Outline: 10/31

- Recap Projection
- WebGL/Three.js intro
- Lab 6: Pyramid in WebGL
- MSA (mid-semester assessment)

- **HW 6**: due Tuesday Nov 8
- **Office Hours**: Mon/Tues 4-5pm
- **Reading**: Section 5.1
Recap Projection
Orthographic Projection

Robert Gardiner, the University of British Columbia
Perspective Projection: Frustum
Perspective Projection: Frustum
WebGL and Three.js
Three.js

- Useful and popular library for WebGL
- Allows us to create and manipulate 3D objects
- Still have a canvas (what we’re drawing on)

Main new elements:
- **Scene**: where we add 3D objects and lights
- **Camera**: where our “eye” is, not part of the scene
- **Renderer**: tool to draw the scene on the screen
Example WebGL code

Global Variables

```javascript
var scene, camera, renderer; // Three.js rendering basics.
var canvas; // The canvas on which the image is rendered.
```
Example WebGL code

**Global Variables**

```javascript
var scene, camera, renderer; // Three.js rendering basics.
var canvas; // The canvas on which the image is rendered.
```

**Inside createWorld()**

```javascript
scene = new THREE.Scene(); // Create a new scene which we can add objects to.
```
Example WebGL code

Global Variables

```javascript
var scene, camera, renderer; // Three.js rendering basics.
var canvas; // The canvas on which the image is rendered.
```

Inside `createWorld()`

```javascript
scene = new THREE.Scene(); // Create a new scene which we can add objects to.

// set up the geometry for our pyramid
var pyramidGeom = new THREE.Geometry();
```
Example WebGL code

Global Variables

```javascript
var scene, camera, renderer; // Three.js rendering basics.
var canvas; // The canvas on which the image is rendered.
```

Inside `createWorld()`

```javascript
scene = new THREE.Scene(); // Create a new scene which we can add objects to.

// set up the geometry for our pyramid
var pyramidGeom = new THREE.Geometry();

// Creates a material for the pyramid that is "matte" not "shiny".
var pyramidFaceMaterial = new THREE.MeshFaceMaterial(
    new THREE.MeshLambertMaterial( { color: 0xffffff, shading: THREE.FlatShading } ),
    new THREE.MeshLambertMaterial( { color: 0x99ffff, shading: THREE.FlatShading } ),
    new THREE.MeshLambertMaterial( { color: 0xff99ff, shading: THREE.FlatShading } ),
    new THREE.MeshLambertMaterial( { color: 0xffffffff, shading: THREE.FlatShading } ),
    new THREE.MeshLambertMaterial( { color: 0xff9999, shading: THREE.FlatShading } )
);

// Create the pyramid using the geometry and materials we've set up.
var pyramid = new THREE.Mesh( pyramidGeom, pyramidFaceMaterial );
```
Example WebGL code

Global Variables

```javascript
var scene, camera, renderer; // Three.js rendering basics.
var canvas; // The canvas on which the image is rendered.
```

Inside `createWorld()`

```javascript
scene = new THREE.Scene(); // Create a new scene which we can add objects to.

// set up the geometry for our pyramid
var pyramidGeom = new THREE.Geometry();

// Creates a material for the pyramid that is "matte" not "shiny".
var pyramidFaceMaterial = new THREE.MeshFaceMaterial( [
    new THREE.MeshLambertMaterial( { color: 0xffffff, shading: THREE.FlatShading } ),
    new THREE.MeshLambertMaterial( { color: 0x99ffff, shading: THREE.FlatShading } ),
    new THREE.MeshLambertMaterial( { color: 0xff99ff, shading: THREE.FlatShading } ),
    new THREE.MeshLambertMaterial( { color: 0xffff99, shading: THREE.FlatShading } ),
    new THREE.MeshLambertMaterial( { color: 0xff9999, shading: THREE.FlatShading } )
];

// Create the pyramid using the geometry and materials we've set up.
var pyramid = new THREE.Mesh( pyramidGeom, pyramidFaceMaterial );

pyramid.position.x = 1.5;

scene.add(pyramid);
```
Rendering Loop

```javascript
function render() {
    renderer.render(scene, camera);
}
```