Assignment #6: WebGL Texture Mapping
Shane Steiner
T00622768
4/3/2021

Problem descriptions:
Create 2 cubes with textures,lighting and perspective projection

```
<!DOCTYPE html>
<html>

<button id = "ButtonX">Rotate X</button>
<button id = "ButtonY">Rotate Y</button>
<button id = "ButtonZ">Rotate Z</button>
<button id = "ButtonT">Toggle Rotation</button>
<button id = "ButtonPers">Toggle Perspective</button>
<div>Note: Perspective projection has fases visible that should not be
visible. i couldn't figure out why</div>


<script id="vertex-shader" type="x-shader/x-vertex">
#version 300 es

in vec4 aPosition;
in vec4 aColor;
in vec2 aTexCoord;
in  vec3 aNormal;

out vec4 vColor;
out vec2 vTexCoord;

uniform mat4 uProjectionMatrix;
uniform mat4 uModelViewMatrix;

uniform vec4 uAmbientProduct, uDiffuseProduct, uSpecularProduct;

uniform vec4 uLightPosition;
uniform float uShininess ;

void main()
{
    vec3 pos = -(uModelViewMatrix * aPosition).xyz;

    //not fixed light postion
    vec3 light = uLightPosition.xyz;
    vec3 L = normalize(light - pos);


    vec3 E = normalize(-pos);
    vec3 H = normalize(L + E);

    vec4 NN = vec4(aNormal,0);

    // Transform vertex normal into eye coordinates

    vec3 N = normalize((uModelViewMatrix*NN).xyz);

    // Compute terms in the illumination equation
    vec4 ambient = vec4(0.2,0,0.2,1);

    float Kd = max(dot(L, N), 0.0);
    vec4  diffuse = Kd*vec4(0.2,0,0.2,1);

    float Ks = pow( max(dot(N, H), 0.0),5.0);
    vec4  specular = Ks * vec4(1.0,0,0.2,1);

    if( dot(L, N) < 0.0 ) {
      specular = vec4(0.0, 0.0, 0.0, 1.0);
    }
    vColor = ( 1.0*ambient  +1.0 * diffuse + 1.0 *specular )+ 1.0* aColor;
    vTexCoord = aTexCoord;
    gl_Position = uProjectionMatrix * uModelViewMatrix * aPosition;
    gl_Position.z = -gl_Position.z;
}
</script>

<script id="fragment-shader" type="x-shader/x-fragment">
#version 300 es

precision mediump float;

in vec4 vColor;
in  vec2 vTexCoord;
```

```
out vec4 fColor;

uniform sampler2D uTextureMap;

void
main()
{
    fColor = vColor*texture(uTextureMap, vTexCoord);
}
</script>

<script type="text/javascript"
src="../assignment6/Common/initShaders.js"></script>
<script type="text/javascript"
src="../assignment6/Common/MVnew.js"></script>
<script type="text/javascript" src="assignement6.js"></script>


<body>
<canvas id="gl-canvas" width="1024" height="1024">
Oops ... your browser doesn't support the HTML5 canvas element
</canvas>
</body>
</html>
```

```
"use strict";

var canvas;
var gl;
var program;

var modelViewMatrix, projectionMatrix;

var numPositions = 72;
var normalsArray = [];
// var numPositions = 36;
```

```
var texSize = 64;
var time =0;
var flag = false;
var perspectiveFlag = false;

var lightPosition = vec4(0.0, 2.0, 0.0, 1.0);

var positionsArray = [];
var colorsArray = [];
var texCoordsArray = [];

var texCoordBack = [
    vec2(0, 0.5), // bottom left
    vec2(0, 0), //top left
    vec2(0.25, 0), //top right
    vec2(0.25, 0.5), // bottom right
];
var texCoordFront = [
    vec2(0.75, 0.5), //top right
    vec2(0.75, 1.0), // bottom right
    vec2(0.5, 1.0), // bottom left
    vec2(0.5, 0.5), //top left
];
var texCoordLeft = [
    vec2(0.5, 0), //top right
    vec2(0.5, 0.5), // bottom right
    vec2(0.25, 0.5), // bottom left
    vec2(0.25, 0), //top left
];
var texCoordRight = [
    vec2(0.75, 0), //top right
    vec2(0.75, 0.5), // bottom right
    vec2(0.5, 0.5), // bottom left
    vec2(0.5, 0), //top left
];
var texCoordTop = [
    vec2(0, 0.5), //top left
    vec2(0.25, 0.5), //top right
    vec2(0.25, 1.0), // bottom right
    vec2(0, 1.0), // bottom left
```

