Game Graphics & Real-time Rendering

CMPM 163, W2018

Prof. Angus Forbes (instructor)
angus@ucsc.edu

Lucas Ferreira (TA)
lferreira@ucsc.edu

creativecoding.soe.ucsc.edu/courses/cmpm163
github.com/CreativeCodingLab

Last class

- How to write an image processing shader
- How to use Frame Buffer Objects to render to an off-screen texture (using Three.js' WebGLRenderTarget)
- How to swap textures using a "pingpong" strategy to perform computation using data stored in textures
- Using textures as arrays of data
- https://creativecoding.soe.ucsc.edu/courses/cmpm163/code/week2_codeExamples.zip

This week

- How to load in an object created with Blender and apply a texture to it
- Difference between ShaderMaterial and RawShaderMaterial
- CubeMap textures
- Create a “skybox”
- Environmental mapping onto reflective surfaces
- https://creativecoding.soe.ucsc.edu/courses/cmpm163/code/week3_codeExamples.zip

This week

- Load an .obj file from Google Poly
- Load a cube map texture from Humus.name
- Environmental mapping the skybox onto your Google Poly object
- https://creativecoding.soe.ucsc.edu/courses/cmpm163/code/week3_codeExamples.zip

Loading a Blender object in Three.js

```
//Re-use method from last week to create a DataTexture (or could instead load in an image).
var texture1 = createDataTexture();

//Use the JSONLoader object to load in an object created with Blender
//In this example, we are ignoring Blender’s material, only using the position and uv coords.
var loader = new THREE.JSONLoader();
loader.load( 'horse.js', processBlenderObject );

//the function that’s called to process the BlenderObject so that it can be used in Three.js
function processBlenderObject (geometry, materials) {
    var bufferGeometry = new THREE.BufferGeometry().fromGeometry( geometry );

    var material = new THREE.RawShaderMaterial( {
        uniforms: { t1: { type: "t", value: texture1  } },
        vertexShader: vs,
        fragmentShader: fs,
    } );
    scene.add(new THREE.Mesh( bufferGeometry, material ) );
}
```

Geometry vs. BufferGeometry

- BufferGeometry stores all data per vertex in a BufferAttribute array. You can define your own attribute arrays, or use ones that are automatically available to you when you load in a Blender object (or both). In this code example, the model we loaded ("horse.js") contains vertices and texture coordinates.

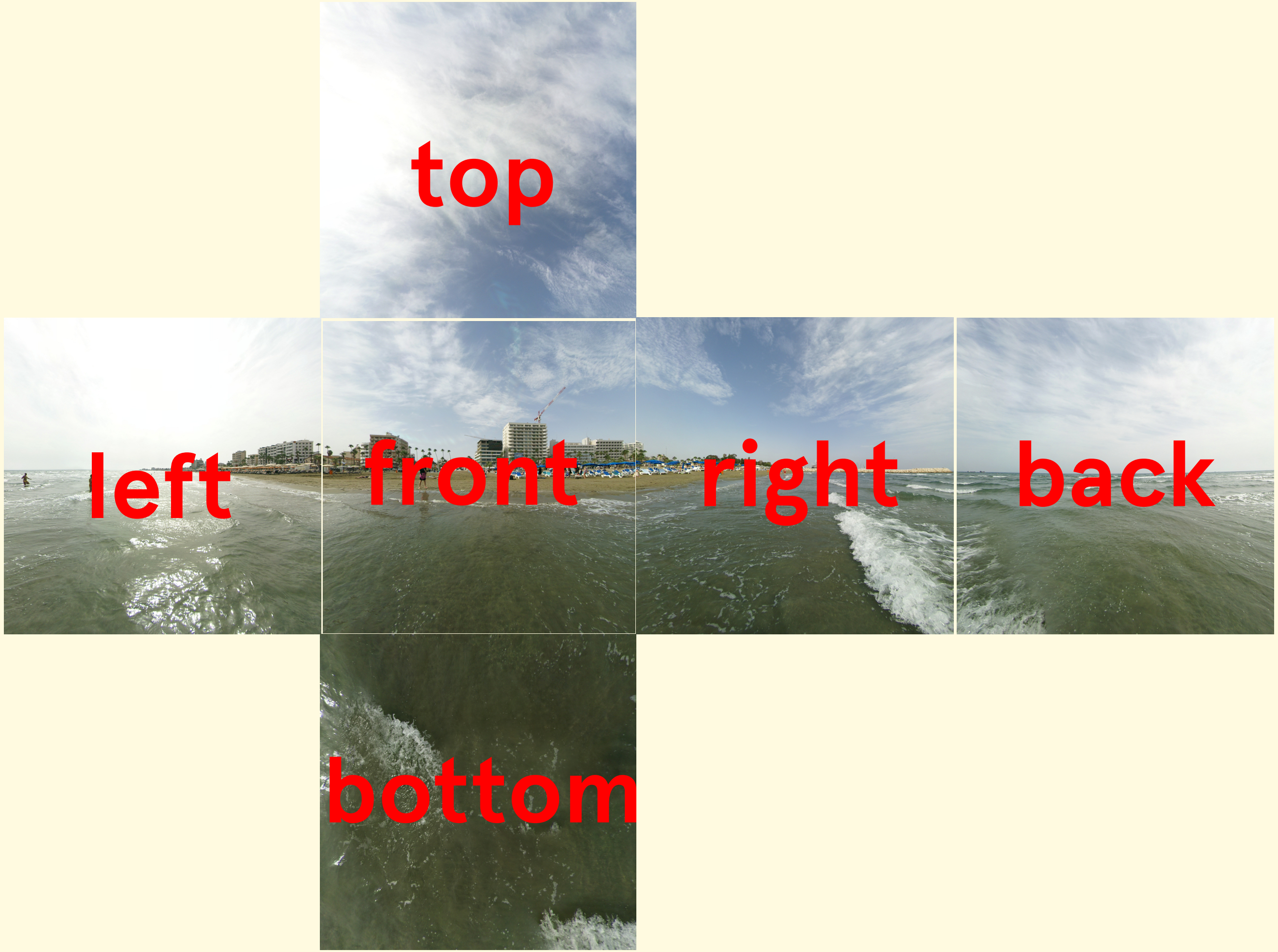
- You can switch between Geometry and BufferGeometry using these convenience methods:

