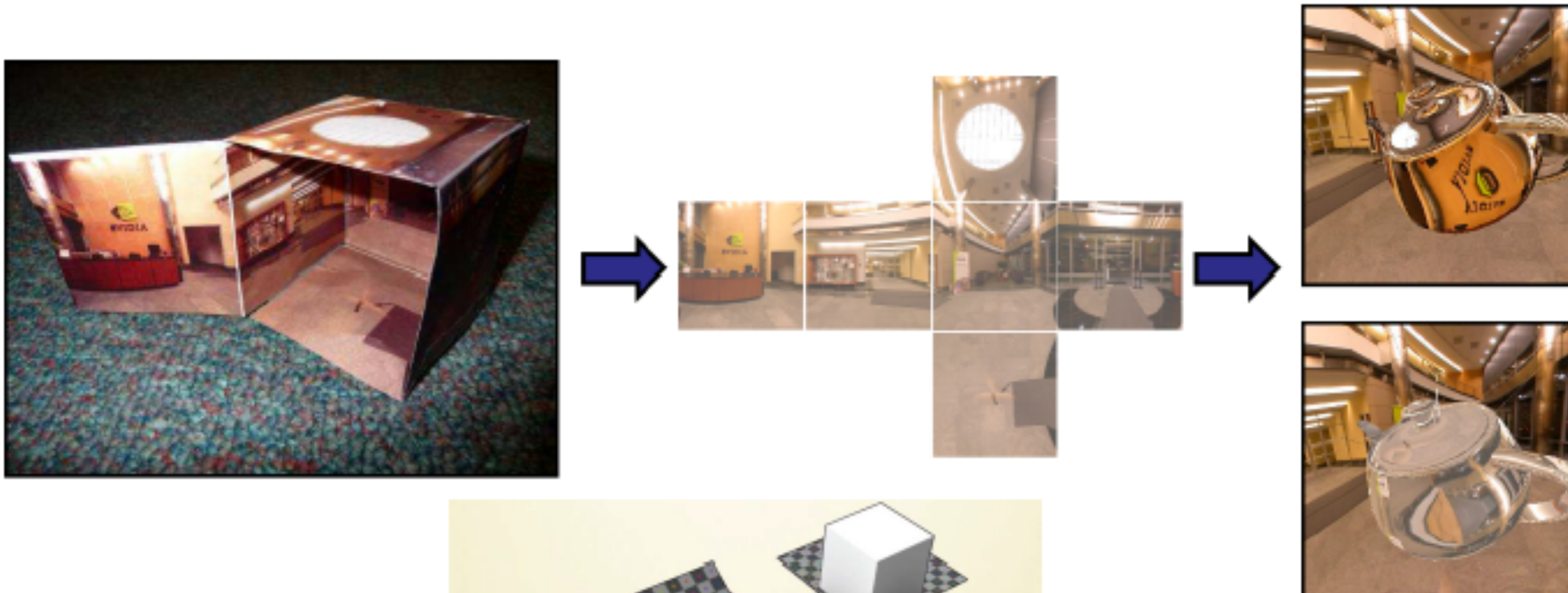
a) Sphere around object (sphere map)

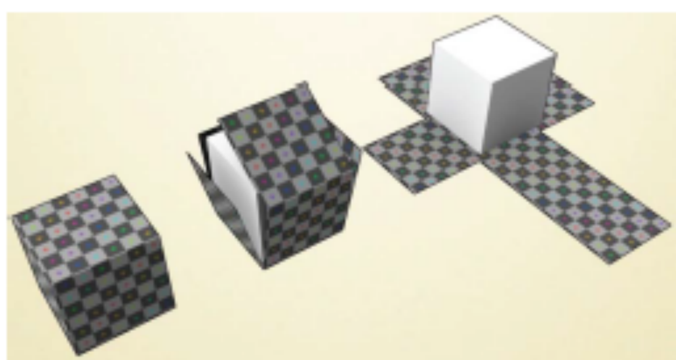


b) Cube around object (cube map)