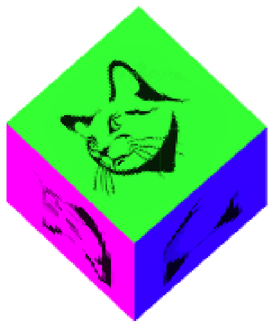
```
];
var texCoordBottom = [
    vec2(0.25, 0.50), //top left
    vec2(0.50, 0.50), //top right
    vec2(0.50, 1.0), // bottom right
    vec2(0.25, 1.0), // bottom left
];


var vertices = [
    vec4(-0.5, -0.5, 0.5, 1.0),
    vec4(-0.5, 0.5, 0.5, 1.0),
    vec4(0.5, 0.5, 0.5, 1.0),
    vec4(0.5, -0.5, 0.5, 1.0),
    vec4(-0.5, -0.5, -0.5, 1.0),
    vec4(-0.5, 0.5, -0.5, 1.0),
    vec4(0.5, 0.5, -0.5, 1.0),
    vec4(0.5, -0.5, -0.5, 1.0),

    vec4(-0.5 + 2.5, -0.5 + 2.5, 0.5 + 2.5, 1.0),
    vec4(-0.5 + 2.5, 0.5 + 2.5, 0.5 + 2.5, 1.0),
    vec4(0.5 + 2.5, 0.5 + 2.5, 0.5 + 2.5, 1.0),
    vec4(0.5 + 2.5, -0.5 + 2.5, 0.5 + 2.5, 1.0),
    vec4(-0.5 + 2.5, -0.5 + 2.5, -0.5 + 2.5, 1.0),
    vec4(-0.5 + 2.5, 0.5 + 2.5, -0.5 + 2.5, 1.0),
    vec4(0.5 + 2.5, 0.5 + 2.5, -0.5 + 2.5, 1.0),
    vec4(0.5 + 2.5, -0.5 + 2.5, -0.5 + 2.5, 1.0),
];


var vertexColors = [
    vec4(0.0, 0.0, 0.0, 1.0),  // black
    vec4(1.0, 0.0, 0.0, 1.0),  // red
    vec4(1.0, 1.0, 0.0, 1.0),  // yellow
    vec4(0.0, 1.0, 0.0, 1.0),  // green
    vec4(0.0, 0.0, 1.0, 1.0),  // blue
    vec4(1.0, 0.0, 1.0, 1.0),  // magenta
    vec4(0.0, 1.0, 1.0, 1.0),  // white
    vec4(0.0, 1.0, 1.0, 1.0),   // cyan
    vec4(0.0, 0.0, 0.0, 1.0),  // black
    vec4(1.0, 0.0, 0.0, 1.0),  // red
    vec4(1.0, 1.0, 0.0, 1.0),  // yellow
    vec4(0.0, 1.0, 0.0, 1.0),  // green
    vec4(0.0, 0.0, 1.0, 1.0),  // blue
    vec4(1.0, 0.0, 1.0, 1.0),  // magenta
    vec4(0.0, 1.0, 1.0, 1.0),  // white
    vec4(0.0, 1.0, 1.0, 1.0),   // cyan
];
window.onload = init;


var xAxis = 0;
var yAxis = 1;
var zAxis = 2;
var axis = xAxis;

var theta = vec3(0, 0, 0);

var thetaLoc;

function configureTexture() {
    var texture = gl.createTexture();

    var image = new Image();
    image.src = "../assignment6/photos/catMap.png";
    image.addEventListener('load', function () {
        gl.activeTexture(gl.TEXTURE0);
        gl.bindTexture(gl.TEXTURE_2D, texture);
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA,
gl.UNSIGNED_BYTE, image);
        gl.generateMipmap(gl.TEXTURE_2D);
    });

}
//loads arrays
function quad(a, b, c, d, texCoord) {

    var t1 = subtract(vertices[b], vertices[a]);
    var t2 = subtract(vertices[c], vertices[b]);
    var normal = cross(t1, t2);
    normal = vec3(normal);

    positionsArray.push(vertices[a]);
    normalsArray.push(normal);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[0]);

    positionsArray.push(vertices[b]);
    normalsArray.push(normal);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[1]);

    positionsArray.push(vertices[c]);
    normalsArray.push(normal);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[2]);

    positionsArray.push(vertices[a]);
    normalsArray.push(normal);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[0]);

    positionsArray.push(vertices[c]);
    normalsArray.push(normal);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[2]);

    positionsArray.push(vertices[d]);
    normalsArray.push(normal);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[3]);
}

//load quads
function colorCube() {
    quad(2, 3, 7, 6, texCoordLeft);
    quad(1, 0, 3, 2, texCoordFront);
    quad(3, 0, 4, 7, texCoordBottom);
    quad(6, 5, 1, 2, texCoordTop);
    quad(4, 5, 6, 7, texCoordBack);
    quad(5, 4, 0, 1, texCoordRight);

    quad(10, 11, 15, 14, texCoordLeft);
    quad(9, 8, 11, 10, texCoordFront);
    quad(11, 8, 12, 15, texCoordBottom);
    quad(14, 13, 9, 10, texCoordTop);
    quad(12, 13, 14, 15, texCoordBack);
    quad(13, 12, 8, 9, texCoordRight);
}


function init() {
    canvas = document.getElementById("gl-canvas");

    gl = canvas.getContext('webgl2');
    if (!gl) alert("WebGL 2.0 isn't available");

    gl.viewport(0, 0, canvas.width, canvas.height);
    gl.clearColor(1.0, 1.0, 1.0, 1.0);

    gl.enable(gl.DEPTH_TEST);

    //
    //  Load shaders and initialize attribute buffers
    //
    program = initShaders(gl, "vertex-shader", "fragment-shader");
    gl.useProgram(program);

    colorCube();

    var nBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, nBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(normalsArray), gl.STATIC_DRAW);

    var normalLoc = gl.getAttribLocation(program, "aNormal");
    gl.vertexAttribPointer(normalLoc, 3, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(normalLoc);

    var cBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, cBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(colorsArray), gl.STATIC_DRAW);
    var colorLoc = gl.getAttribLocation(program, "aColor");
    gl.vertexAttribPointer(colorLoc, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(colorLoc);

    var vBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, vBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(positionsArray),
gl.STATIC_DRAW);
    var positionLoc = gl.getAttribLocation(program, "aPosition");
    gl.vertexAttribPointer(positionLoc, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(positionLoc);

    var tBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, tBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(texCoordsArray),
gl.STATIC_DRAW);
    var texCoordLoc = gl.getAttribLocation(program, "aTexCoord");
    gl.vertexAttribPointer(texCoordLoc, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(texCoordLoc);

    configureTexture();

    gl.uniform1i(gl.getUniformLocation(program, "uTextureMap"), 0);

    thetaLoc = gl.getUniformLocation(program, "uTheta");

    //initalise projectionMatrix
    projectionMatrix = mat4();

    document.getElementById("ButtonX").onclick = function () { axis =
xAxis; };
    document.getElementById("ButtonY").onclick = function () { axis =
yAxis; };
    document.getElementById("ButtonZ").onclick = function () { axis =
zAxis; };
    document.getElementById("ButtonT").onclick = function () { flag =
!flag; };
```