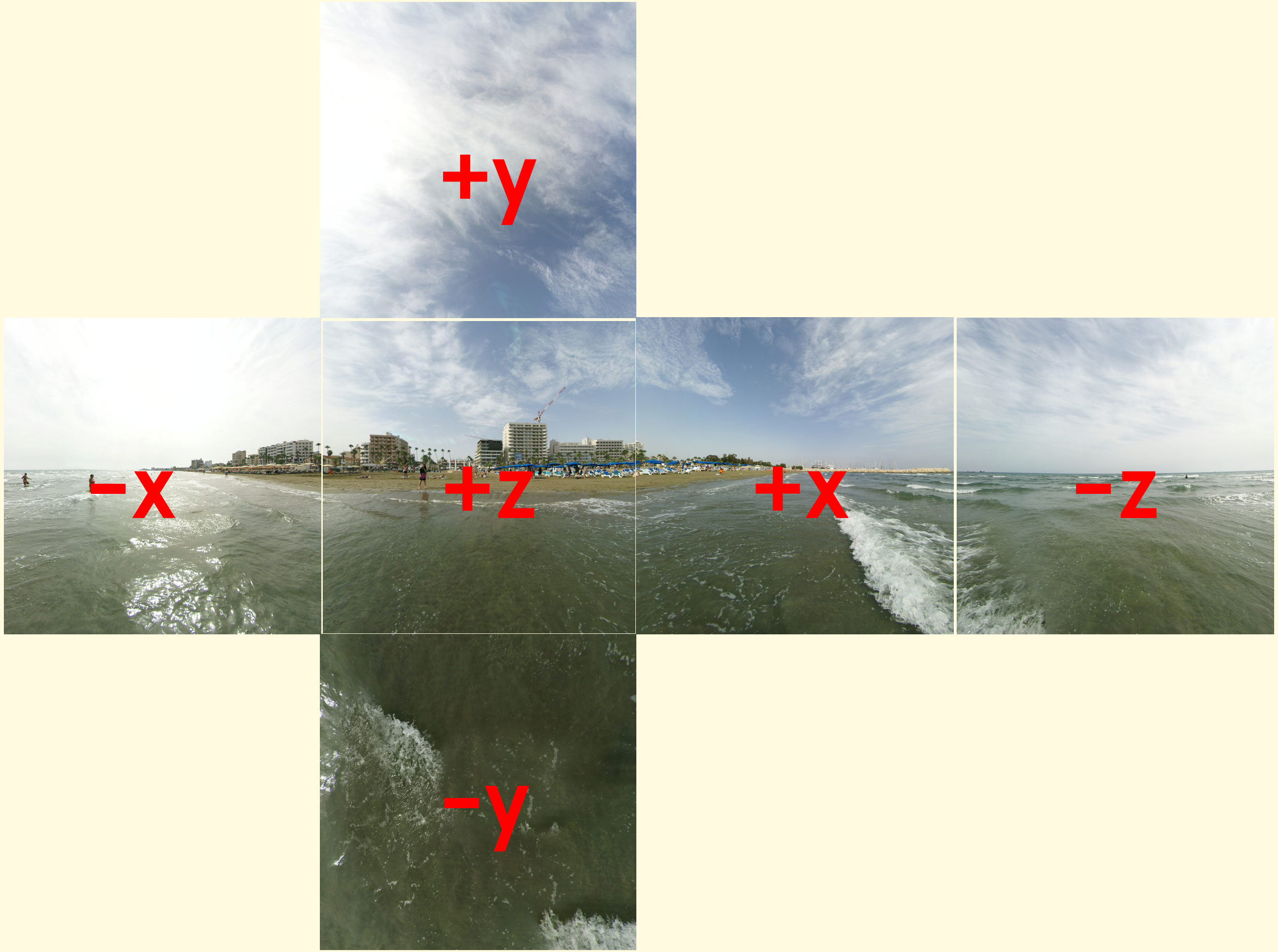
```
var bufferGeometry = new THREE.BufferGeometry().fromGeometry( geometry );
var geometry = new THREE.Geometry().fromBufferGeometry( bufferGeometry );
```