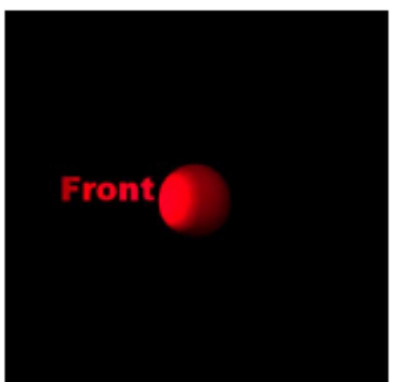
Cube Map

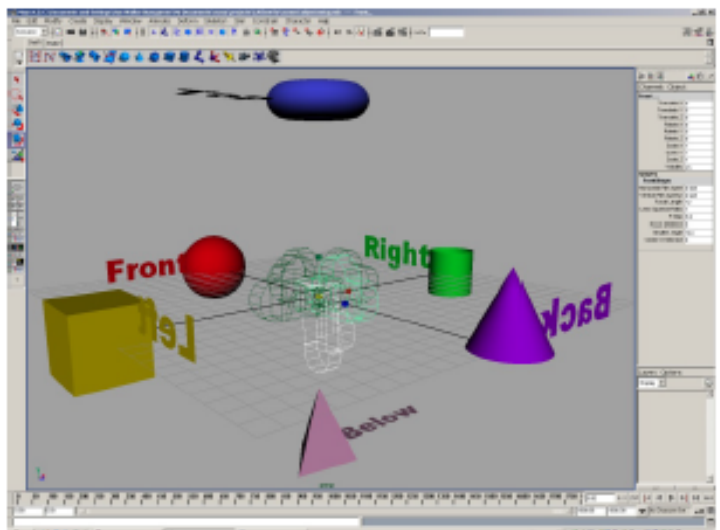


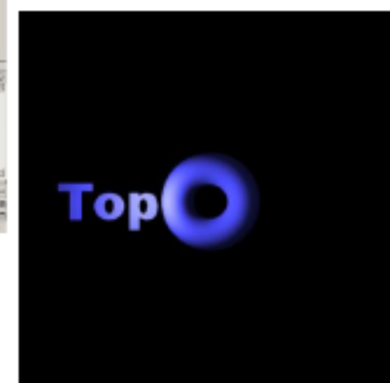


Forming a Cube Map

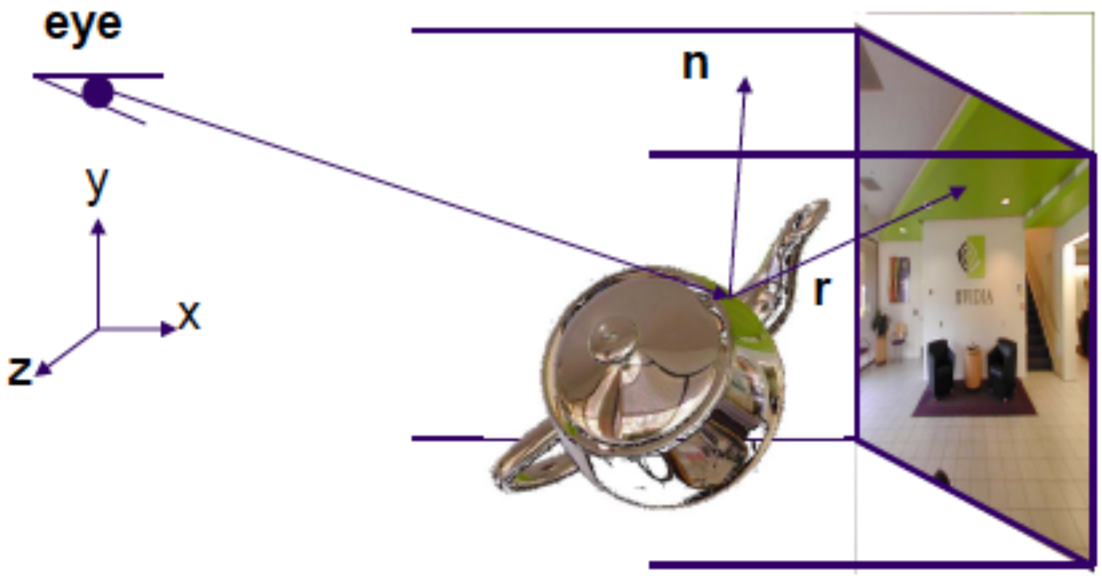
- Use 6 cameras directions from scene center
  - each with a 90 degree angle of view