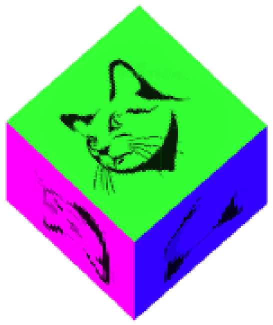
```
    document.getElementById("ButtonPers").onclick = function () {
perspectiveFlag = !perspectiveFlag; };

    render();
}

var render = function () {

    var scaleFactor = 0.2;
    var cordShiftX = 0;
    var cordShiftY = 0;
    var cordShiftZ = 0;

    var rotateX = 0;
    var rotateY = 0;
    var rotateZ = 0;

    var scaleMat = mat4(scaleFactor, 0, 0, 0,
        0, scaleFactor, 0, 0,
        0, 0, scaleFactor, 0,
        0, 0, 0, 1);
    var translationMat = mat4(1, 0, 0, cordShiftX,
        0, 1, 0, cordShiftY,
        0, 0, 1, cordShiftZ,
        0, 0, 0, 1);


    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    if (flag) theta[axis] += 1.0;

    modelViewMatrix = mat4();
    modelViewMatrix = mult(modelViewMatrix, scaleMat);
    modelViewMatrix = mult(modelViewMatrix, translationMat);
    modelViewMatrix = mult(modelViewMatrix, rotate(theta[xAxis] + rotateX,
vec3(1, 0, 0)));
    modelViewMatrix = mult(modelViewMatrix, rotate(theta[yAxis] + rotateY,
vec3(0, 1, 0)));
    modelViewMatrix = mult(modelViewMatrix, rotate(theta[zAxis] + rotateZ,
vec3(0, 0, 1)));
```

```
    gl.uniformMatrix4fv(gl.getUniformLocation(program,
"uModelViewMatrix"), false, flatten(modelViewMatrix));

    //light
    lightPosition[0] = 2.5*Math.sin(0.06*time);
    lightPosition[2] = 2.5*Math.cos(0.06*time);
    time += 1;
    gl.uniform4fv(gl.getUniformLocation(program, "uLightPosition"),
lightPosition);

    //perspective
    if(perspectiveFlag)
    projectionMatrix = ProjectionPerspective(projectionMatrix);
    else
    projectionMatrix = mat4();
    gl.uniformMatrix4fv(gl.getUniformLocation(program,
"uProjectionMatrix"), false, flatten(projectionMatrix));

    gl.drawArrays(gl.TRIANGLES, 0, numPositions);
    requestAnimationFrame(render);
}

//moves camera back
var translationMat2 = mat4(1, 0, 0, 0,
    0, 1, 0, 0,
    0, 0, 1, -1,
    0, 0, 0, 1);
    //used for projection perspective
function ProjectionPerspective(projectionMatrix) {
    var fovy = 100, aspect = 1, near = 0, far = 100;
    projectionMatrix = perspective(fovy, aspect, near, far);

    projectionMatrix = mult(projectionMatrix, translationMat2);
    gl.uniformMatrix4fv(gl.getUniformLocation(program,
"uProjectionMatrix"),
        false, flatten(projectionMatrix));
      return projectionMatrix;
}
```

Rotate X Rotate Y Rotate Z Toggle Rotation Toggle Perspective

Note: Perspective projection has fases visible that should not be visible. i couldn't figure out why





Rotate X | Rotate Y | Rotate Z | Toggle Rotation | Toggle Perspective

Note: Perspective projection has fases visible that should not be visible. i couldn't figure out why