- BufferGeometry is sent to the GPU more quickly
- BufferGeometry lets you define custom attribute data
- If your GLSL shader is just using common attributes (position, uv, normal), then you can use Geometry

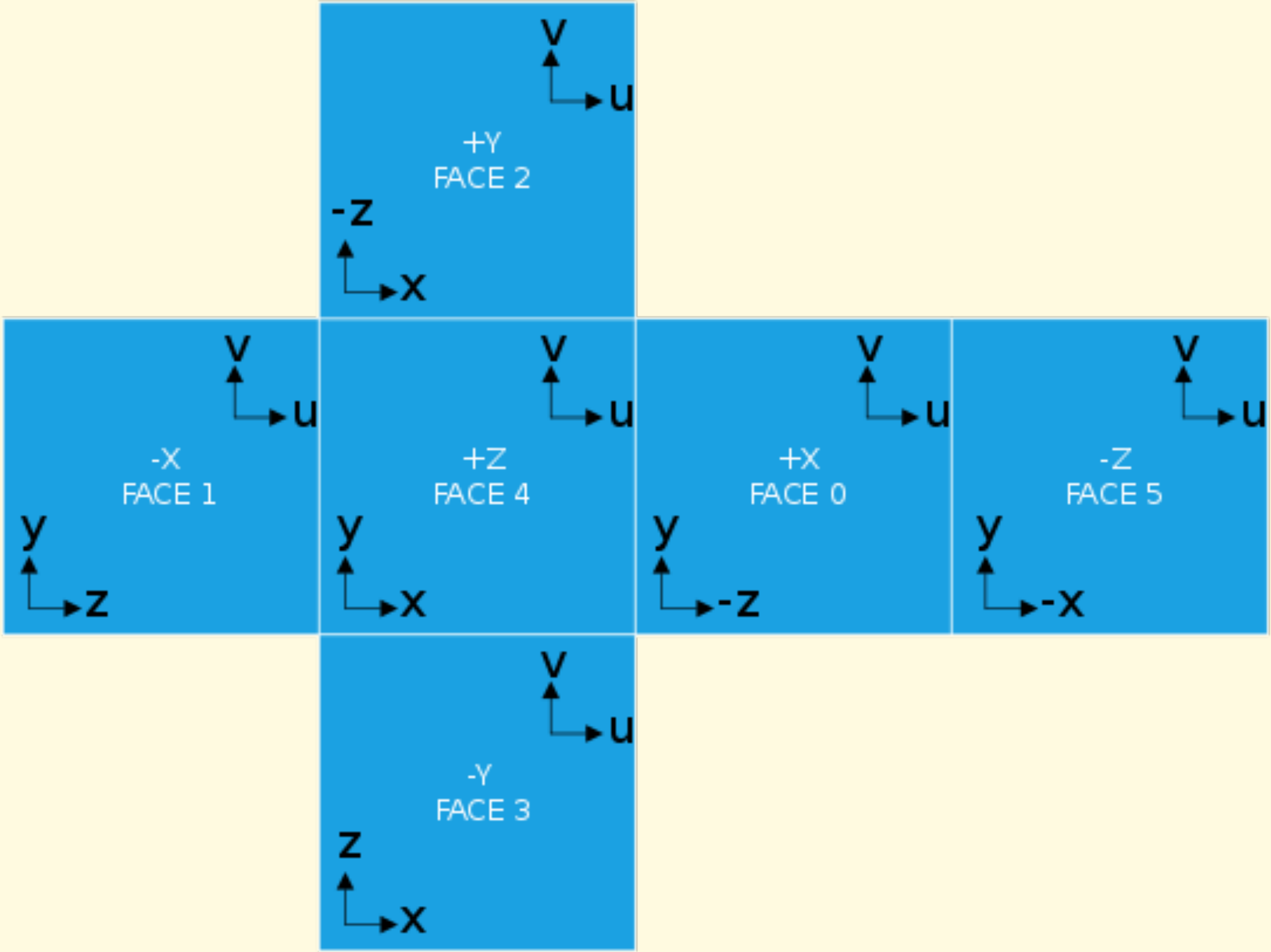
ShaderMaterial vs. RawShaderMaterial

- Both allow you to use a custom GLSL shader when rendering geometry.
- ShaderMaterial tries to simplify your life by always automatically including some common attributes and uniforms at the top of your GLSL shaders. (I always forget what some of them are though! – see, e.g., the answer to https://stackoverflow.com/questions/37342114/three-js-shadermaterial-lighting-not-working)
- RawShaderMaterial forces you to define all attributes and uniforms manually in your GLSL shader.
- If you are just using common attributes (position, uv, normal), and uniforms (transform matrices, light), and you can remember how Three.js represents them, then use ShaderMaterial.
- If you need more flexibility in regards to the data you need to manipulate in the GLSL sahder, then use RawShaderMaterial.

