COSC342: WebGL Introduction

WebGL, Three.js

Objectives

- Introduction to WebGL
- Introduction to Three.js

Introduction

The following lab will give you the background for implementing WebGL and using Three.js for creating 3D graphics in the browser. Make sure that you copy the source files from the

`/home/cshome/coursework/342/pickup/labs/lab12-WebGLIntro/` directory into a directory within your home directory.

WebGL

Have a look into the folder WebGLdemo. As explained in the lecture these files render a red triangle in the browser. Double click on the html file to open it in a browser. ⇒ *Read through the program code, and see what it is doing.*

Exercise

- Change the background colour from black background to dark grey by using gl.clearColor.
- Change the canvas size of the WebGL output to 1200x1000.
- Use the following methods for passing a colour to the shader:

```
1    var someColLoc = gl.getUniformLocation(gl.program, "color")
     )
     gl.uniform3f (someColLoc,  1.0, 0.0, 1.0);
3
```

Make sure that you don't copy and paste because otherwise the quotation marks will be in the wrong format. Use the passed color in the fragment shader:

```
   uniform mediump vec3 color;
2
```

- Use this uniform variable **color** in the fragment shader to change the colour of the triangle.
- Change the triangle output into a quad output by rendering two triangles.

Three.js

Have a look into the folder ThreeJSDemo and helloworld.html. As explained in the lecture these files render a rotating box in the browser using the THREE.JS library. Double click on the html file to open it in a browser. ⇒ *Read through the program code, and see what it is doing.*

Exercise

- Create some more realistic illumination by creating a point light source at position =(1.5,1.5,2.0) and adding it to the scene. Also change the cube's material from basic material to MeshLambertMaterial. For more details have look into the documentation of Three.js: `http://threejs.org/docs/`
- Create a second cube with a different colour in your scene and make it rotate in the opposite direction.
- Create a sphere with a colour of your choice and add it to your scene.
- Change the light source to a spot light and the frustum of the light source using SpotLightHelper and adding it to the scene:

```
1    lightHelper = new THREE.SpotLightHelper( light );
     scene.add(lightHelper)
3
```

- Create a ground plane geometry using a flat BoxGeometry and enable shadow mapping:

```
  renderer.shadowMap.enabled = true;
2 renderer.shadowMap.type = THREE.PCFSoftShadowMap;
  light.castShadow = true;
4 groundPlane.receiveShadow = true;
```

For more details about creating shadows have look into the documentation of Three.js: `http://threejs.org/docs/`