Materiais usando WebGL, baseado nos materiais "Learning WebGL- 3D Programming for the Web": <http://learningwebgl.com/blog/?page_id=1217>

**Notas de 2014:**

Você pode rodar os programas hae???.html acessando-os de internet ou como baixando-os e executando do HD local.

1) Acessando-os da internet, esses programas executaram corretamente em: Firefox, Explorer e Chrome de Windows; Chrome, Firefox e Internet do Android; Safari e Chrome de iOS-iPad. No ano passado (2013), Explorer e iPad não aceitavam WebGL.

2) Dando download e rodando como em disco local, consegui executá-los em Firefox e em Explorer com a opção “permitir conteúdo bloqueado”. Não consegui executar alguns dos programas em Chrome. Talvez Chrome permita executá-los se chamá-lo com a opção “--allow-file-access-from-files”.

3) Você pode acessar os arquivos locais como se viessem da internet instalando  “XAMPP” (que instala Apache).

Nota 1: Leia os programas hae???.html num editor de texto para ver os comentários.

Nota 2: Para ver as mensagens de erro e warning em Chrome ou Firefox, aperte CTRL-SHIFT-I.

**HTML puro:**

**<!-- hae00a.html -->**

**<html>**

 **<head>**

 **<title>hae00a.html</title>**

 **</head>**

 **<body>**

 **teste de HTML**

 **</body>**

**</html>**

**-------------------------------------**

**HTML com javascript. Abre uma janela de alerta.**

**<!-- hae00b.html -->**

**<html>**

 **<head>**

 **<title>hae00b.html</title>**

 **<script type="text/javascript">**

 **function webGLStart() {**

 **alert("Alo WebGL e javascript");**

 **}**

 **</script>**

 **</head>**

 **<body onload="webGLStart();">**

 **Testando HTML com javascript**

 **</body>**

**</html>**

**----------------------------------------**

**HTML e javascript acessando WebGL**

**<!-- hae00c.html -->**

**<html>**

 **<head>**

 **<title>hae00c.html</title>**

 **<script type="text/javascript">**

 **var gl; // variavel global contexto WebGL**

 **function initGL(canvas) {**

 **try {**

 **gl = canvas.getContext("experimental-webgl");**

 **gl.viewportWidth = canvas.width;**

 **gl.viewportHeight = canvas.height;**

 **} catch (e) {**

 **alert("Peguei erro. Hae.");**

 **}**

 **if (!gl) {**

 **alert("Could not initialise WebGL, sorry :-(");**

 **}**

 **}**

 **function drawScene() {**

 **gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);**

 **gl.clear(gl.COLOR\_BUFFER\_BIT);**

 **}**

 **function webGLStart() {**

 **var canvas = document.getElementById("hae00c-canvas");**

 **initGL(canvas);**

 **gl.clearColor(0.3, 0.5, 0.7, 1.0); //cor de fundo**

 **drawScene();**

 **}**

 **</script>**

 **</head>**

 **<body onload="webGLStart();">**

 **<canvas id="hae00c-canvas" width="800" height="800">**

 **</canvas>**

 **</body>**

**</html>**

**<!