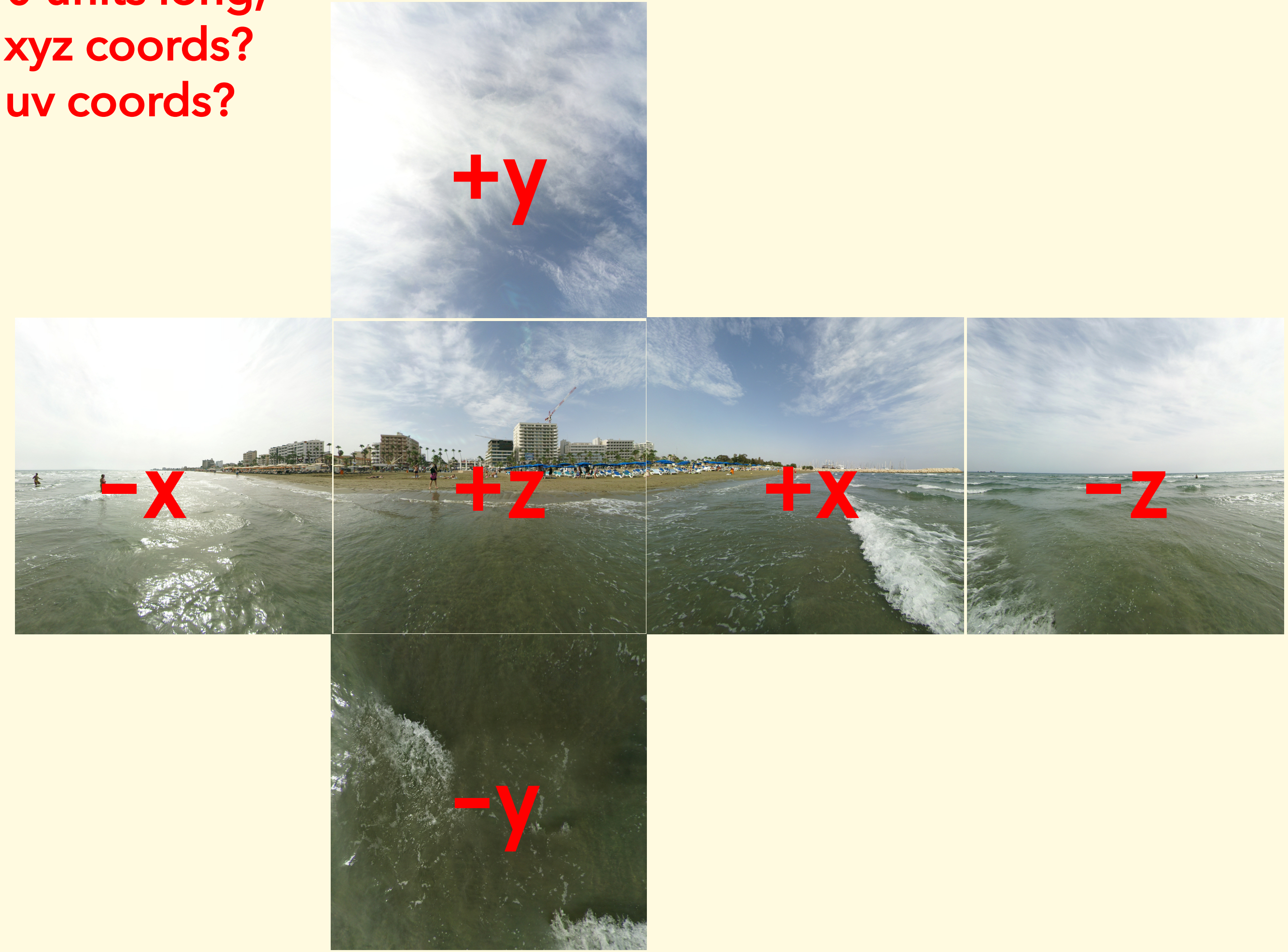

top
left
front
right
back
bottom



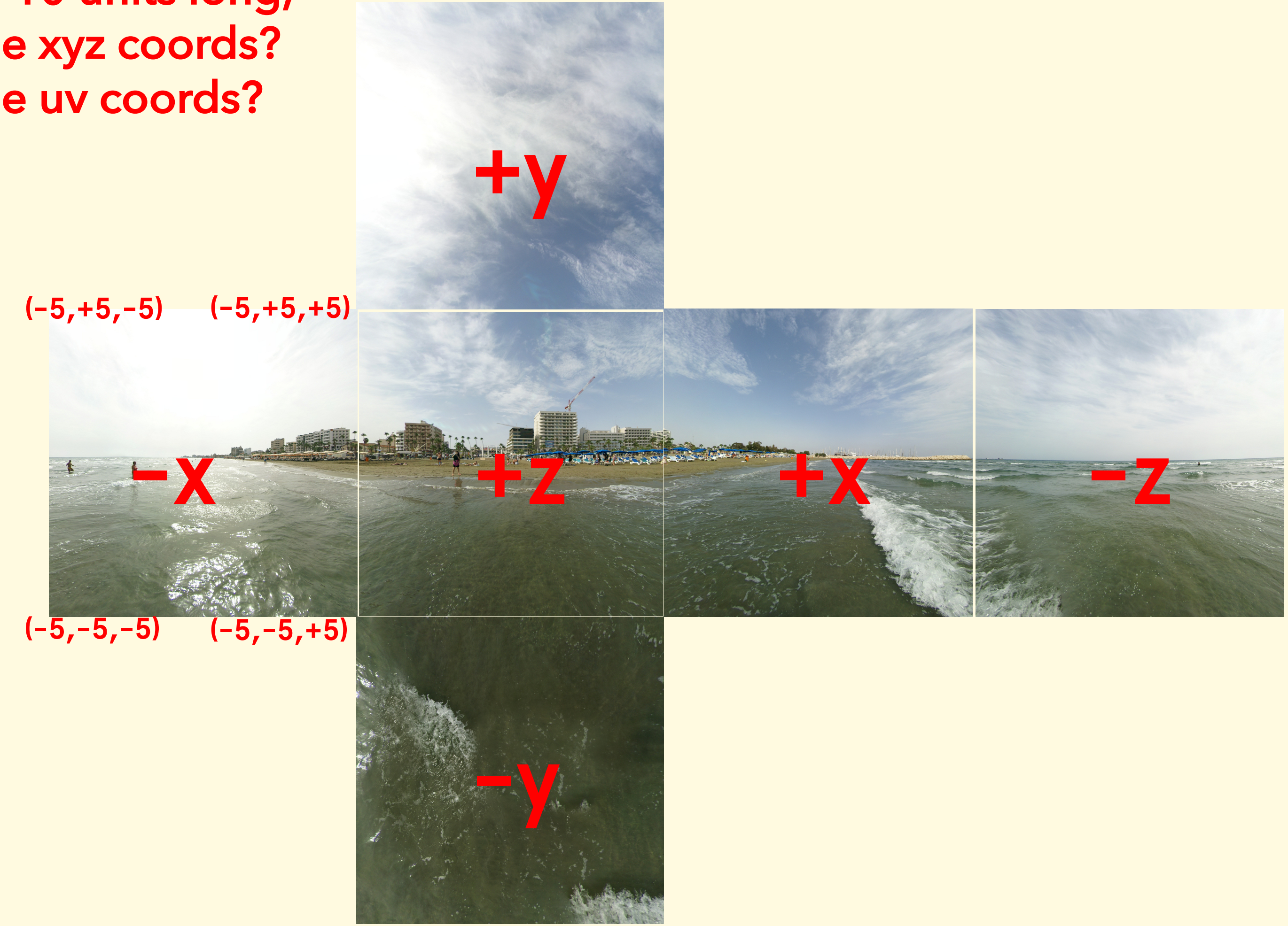
+y
−x
+z
+x
−z
−y


v
u
+Y
FACE 2
-z
x
v
u
-X
FACE 1
y
z
v
u
+Z
FACE 4
y
x
v
u
+X
FACE 0
y
-z
v
u
-Z
FACE 5
y
-x
v
u
-Y
FACE 3
z
x

