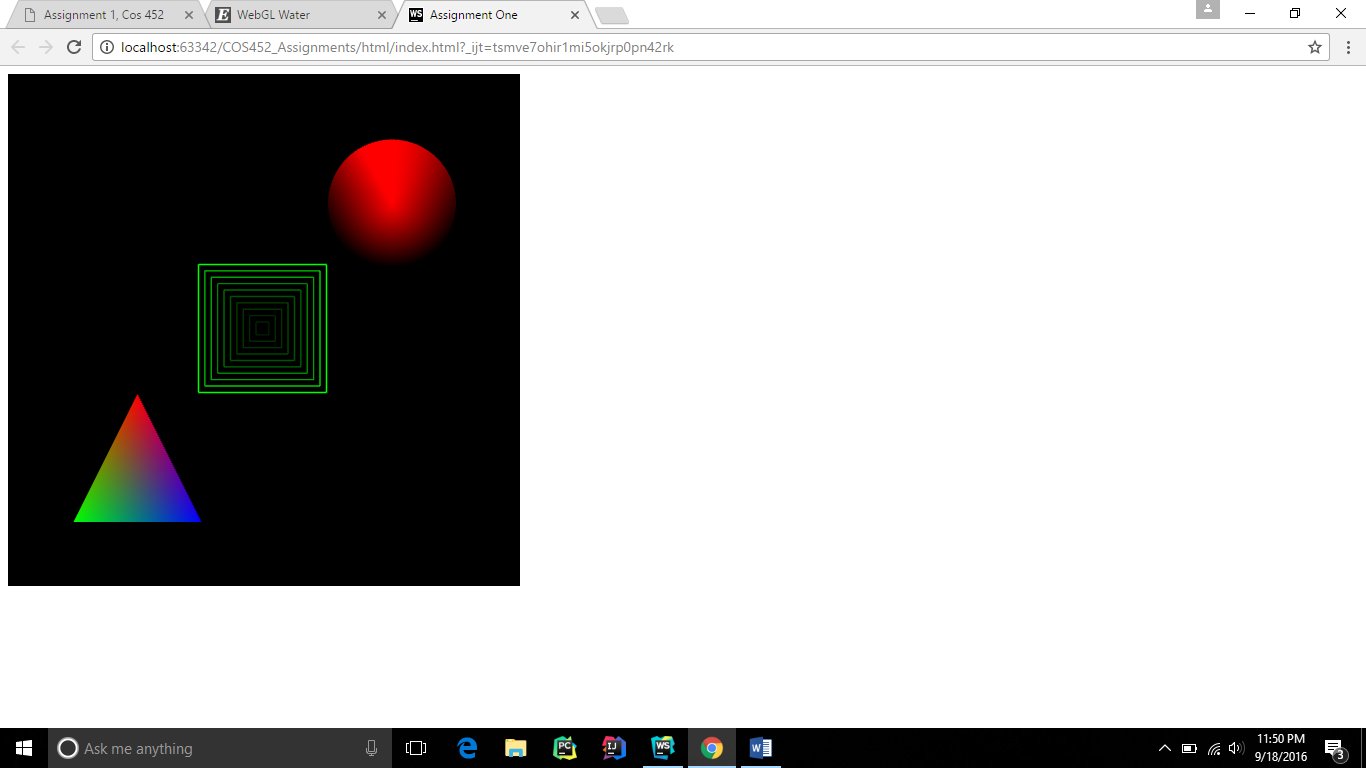
COS 452

Assignment 1

Tyler LaBerge

Output screenshot:



GitHub repo link: https://github.com/tylerlaberge/COS452\_Assignments

Interesting WebGL example: <http://madebyevan.com/webgl-water/>

I found this WebGL example interesting because the water effects and interactivity looked very nice. The way the water propagates out when you drop the ball looks very realistic.

Some things I understood from this example follow.

In main.js in the draw function for this example there are a few things I understand. First off the gl.clear function call I know clears all the colors from the screen, except for the background color if that was set earlier. I also see that gl.DEPTH\_TEST is being enabled which will discard some fragments from the fragment shader output if they fail the depth test. Finally, I would assume renderCube, renderWater, and renderSphere will render the cube, water, and sphere respectively.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function draw() {

// Change the light direction to the camera look vector when the L key is pressed

if (GL.keys.L) {

renderer.lightDir = GL.Vector.fromAngles((90 - angleY) \* Math.PI / 180, -angleX \* Math.PI / 180);

if (paused) renderer.updateCaustics(water);

}

gl.clear(gl.COLOR\_BUFFER\_BIT | gl.DEPTH\_BUFFER\_BIT);

gl.loadIdentity();

gl.translate(0, 0, -4);

gl.rotate(-angleX, 1, 0, 0);

gl.rotate(-angleY, 0, 1, 0);

gl.translate(0, 0.5, 0);

gl.enable(gl.DEPTH\_TEST);

renderer.sphereCenter = center;

renderer.sphereRadius = radius;

renderer.renderCube();

renderer.renderWater(water, cubemap);

renderer.renderSphere();

gl.disable(gl.DEPTH\_TEST);

}

};

In water.js there is a lot of shader code, one of the snippets being the dropShader below. I understand that uniform attributes of the shader stay the same during a draw and that a varying attribute is passed between the vertex shader and fragment shader.

this.dropShader = new GL.Shader(vertexShader, '\

const float PI = 3.141592653589793;\

uniform sampler2D texture;\

uniform vec2 center;\

uniform float radius;\

uniform float strength;\

varying vec2 coord;\

void main() {\

/\* get vertex info \*/\

vec4 info = texture2D(texture, coord);\

\

/\* add the drop to the height \*/\

float drop = max(0.0, 1.0 - length(center \* 0.5 + 0.5 - coord) / radius);\

drop = 0.5 - cos(drop \* PI) \* 0.5;\

info.r += drop \* strength;\

\

gl\_FragColor = info;\

}\

');

In the Water objects addDrop method the dropShader above is being used. It looks like the dropShaders uniforms are being set before the draw in the dropShader.uniforms call.

Water.prototype.addDrop = function(x, y, radius, strength) {

var this\_ = this;

this.textureB.drawTo(function() {

this\_.textureA.bind();

this\_.dropShader.uniforms({

center: [x, y],

radius: radius,

strength: strength

}).draw(this\_.plane);

});

this.textureB.swapWith(this.textureA);

};

Lastly, in the renderer’s renderWater function I can see a water shaders uniforms are being set in a similar manner as above in the this.waterShaders[i].uniforms call.

Renderer.prototype.renderWater = function(water, sky) {

var tracer = new GL.Raytracer();

water.textureA.bind(0);

this.tileTexture.bind(1);

sky.bind(2);

this.causticTex.bind(3);

gl.enable(gl.CULL\_FACE);

for (var i = 0; i < 2; i++) {

gl.cullFace(i ? gl.BACK : gl.FRONT);

this.waterShaders[i].uniforms({

light: this.lightDir,

water: 0,

tiles: 1,

sky: 2,

causticTex: 3,

eye: tracer.eye,

sphereCenter: this.sphereCenter,

sphereRadius: this.sphereRadius

}).draw(this.waterMesh);

}

gl.disable(gl.CULL\_FACE);

};

Questions:

How is the ball reflecting off the water?

How is the texture of the water created?