-- hae01.html -->**

**<html>**

**<head>**

**<title>Learning WebGL - lesson 1</title>**

**<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>**

**<script id="shader-fs" type="x-shader/x-fragment">**

 **//fragment shader ou pixel shader**

 **precision mediump float;**

 **void main(void) {**

 **// cor do fragmento**

 **gl\_FragColor = vec4(1.0, 1.0, 0.0, 1.0);**

 **}**

**</script>**

**<script id="shader-vs" type="x-shader/x-vertex">**

 **attribute vec3 aVertexPosition;**

 **uniform mat4 uMVMatrix;**

 **uniform mat4 uPMatrix;**

 **void main(void) {**

 **// coordenadas homogeneas do vertice**

 **gl\_Position = uPMatrix \* uMVMatrix \* vec4(aVertexPosition, 1.0);**

 **}**

**</script>**

**<script type="text/javascript">**

 **var gl;**

 **function initGL(canvas) {**

 **try {**

 **gl = canvas.getContext("experimental-webgl");**

 **gl.viewportWidth = canvas.width;**

 **gl.viewportHeight = canvas.height;**

 **} catch (e) {**

 **}**

 **if (!gl) {**

 **alert("Could not initialise WebGL, sorry :-(");**

 **}**

 **}**

 **function getShader(gl, id) {**

 **var shaderScript = document.getElementById(id);**

 **if (!shaderScript) {**

 **return null;**

 **}**

 **var str = "";**

 **var k = shaderScript.firstChild;**

 **while (k) {**

 **if (k.nodeType == 3) {**

 **str += k.textContent;**

 **}**

 **k = k.nextSibling;**

 **}**

 **var shader;**

 **if (shaderScript.type == "x-shader/x-fragment") {**

 **shader = gl.createShader(gl.FRAGMENT\_SHADER);**

 **} else if (shaderScript.type == "x-shader/x-vertex") {**

 **shader = gl.createShader(gl.VERTEX\_SHADER);**

 **} else {**

 **return null;**

 **}**

 **gl.shaderSource(shader, str);**

 **gl.compileShader(shader);**

 **if (!gl.getShaderParameter(shader, gl.COMPILE\_STATUS)) {**

 **alert(gl.getShaderInfoLog(shader));**

 **return null;**

 **}**

 **return shader;**

 **}**

 **var shaderProgram;**

 **function initShaders() {**

 **var fragmentShader = getShader(gl, "shader-fs");**

 **var vertexShader = getShader(gl, "shader-vs");**

 **shaderProgram = gl.createProgram();**

 **gl.attachShader(shaderProgram, vertexShader);**

 **gl.attachShader(shaderProgram, fragmentShader);**

 **gl.linkProgram(shaderProgram);**

 **if (!gl.getProgramParameter(shaderProgram, gl.LINK\_STATUS)) {**

 **alert("Could not initialise shaders");**

 **}**

 **gl.useProgram(shaderProgram);**

 **shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram,**

 **"aVertexPosition");**

 **gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);**

 **shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");**

 **shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");**

 **}**

 **var mvMatrix = mat4.create();**

 **var pMatrix = mat4.create();**

 **function setMatrixUniforms() {**

 **gl.uniformMatrix4fv(shaderProgram.pMatrixUniform, false, pMatrix);**

 **gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform, false, mvMatrix);**

 **}**

 **var triangleVertexPositionBuffer; // variavel global - vertices do triangulo**

 **var squareVertexPositionBuffer; // variavel global - vertices do quadrado**

 **function initBuffers() {**

 **triangleVertexPositionBuffer = gl.createBuffer(); // cria buffer na placa grafica**

 **// todas as operacoes em seguida serao feitas em triangleVertexPositionBuffer**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexPositionBuffer);**

