CSC 240 Computer Graphics
Video 11: Three.js & WebGL

Nick Howe
Smith College

Some slides & content courtesy Sara Mathieson
WebGL

- Subset of OpenGL adapted for use in browsers*
- OpenGL: “cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics”
- Common baseline standard for graphics programming

* GL = Graphics Library
Three.js

- Useful and popular library for WebGL: https://threejs.org/
- Allows us to create and manipulate 3D objects
- Still have a canvas (what we’re drawing on/in)

- 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

The scene object keeps track of all objects that will make up your output image.

- For now, all we do with scenes is create them and add objects to them.

```javascript
// Create a new scene for adding objects to:
scene = new THREE.Scene();

// Add an already-created object to the scene:
scene.add(myObject);
```
Camera

A camera object describes the projection type and viewing frustum to be used by the renderer to create the image

- It is positioned using world coordinates but is not part of the scene (invisible!)

- Specify four parameters to create perspective camera
  - Field of view (in degrees)
  - Aspect ratio (should match the canvas)
  - Near clipping plane
  - Far clipping plane

```javascript
camera = new THREE.PerspectiveCamera(fov, aspect, near, far);
camera.position.z = z_pos;  // adjust position
```
Choosing Camera Parameters

Camera parameters exert critical control over what is visible

- Position: `z_pos` variable specifies height of camera above the world origin
- Clipping: `near` and `far` variables specify distances from the camera
- Field of view: `fov` variable specifies width of visible cone

```
z_pos = 90
near = 30
far = 60
```

```
z_pos = 45
near = 30
far = 60
```

```
z_pos = 90
near = 30
far = 120
```
The renderer does a lot of work behind the scenes

- Can take optional parameter list during creation
  ```javascript
  canvas = document.getElementById("glcanvas");
  renderer = new THREE.WebGLRenderer( { canvas: canvas, antialias: true } );
  ```
- Use it to set the background color
  ```javascript
  renderer.setClearColor(0); // Set background color (0, or 0x000000, is black).
  ```
- Tell it to draw the scene when ready
  ```javascript
  function render() {
    renderer.render(scene, camera);
  }
  ```
Questions

1. What is the relationship between OpenGL, WebGL, and Three.js?
   OpenGL is a generic graphics standard, WebGL is a variant specialized for web pages, and Three.js is an implementation and API for WebGL.

2. Where can I find documentation of Three.js?
   https://threejs.org/

3. Which object would be responsible for the following?
   a. Changing the background color
      Renderer
   b. Determining which objects are visible
      Camera
   c. Placing a new object in relation to others
      Scene
Scene Objects

Once your scene is ready, you’ll want to fill it with cool looking stuff.
What can go in a scene?

- **Lights** 🌞

- **Meshes**, composed of
  - Geometry
  - Material

[Image of a scene with objects and a link to portfolio]
Lights

Lights illuminate the scene. No lights = you see nothing!

- Lights come in various configurations
  - Point sources
  - Rectangular emitters
  - Ambient light & hemisphere light
  - Directional & spot lights

- All lights have color and intensity
- Spatial parameters will vary by type of light

```
// create bright white ambient light
var ambientLight = new THREE.AmbientLight(0xffffff,1);
scene.add(ambientLight);
```
Types of Lights in Three.js

- Directional
- Ambient
- Point
- Spot
- Hemisphere
- RectArea
Activity

Find the **three.js documentation web pages** and identify a class that represents each of the following:

- A type of light
- A type of camera
- A type of surface material
- The geometry for some predefined shape
Three.js Coding Demo
Object Creation Checklist

Steps to create a custom object in the scene:
1. Create a list of vertices, giving 3D coordinates
2. Create a list of faces, giving 3 vertices each using RHR
3. Specify materials for each face
4. Combine the geometry and materials together as a mesh
5. Add to scene and modify position & rotation as desired

https://www.reddit.com/r/blender/comments/b1m6jl/triceratops_render_critique/
Geometry defines the positions of an object’s surfaces

- **Vertices** are the corners
- **Faces** (made of triangles) connect coplanar vertices

[Face-Vertex Meshes](https://en.wikipedia.org/wiki/Polygon_mesh)
Geometry defines the positions of an object’s surfaces

```javascript
// set up the geometry for a shape
var myGeom = new THREE.Geometry();

// create the (x,y,z) points needed for this shape
myGeom.vertices = [
  new THREE.Vector3(1, 1, 0), // vertex number 0
  // ...
];

// using the indices of the vertices above, create triangular faces
myGeom.faces = [
  new THREE.Face3(0, 1, 2), // vertex number 0
  // ...
];
```

Order here matters!
Unlike real-world objects, in Three.js faces point only in one direction. Reverse side is totally transparent! Actually, you won’t even see the outline.

Visible side is determined by right hand rule. Fingers curl in order vertices are listed. Thumb side is exterior.

A-B-C points out of screen (also B-C-A or C-A-B)
A-C-B points into screen (also C-B-A or B-A-C)

Right hand rule:
Fingers curl in order vertices are listed.
Thumb side is exterior.

Looking at the side you want visible, list the points counterclockwise.
Materials define the way an object interacts with light in the environment (reflection, absorption, color mixing, etc.)

- Basic materials include color and reflectivity profile
- Advanced materials can have designs, texture, etc. — more on this later

```javascript
// Creates a material for an object
// that is "matte" not "shiny"
var myMaterial = [
    new THREE.MeshLambertMaterial( {
        color: 0xffffff,
        flatShading: true
    } ),
    // ...
];
```

https://www.ltu.se/edu/program/TMMA/TMMA-Materialteknik-master-1.76945?l=en
Mesh

Mesh is used to create all sorts of solid objects

- Combination of **geometry** and **material**

```javascript
// Create an object with prepared geometry and materials
var myObject = new THREE.Mesh( myGeom, myMaterial );
scene.add(myObject);
```
Putting It Together In Code

- **init:**
  - Create world
  - Start rendering loop

- **createWorld:**
  - Create the scene
  - Create a camera
  - Create a light & add it to scene
  - Create visible object(s) & add to scene
    - Standard object (cube, cylinder, etc.)
    - or-
    - Define geometry
    - Define materials
    - Build object

- **render:**
  - (Update scene)
  - Run renderer
Questions

1. Which of the polygons below is facing outwards from the screen?

2. Which light best simulates sunlight on earth, with parallel rays?
   DirectionalLight

3. Which of the following is NOT a predefined geometry offered by Three.js?
   a. SphereGeometry
   b. CubeGeometry
   c. ConeGeometry
   d. CylinderGeometry
   e. TorusGeometry
   f. IcosahedronGeometry
Review

After watching this video, you should be able to...

- Set up a web page for 3D rendering using Three.js
- Add a camera, lights, and visible objects to your scene
- Make informed choices about the different available options
- Create custom object geometries and combine them with materials
  - Write code to define vertices and combine them into faces
  - Control the directionality of triangular faces

Music: [https://www.bensound.com](https://www.bensound.com)