Reflection Mapping



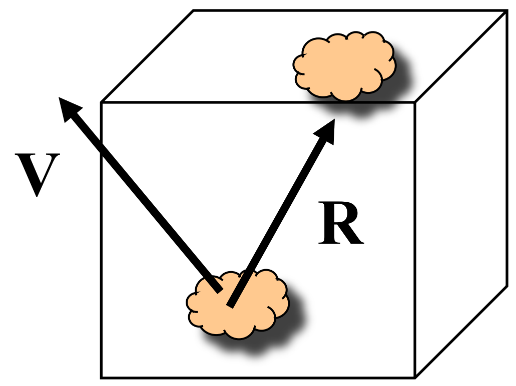
- Need to compute reflection vector, $\mathbf{r}$

Indexing into Cube Map

- Compute $\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{V})\mathbf{N} - \mathbf{V}$
- Object at origin

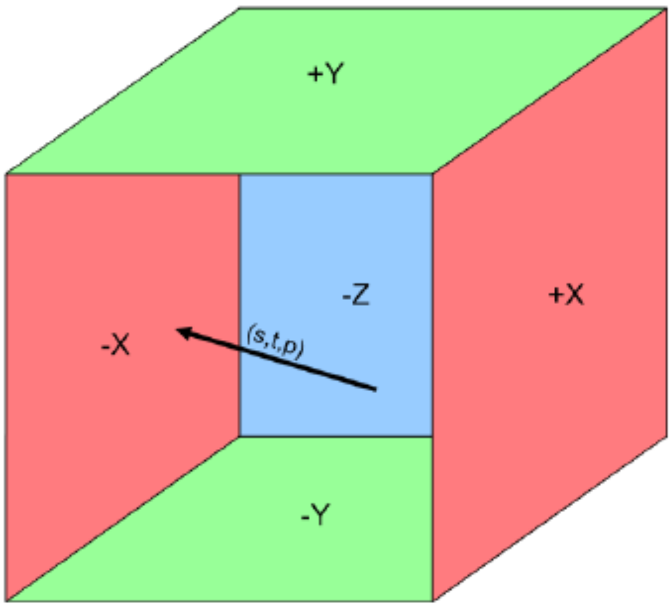
- Use largest magnitude component of R to determine face of cube

- Other two components give texture coordinates



Indexing into a Cube Map

Cube Map Texture Lookup:
Given an (s,t,p) direction vector , what (r,g,b) does that correspond to?



- Let L be the texture coordinate of (s, t, and p) with the largest magnitude
- L determines which of the 6 2D texture “walls” is being hit by the vector (-X in this case)
- The texture coordinates in that texture are the remaining two texture coordinates divided by L: (a/L,b/L)

Built-in GLSL functions

```
vec3 ReflectVector = reflect( vec3 eyeDir, vec3 normal );

vec3 RefractVector = refract( vec3 eyeDir, vec3 normal, float Eta );
```

Example

- **R** = (-4, 3, -1)
- Same as **R** = (-1, 0.75, -0.25)
- Use face x = -1 and y = 0.75, z = -0.25
- Not quite right since cube defined by x, y, z = ± 1 rather than [0, 1] range needed for texture coordinates
- Remap by $s = \frac{1}{2} + \frac{1}{2}y$, $t = \frac{1}{2} + \frac{1}{2}z$
- Hence, s =0.875, t = 0.375

WebGL Implementation

- WebGL supports only cube maps
  - `vec4 texColor = textureCube(mycube, texcoord);`
  - desktop OpenGL also supports sphere maps
- First must form map
  - Use images from a real camera
  - Form images with WebGL
- Texture map it to object

Issues

- Assumes environment is very far from object
  - (equivalent to the difference between near and distant lights)
- Object cannot be concave (no self reflections possible)
- No reflections between objects
- Need a reflection map for each object
- Need a new map if viewer moves

Doing it in WebGL

gl.textureMap2D(
    gl.TEXTURE_CUBE_MAP_POSITIVE_X,
    level, rows, columns, border, gl.RGBA,
    gl.UNSIGNED_BYTE, image1)

- Same for other five images
- Make one texture object out of the six images

Example

- Consider rotating cube that reflects the color of the walls
- Each wall is a solid color (red, green, blue, cyan, magenta, yellow)
  - Each face of room can be a texture of one texel

```
var red = new Uint8Array([255, 0, 0, 255]);
var green = new Uint8Array([0, 255, 0, 255]);
var blue = new Uint8Array([0, 0, 255, 255]);
var cyan = new Uint8Array([0, 255, 255, 255]);
var magenta = new Uint8Array([255, 0, 255, 255]);
var yellow = new Uint8Array([255, 255, 0, 255]);
```

