Cube Mapping

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License
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.
Using Cube Mapping to Model an Environment

http://codemonsters.de
The Kelley Engineering Center Atrium
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

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

Oregon State University
Computer Graphics
Creating a Globe from the World Cubemap
(some shapes map better than others…)

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

```cpp
out vec3 vNormal;

void main( )
{
    vNormal = normalize( aNormal );
    gl_Position = uModelViewProjectionMatrix * aVertex;
}
```

```cpp
uniform samplerCube uTexUnit;
in vec3 vNormal;
out vec4 fFragColor;

void main( )
{
    vec4 newcolor = textureCube( uTexUnit, normalize(vNormal) );
fFragColor = newcolor;
}
```
Using the Cube Map for Reflection
Using the Cube Map for Reflection

```cpp
out vec3 vReflectVector;

void main( )
{
    vec3 ECposition = ( uModelViewMatrix * aVertex ).xyz;
    vec3 eyeDir = ECposition - vec3(0.,0.,0.); // vector from eye to pt
    vec3 normal = normalize( uNormalMatrix * aNormal );
    vReflectVector = reflect( eyeDir, normal );
    gl_Position = uModelViewProjectionMatrix * aVertex;
}

in vec3 vReflectVector;
uniform samplerCube uReflectUnit;
out vec4 fFragColor;

void main( )
{
    vec4 newcolor = textureCube(u ReflectUnit, vReflectVector );
    fFragColor = newcolor;
}
```
Using the Cube Map for Refraction
Using the Cube Map for Refraction

```glsl
vec3 ECposition = vec3( uModelViewMatrix * aVertex );
vec3 eyeDir = normalize( ECposition – vec3(0.,0.,0.) ); // vector from eye to pt
vec3 normal = normalize( uNormalMatrix * aNormal );
vRefractVector = refract( eyeDir, normal, uEta );
vReflectVector = reflect( eyeDir, normal );
gl_Position = uModelViewProjectionMatrix * aVertex;
```

```glsl
vec4 refractcolor = textureCube( uRefractUnit, vRefractVector );
vec4 reflectcolor = textureCube( uReflectUnit, vReflectVector );
refractcolor = mix( refractcolor, WHITE, .40 );
fFragColor = mix( refractcolor, reflectcolor, uMix );
```