1

Cube Mapping

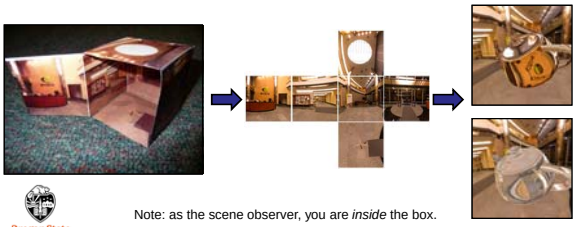
Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu

2

What is Cube Mapping?

Cube Mapping is the process of creating a representation of an object's surrounding environment as a collection of 6 images, grouped together as a single "cube map texture".

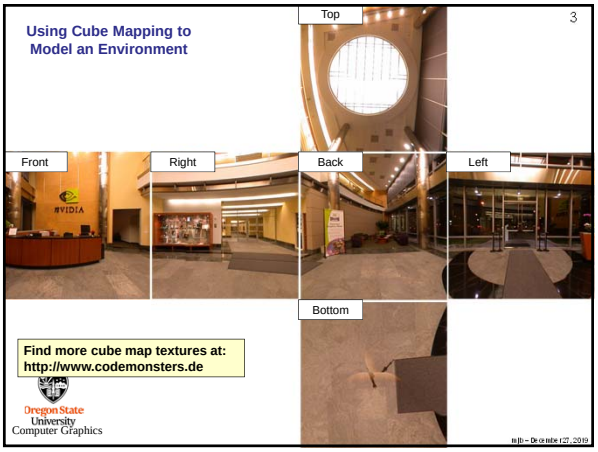
Think of it as a folding box



Note: as the scene observer, you are *inside* the box.

3

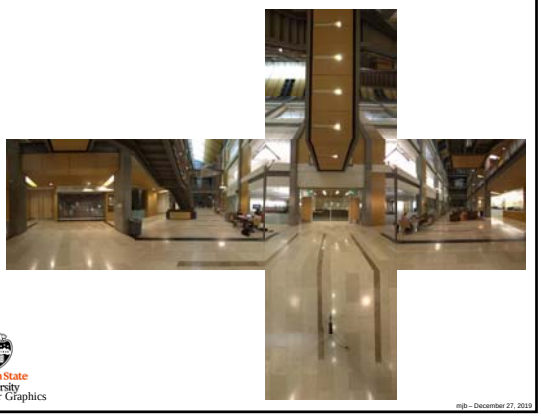
Using Cube Mapping to Model an Environment



Find more cube map textures at: http://www.codemonsters.de

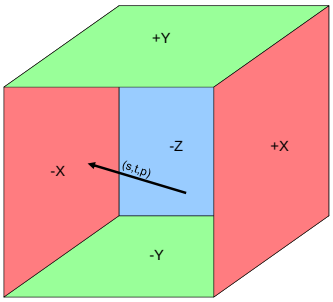
4

The Kelley Engineering Center Atrium



5

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



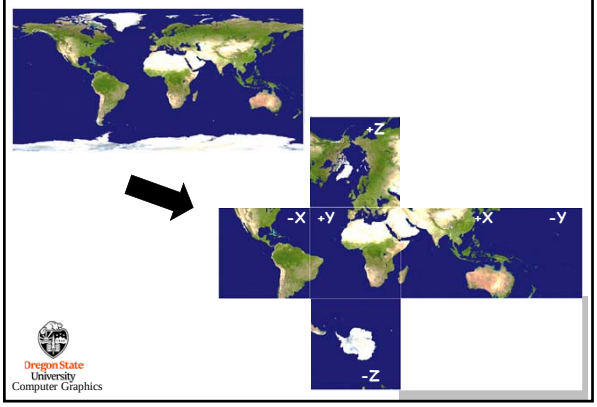
- **Let L be the texture coordinate of (s, t, and p) with the largest magnitude**
- **L determines which of the 6 2D texture "walls" is being hit by the vector (-X in this case)**
- **The texture coordinates in that texture are the remaining two texture coordinates divided by L: (a/L,b/L)**

Built-in GLSL functions

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


6

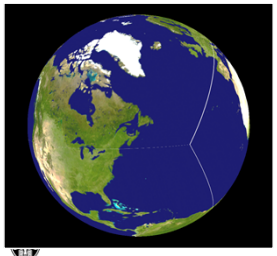
Cube Map of the World

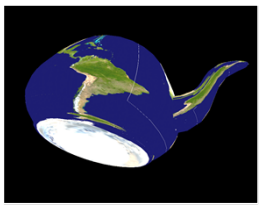


Creating a Globe from the World Cubemap (some shapes map better than others…)

7

Use the normal (n_x,n_y,n_z) as the (s,t,p) for the lookup





(some shapes map better than others…)

Creating a Globe from the World Cubemap

8

```
out  vec3  vNormal;

void
main( )
{
        vNormal = normalize( gl_Normal );
        gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

```
uniform samplerCube uTexUnit;
in  vec3   vNormal;

void
main( )
{
        vec4 newcolor = textureCube( uTexUnit, normalize(vNormal) );
        gl_FragColor = vec4( newcolor.rgb, 1. );
}
```


Using the Cube Map for Reflection

9

Using the Cube Map for Reflection

10

```
out vec3                vReflectVector;

void
main( )
{
        vec3 ECposition = ( gl_ModelViewMatrix * gl_Vertex ).xyz;
        vec3 eyeDir = ECposition – vec3(0.,0.,0.);          // vector from eye to pt
        vec3 normal = normalize( gl_NormalMatrix * gl_Normal );
        vReflectVector = reflect( eyeDir, normal );
        gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

```
in vec3                 vReflectVector;
uniform samplerCube   uReflectUnit;

void
main( )
{
        vec4 newcolor = textureCube( uReflectUnit, vReflectVector );
        gl_FragColor = vec4( newcolor.rgb, 1. )
}
```


Using the Cube Map for Refraction

11

Using the Cube Map for Refraction

12

```
out vec3                vRefractVector;
out vec3                vReflectVector;
uniform float           uEta;

void main( )
{
        vec3 ECposition = vec3( gl_ModelViewMatrix * gl_Vertex );
        vec3 eyeDir = normalize( ECposition – vec3(0.,0.,0. ) );     // vector from eye to pt
        vec3 normal = normalize( gl_NormalMatrix * gl_Normal );
        vRefractVector = refract( eyeDir, normal, uEta );
        vReflectVector = reflect(  eyeDir, normal );
        gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

```
in vec3                 vReflectVector;
in vec3                 vRefractVector;
uniform float           uMix;
uniform samplerCube   uReflectUnit;
uniform samplerCube   uRefractUnit;
const vec4 WHITE = vec4( 1.,1.,1.,1. );

void main( )
{
        vec4 refractcolor = textureCube( uRefractUnit, vRefractVector );
        vec4 reflectcolor = textureCube( uReflectUnit,  vReflectVector );
        refractcolor = mix( refractcolor, WHITE, .40 );
        gl_FragColor = vec4( mix( refractcolor, reflectcolor, uMix ).rgb, 1. );
}
```

13

Oregon State
University
Computer Graphics

14

Oregon State
University
Computer Graphics

15

Oregon State
University
Computer Graphics