Cube Mapping

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

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 $L$: \((s/L, b/L)\)

Built-in GLSL functions

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

Cube Map of the World
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

Creating a Globe from the World Cubemap

\[
\text{out vec3 vNormal;}
\]
\[
\text{void main( )}
\]
\[
\begin{array}{l}
\text{vec4 newcolor = textureCube( uTexUnit, normalize(vNormal) );}
\text{gl_FragColor = vec4( newcolor.rgb, 1. );}
\end{array}
\]

Using the Cube Map for Reflection

\[
\text{out vec3 vReflectVector;}
\]
\[
\text{void main( )}
\]
\[
\begin{array}{l}
\text{vec3 ECposition = ( gl_ModelViewMatrix * gl_Vertex ).xyz;}
\text{vec3 eyeDir = ECposition – vec3(0.,0.,0.); // vector from eye to pt}
\text{vec3 normal = normalize( gl_NormalMatrix * gl_Normal );}
\text{vReflectVector = reflect( eyeDir, normal );}
\text{gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;}
\end{array}
\]

Using the Cube Map for Refraction

\[
\text{out vec3 vRefractVector;}
\text{out vec3 vReflectVector;}
\text{uniform float uEta;}
\]
\[
\text{void main( )}
\]
\[
\begin{array}{l}
\text{vec3 ECposition = vec3( gl_ModelViewMatrix * gl_Vertex );}
\text{vec3 eyeDir = normalize( vec3(0.,0.,0.) ); // vector from eye to pt}
\text{vec3 normal = normalize( gl_NormalMatrix * gl_Normal );}
\text{vRefractVector = refract( eyeDir, normal, uEta );}
\text{vReflectVector = reflect( eyeDir, normal );}
\text{gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;}
\end{array}
\]