Texture Object

```
cubeMap = gl.createTexture();
gl.bindTexture(gl.TEXTURE_CUBE_MAP, cubeMap);
gl.texImage2D(gl.TEXTURE_CUBE_MAP_POSITIVE_X, 0, gl.RGBA,
    1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, red);
gl.texImage2D(gl.TEXTURE_CUBE_MAP_NEGATIVE_X, 0, gl.RGBA,
    1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, green);
gl.texImage2D(gl.TEXTURE_CUBE_MAP_POSITIVE_Y, 0, gl.RGBA,
    1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, blue);
gl.texImage2D(gl.TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, gl.RGBA,
    1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, cyan);
gl.texImage2D(gl.TEXTURE_CUBE_MAP_POSITIVE_Z, 0, gl.RGBA,
    1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, yellow);
gl.texImage2D(gl.TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, gl.RGBA,
    1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, magenta);
gl.activeTexture( gl.TEXTURE0 );
gl.uniform1i(gl.getUniformLocation(program, "texMap"),0);
```

Vertex Shader

```
varying vec3 R;
attribute vec4 vPosition;
attribute vec4 vNormal;
uniform mat4 modelViewMatrix;
uniform mat4 projectionMatrix;
uniform vec3 theta;
void main(){
    vec3 angles = radians( theta );
    // compute rotation matrices rx, ry, rz here
    mat4 ModelViewMatrix = modelViewMatrix*rz*ry*rx;
   gl_Position = projectionMatrix*ModelViewMatrix*vPosition;
   vec4 eyePos  = ModelViewMatrix*vPosition;
   vec4 N = ModelViewMatrix*vNormal;
    R = reflect(eyePos.xyz, N.xyz);   }
```

Fragment Shader

```
precision mediump float;

varying vec3 R;
uniform samplerCube texMap;

void main()
{
   vec4 texColor = textureCube(texMap, R);
   gl_FragColor = texColor;
}
```

Sphere Mapping

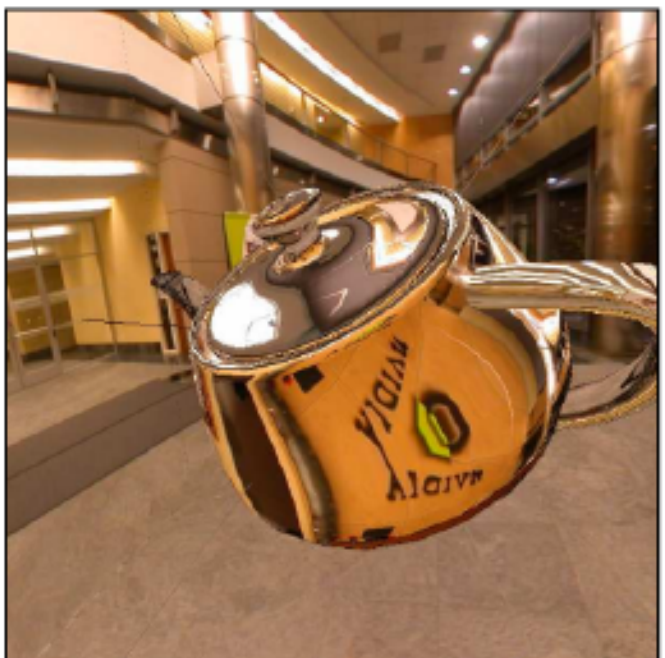
- Original environmental mapping technique proposed by Blinn and Newell based in using lines of longitude and latitude to map parametric variables to texture coordinates
- OpenGL supports sphere mapping which requires a circular texture map equivalent to an image taken with a fisheye lens

Sphere map (texture)

Sphere map applied on torus

Refraction

- Can also use cube map for refraction (transparent)