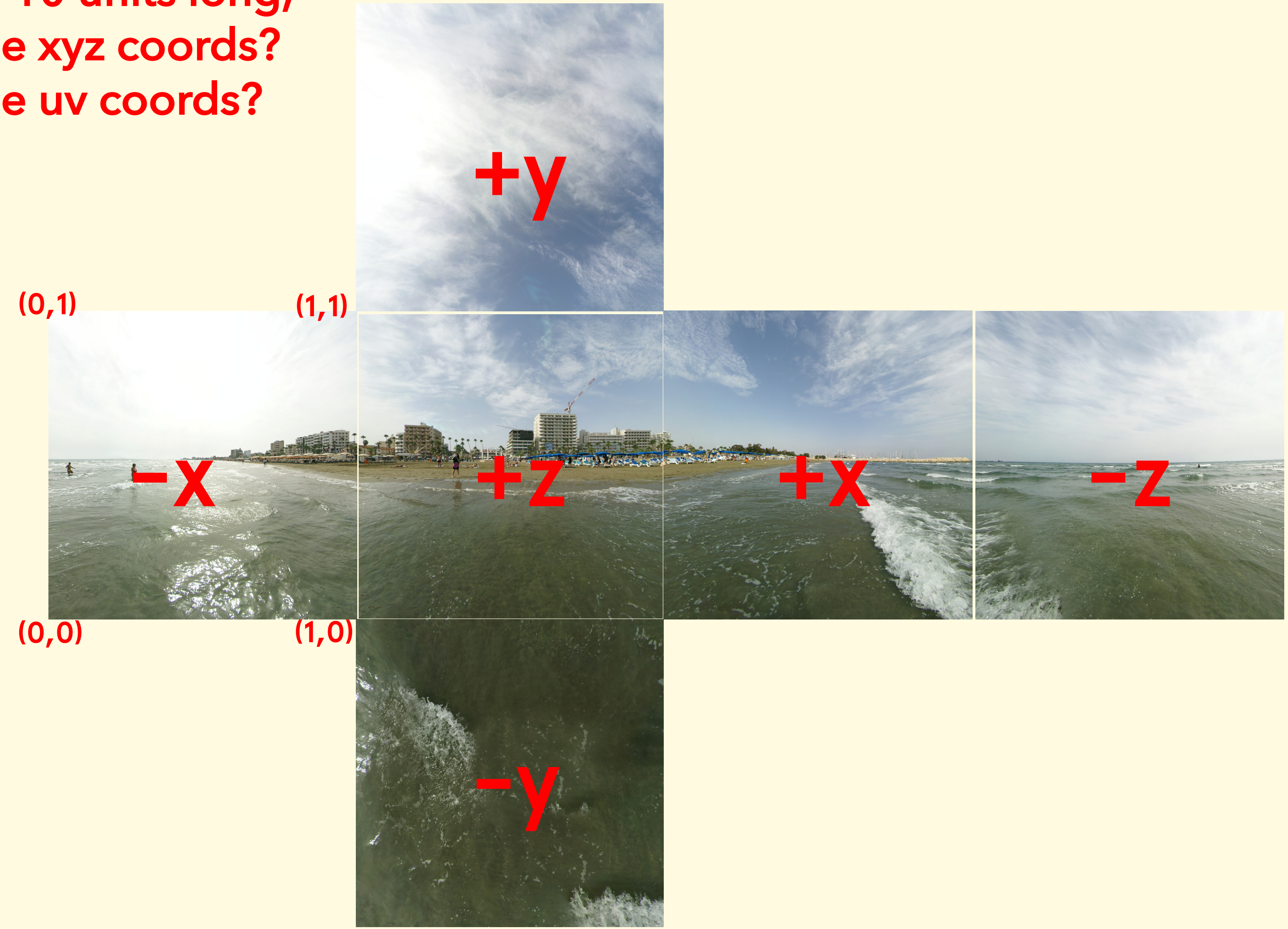
If the box is 10 units long,
What are the xyz coords?
What are the uv coords?



