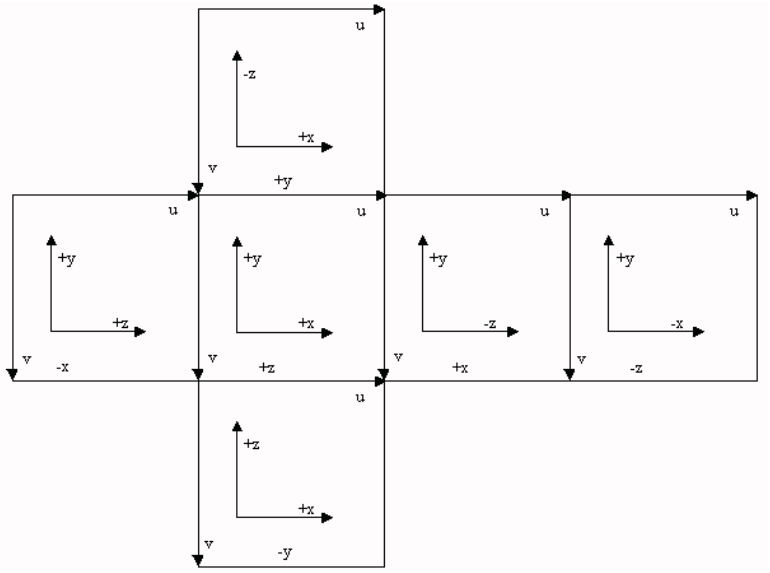
CS 537 – Interactive Computer Graphics

Lecture 7 – Part 2

Cube Mapping and Skyboxes

Cube Mapping

- We already saw one way to map a cube
  - For each vertex of each face, specify the (s, t) coordinates for that vertex and use the same texture for each face
- We could extend this to have a different image on each face
- If we unfold an environment to become 6 rectangles, we can then map a cube to resemble an environment



OpenGL Cube Mapping

- OpenGL has a built-in way to handle these “cube maps”
  - Make a texture an active cube map texture
  - Set the properties for the cube map

```
//SKYBOX
//Skybox textures
//Load Skybox Images. 6 images to represent the 6 angles of view. Inside it's own structured Cubemap

glBindTexture(GL_TEXTURE_CUBE_MAP, textures[2]);
glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);
```

OpenGL Cube Mapping

- Link the texel data to each face the bound texture

```
skyTop = glmReadPPM("skybox\\sky-top.ppm", &TextureSize, &TextureSize);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, 0, GL_RGB,  TextureSize, TextureSize, 0, GL_RGB, GL_UNSIGNED_BYTE,skyTop);
skyBottom = glmReadPPM("skybox\\sky-bottom.ppm", &TextureSize, &TextureSize);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, GL_RGB,  TextureSize, TextureSize, 0, GL_RGB, GL_UNSIGNED_BYTE,skyBottom);
skyRight = glmReadPPM("skybox\\sky-right.ppm", &TextureSize, &TextureSize);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X, 0, GL_RGB,  TextureSize, TextureSize, 0, GL_RGB, GL_UNSIGNED_BYTE,skyRight);
skyLeft = glmReadPPM("skybox\\sky-left.ppm", &TextureSize, &TextureSize);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, 0, GL_RGB,  TextureSize, TextureSize, 0, GL_RGB, GL_UNSIGNED_BYTE,skyLeft);
skyFront = glmReadPPM("skybox\\sky-front.ppm", &TextureSize, &TextureSize);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z, 0, GL_RGB,  TextureSize, TextureSize, 0, GL_RGB, GL_UNSIGNED_BYTE,skyFront);
skyBack = glmReadPPM("skybox\\sky-back.ppm", &TextureSize, &TextureSize);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, GL_RGB,  TextureSize, TextureSize, 0, GL_RGB, GL_UNSIGNED_BYTE,skyBack);
```

- Then you can assign each vertex of the cube to a point in (u, v, w) space where

$$-1 \leq u, v, w \leq 1$$

OpenGL Cube Mapping

- The only difference in the shaders is that the vertex shader will now have an *in vec3* attribute for the tex coordinate
- The fragment shader will now have a *uniform samplerCube* variable (instead of a sampler2D)

```
#version 150
in  vec3 texCoord;
uniform samplerCube cubeMap;
out vec4 fColor;
void main() {

    fColor = texture(cubeMap, texCoord);
    fColor.a = 1.0;

}
```