Reflection

Refraction

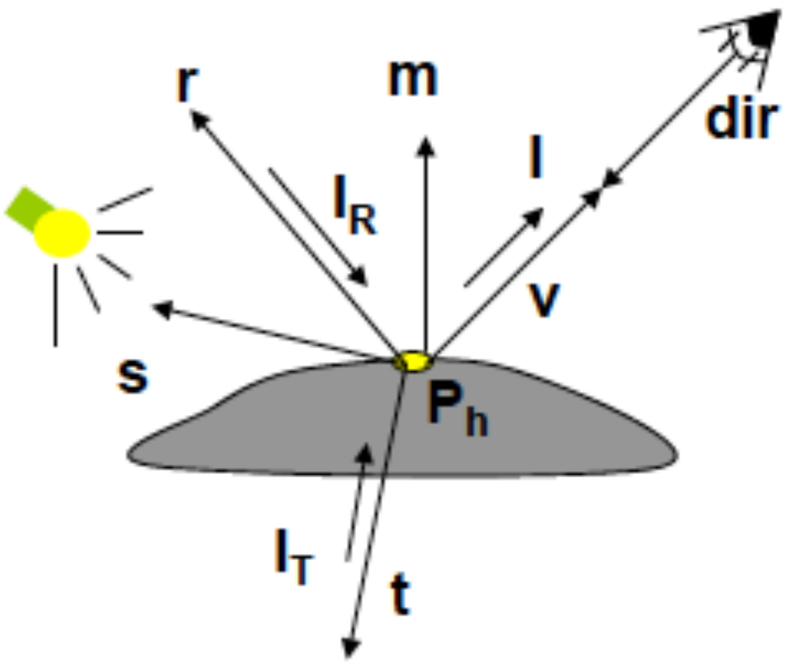
Refraction

Reflection

Refraction

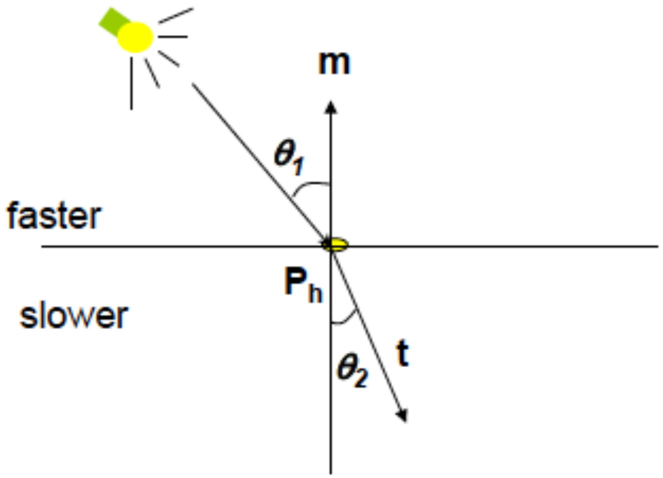
Need to Compute Refraction Vector

$$I = I_{amb} + I_{diff} + I_{spec} + I_{refl} + I_{tran}$$



Snell's Law

- Transmitted direction obeys **Snell's law**
- Snell's law: relationship holds in diagram below



$$\frac{\sin(\theta_2)}{c_2} = \frac{\sin(\theta_1)}{c_1}$$

c_1, c_2 are speeds of light in medium 1 and 2

Medium is Important

- If ray goes from faster to slower medium, ray is bent **towards** normal
- If ray goes from slower to faster medium, ray is bent **away** from normal
- c1/c2 is important. Usually measured for medium-to-vacuum. E.g water to vacuum
- Some measured relative c1/c2 are:
  - Air: 99.97%
  - Glass: 52.2% to 59%
  - Water: 75.19%
  - Sapphire: 56.50%
  - Diamond: 41.33%

Refraction Vertex Shader

```
out vec3 T;
in vec4 vPosition;
in vec4 Normal;
uniform mat4 ModelView;
uniform mat4 Projection;

void main() {
    gl_Position = Projection*ModelView*vPosition;
    vec4 eyePos  = vPosition;                    // calculate view vector V
    vec4 NN = ModelView*Normal;                  // transform normal
    vec3 N =normalize(NN.xyz);                   // normalize normal
    T = refract(eyePos.xyz, N, iorefr);          // calculate refracted vector T
}
```

Was previously R = reflect(eyePos.xyz, N);

Refraction Fragment Shader

```
in vec3 T;
uniform samplerCube RefMap;

void main()
{
    vec4 refractColor = textureCube(RefMap, T);   // look up texture map using T
    refractcolor = mix(refractcolor, WHITE, 0.3);  // mix pure color with 0.3 white

   gl_FragColor = texColor;
}
```