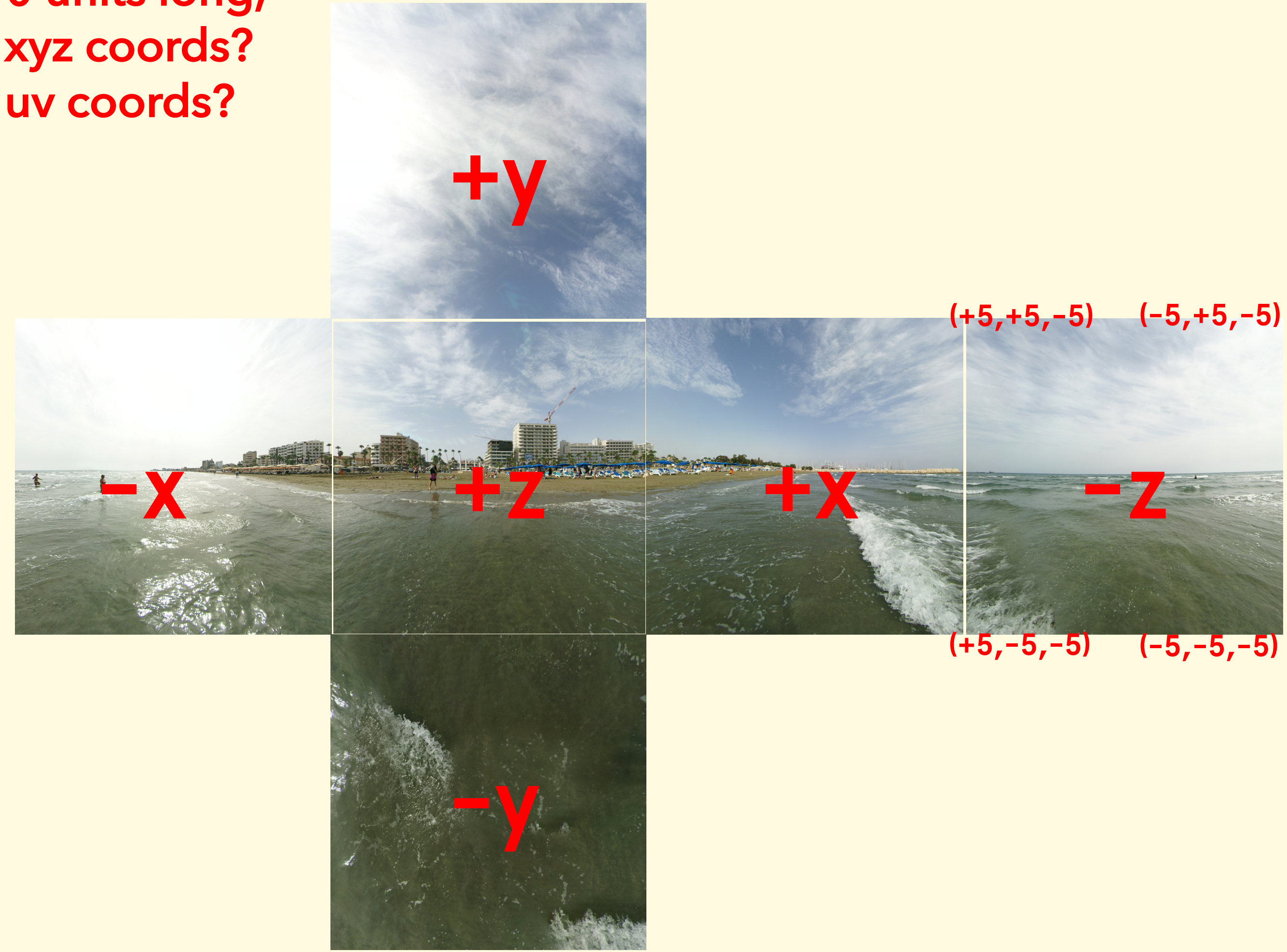
If the box is 10 units long,
What are the xyz coords?
What are the uv coords?



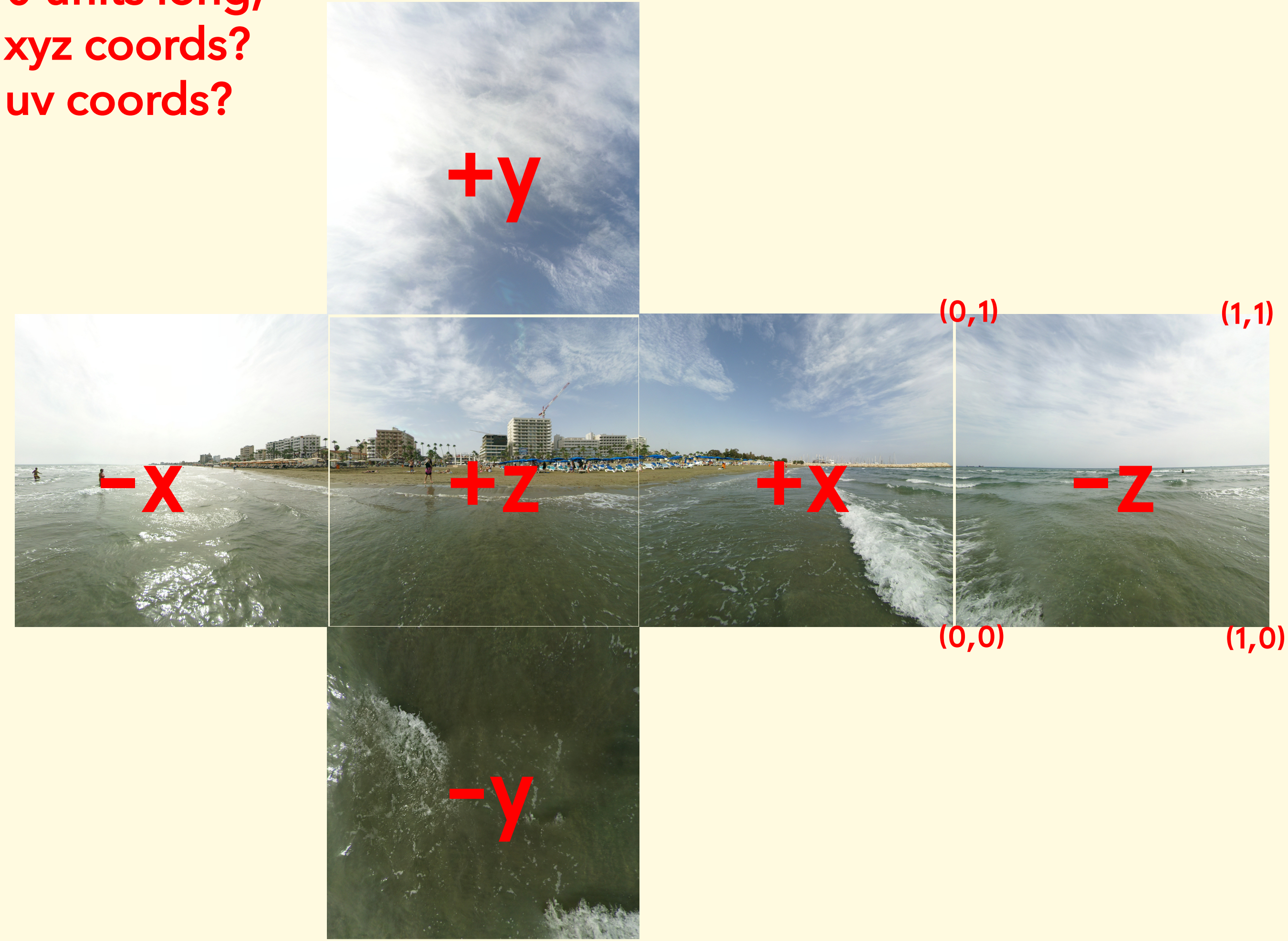
If the box is 10 units long,
What are the xyz coords?
What are the uv coords?