Skyboxes

- A common use for cube-maps is making a *sky-box*.
- The idea is that an entire environment/scene is wrapped around a large cube with the viewer at the center
  - This cube can move as the user moves so that he/she can never reach the horizon

Skybox: General Approach

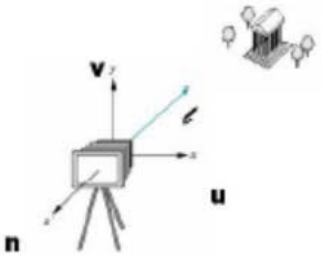
Just like with cube-mapping

1. Load 6 textures, one for each side of our cube
   - `GLubyte *front, *right, *back, *left, *right, *top, *bottom;`
2. Set up the texture object
   - Make a texture object active (bind it)
   - Add links to the texture images to the active texture
     - `glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, …)`
   - Add any parameters
3. Build the cube
4. When rendering select a texture unit, enable the desired texture type, and make active a texture for that unit

Skybox

- The only difference between our previous cube map example and a skybox is that we want a skybox centered at the camera and it should not occlude any other objects.

Skybox



- Here's how we can do this:
  1. When specifying the **vertex locations** for the cube, just do them all in the range of $[-1,1]$
     - That way we don't even need to provide texture mappings!
  2. When we render the skybox set its model-view (`model-view=camera_matrix*model_matrix`) to
     - `vec4 skyboxEye = vec4(0,0,0,1);`
     - `mat4 model_view = LookAt(skyboxEye, skyboxEye – n, v);`
     - The result is the box centered at the camera, oriented in the *at* direction and with the correct *up* direction

Skybox

3. We must also **disable depth test and depth mask** before rendering and re-enable them after
    - This way the skybox won't occlude anything
    - Therefore, render the skybox **first** then render the rest of the scene
4. Now in the vertex shader we just extract the xyz coordinates as the texel coordinates
    - `texCoord = vPosition.xyz;`

- Two more "gotchas"
    - We should render the skybox first so it is not occluded by other objects
    - When setting up the projection matrix make sure that near plane is in front of the skybox
        - Otherwise you won't see the skybox at all

Skybox

```
#version 150

in vec4 vPosition;
uniform   mat4 ModelView;
uniform   mat4 Projection;

out vec3 texCoord;

void main() {

      exCoord     = normalize(vPosition.xyz);


      l_Position = Projection*ModelView*vPosition;
```

```
#version 150
in  vec3 texCoord;
uniform samplerCube cubeMap;
out vec4 fColor;
void main() {

   fColor = texture(cubeMap, texCoord);
   fColor.a = 1.0;

}
```

```
void MySkyBox::draw(Camera cam, vector<Light> lights){

    glBindVertexArray(VAO);

    glUseProgram(program);

    GLuint model_loc = glGetUniformLocation(program, "ModelView");
    glUniformMatrix4fv(model_loc, 1, GL_TRUE, cam.getViewMatrix());

    GLuint proj_loc = glGetUniformLocation(program, "Projection");
    glUniformMatrix4fv(proj_loc, 1, GL_TRUE, cam.getProjectionMatrix());

    glEnable(GL_TEXTURE_CUBE_MAP);
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_CUBE_MAP, texture);
    GLuint mapPos = glGetUniformLocation( program, "cubeMap" );
    glUniform1i(mapPos, 0);

    glDisable(GL_DEPTH_TEST);
    glDepthMask(GL_FALSE);
    glDrawArrays(GL_TRIANGLES, 0, 6*3*2);
    glEnable(GL_DEPTH_TEST);
    glDepthMask(GL_TRUE);

}
```

```
//camera location
vec4 eye = vec4(0.0, 0.0, 2.0, 1.0);
vec4 at = vec4(0.0, 0.0, 0.0, 1.0);
vec4 up = vec4(0.0, 1.0, 0.0, 0.0);

camera.positionCamera(eye, normalize(eye - at), up, vec4(1, 0, 0, 0));
skyboxCamera.positionCamera(vec4(0, 0, 0, 1), normalize(eye - at), up,vec4(1, 0, 0, 0));
```

Should the skybox have it's own camera object?

Skybox