If the box is 10 units long,
What are the xyz coords?
What are the uv coords?



If the box is 10 units long,
What are the xyz coords?
What are the uv coords?



CubeMap textures

```
var cubeMap = new THREE.CubeTextureLoader().load( [
    'posx.jpg', 'negx.jpg',
    'posy.jpg', 'negy.jpg',
    'posz.jpg', 'negz.jpg'
] );

//GLSL Fragment shader
precision mediump float;
uniform samplerCube cubeMap;
varying vec3 vWorldPosition;

void main() {
    gl_FragColor = textureCube(cubeMap, vec3(  vWorldPosition ) );
}
```

.OBJ files

- Can be created and exported in Blender, Maya, 3DStudio Max, etc.
- Google Poly has a library of 3D objects that you can use
- Contains:
  - Vertices
  - Normals
  - Texture coordinates
- Also contains a series of Faces
  - Each face is either a triangle or a rectangle (almost always triangles)
  - Each point on the face refers to one vertex (v), one normal (vn), and one texture coordinate (vt).

.OBJ files

```
<script src="js/OBJLoader.js"></script> //need to include this!

var loader = new THREE.OBJLoader( );

loader.load( 'jaguar.obj', function ( object ) {
    object.traverse( function ( child ) {
        if ( child instanceof THREE.Mesh ) {
            //override any material associated with .obj to customize
            child.material = myMaterial; //ie, a material you've already defined
    } } );

    //may need to scale object to fit your scene!
    var s = 0.2; object.scale.set( s, s, s );

    scene.add( object ); //add the object to your scene
} );
```

.OBJ files

```
//code to load in a regular texture
var objTex = new THREE.TextureLoader().load( 'jaguar.png' );
var uniforms = {  tex: { type: "t", value: objTex  } };
var myMaterial = new THREE.RawShaderMaterial( {
    uniforms: uniforms,
    vertexShader: tex_vs,
    fragmentShader: tex_fs,
} );
```

```
//GLSL fragment shader
uniform sampler2D tex;
varying vec2 vUV;
void main() {
    gl_FragColor = vec4(texture2D(tex, vUV).rgb, 1.0);
}
```

.OBJ files

```
//code to load in a environmental mapping texture
var material2 = new THREE.RawShaderMaterial( {
      uniforms: uniforms,
      vertexShader: em_vs,
      fragmentShader: em_fs
} );

//GLSL fragment shader
uniform samplerCube envMap;
varying vec3 vl; //this is the
Varying vec3 vWorldNormal;
void main() {
      vec3 rval = reflect( vl, vWorldNormal );
      vec4 envColor = textureCube( envMap, vec3( -rval.x, rval.yz ) );
      gl_FragColor = vec4(envColor);
}
```

In class exercise

- Find an .obj file from Google Poly
- Find a cube map texture from Humus.name
- https://creativecoding.soe.ucsc.edu/courses/cmpm163/code/week3_codeExamples.zip
- Use my "cubeMap.html" as a template to load in your object and skybox
- Can you create a shader that mixes together the jaguar's texture and the reflection from the skybox?

Game Graphics & Real-time Rendering

CMPM 163, W2018

Prof. Angus Forbes (instructor)
angus@ucsc.edu

Lucas Ferreira (TA)
lferreira@ucsc.edu

creativecoding.soe.ucsc.edu/courses/cmpm163
github.com/CreativeCodingLab