 **var vertices = [**

 **0.0, 1.0, 0.0, // alterei**

 **-1.0, -1.0, 0.0,**

 **1.0, -1.0, 0.0**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);**

 **triangleVertexPositionBuffer.itemSize = 3;**

 **triangleVertexPositionBuffer.numItems = 3;**

 **squareVertexPositionBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexPositionBuffer);**

 **vertices = [**

 **1.0, 1.0, 0.0,**

 **-1.0, 1.0, 0.0,**

 **1.0, -1.0, 0.0,**

 **-1.0, -1.0, 0.0**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);**

 **squareVertexPositionBuffer.itemSize = 3;**

 **squareVertexPositionBuffer.numItems = 4;**

 **}**

 **function drawScene() {**

 **gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);**

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

 **mat4.perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);**

 **mat4.identity(mvMatrix);**

 **mat4.translate(mvMatrix, [-1.5, 0.0, -7.0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,**

 **triangleVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **setMatrixUniforms();**

 **gl.drawArrays(gl.TRIANGLES, 0, triangleVertexPositionBuffer.numItems);**

 **mat4.translate(mvMatrix, [3.0, 0.0, 0.0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,**

 **squareVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **setMatrixUniforms();**

 **gl.drawArrays(gl.TRIANGLE\_STRIP, 0, squareVertexPositionBuffer.numItems);**

 **}**

 **function webGLStart() {**

 **var canvas = document.getElementById("lesson01-canvas");**

 **initGL(canvas);**

 **initShaders();**

 **initBuffers();**

 **gl.clearColor(0.3, 0.5, 0.7, 1.0); //cor de fundo**

 **gl.enable(gl.DEPTH\_TEST);**

 **drawScene();**

 **}**

**</script>**

**</head>**

**<body onload="webGLStart();">**

 **<canvas id="lesson01-canvas" width="500" height="500"></canvas>**

**</body>**

**</html>**

****

**<!-- hae01c.html -->**

**<html>**

**<head>**

**<title>Learning WebGL - lesson 1</title>**

**<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>**

**<script type="text/javascript">**

 **var gl; // variavel global contexto WebGL**

 **function initGL(canvas) {**

 **try {**

 **gl = canvas.getContext("experimental-webgl");**

 **gl.viewportWidth = canvas.width;**

 **gl.viewportHeight = canvas.height;**

 **} catch (e) {**

 **alert("Peguei erro. Hae.");**

 **}**

 **if (!gl) {**

 **alert("Could not initialise WebGL, sorry :-(");**

 **}**

 **}**

 **function compilaShader(type, st) {**

 **var shader;**

 **if (type == "f") {**

 **shader = gl.createShader(gl.FRAGMENT\_SHADER);**

 **} else if (type == "v") {**

 **shader = gl.createShader(gl.VERTEX\_SHADER);**

 **} else {**

 **return null;**

 **}**

 **gl.shaderSource(shader, st);**

 **gl.compileShader(shader);**

 **if (!gl.getShaderParameter(shader, gl.COMPILE\_STATUS)) {**

 **alert(gl.getShaderInfoLog(shader));**

 **return null;**

 **}**

 **return shader;**

 **}**

 **var shaderProgram;**

 **function initShaders() {**

 **var fragmentShader = compilaShader("f",**

 **"precision mediump float; \**

 **void main(void) { \**

 **gl\_FragColor = vec4(1.0, 1.0, 0.0, 1.0); \**

 **}"**

 **);**

 **var vertexShader = compilaShader("v",**

 **"attribute vec3 aVertexPosition; \**

 **uniform mat4 uMVMatrix; \**

 **uniform mat4 uPMatrix; \**

 **void main(void) { \**

 **gl\_Position = uPMatrix \* uMVMatrix \* vec4(aVertexPosition, 1.0); \**

 **}"**

 **);**

 **shaderProgram = gl.createProgram();**

 **gl.attachShader(shaderProgram, vertexShader);**

 **gl.attachShader(shaderProgram, fragmentShader);**

 **gl.linkProgram(shaderProgram);**

 **if (!gl.getProgramParameter(shaderProgram, gl.LINK\_STATUS)) {**

 **alert("Could not initialise shaders");**

 **}**

 **gl.useProgram(shaderProgram);**

 **shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram,**

 **"aVertexPosition");**

 **gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);**

 **shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");**

 **shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");**

 **}**

 **var mvMatrix = mat4.create();**

 **var pMatrix = mat4.create();**

 **function setMatrixUniforms() {**

 **gl.uniformMatrix4fv(shaderProgram.pMatrixUniform, false, pMatrix);**

 **gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform, false, mvMatrix);**

 **}**

 **var triangleVertexPositionBuffer; // variavel global - vertices do triangulo**

 **var squareVertexPositionBuffer; // variavel global - vertices do quadrado**

 **function initVertices(vertices, numItems) {**

 **vertexPositionBuffer = gl.createBuffer(); // cria buffer na placa grafica**

 **// todas as operacoes em seguida serao feitas em triangleVertexPositionBuffer**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, vertexPositionBuffer);**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);**

 **vertexPositionBuffer.itemSize = 3;**

 **vertexPositionBuffer.numItems = numItems;**

 **return vertexPositionBuffer;**

 **}**

 **// envia vertices para placa grafica**

 **function initBuffers() {**

 **triangleVertexPositionBuffer=initVertices(**

 **[ 0.0, 2.0, 0.0, // alterei**

 **-1.0, -1.0, 0.0,**

 **1.0, -1.0, 0.0**

 **], 3);**

 **squareVertexPositionBuffer=initVertices(**

 **[ 1.0, 1.0, 0.0,**

 **-1.0, 1.0, 0.0,**

 **1.0, -1.0, 0.0,**

 **-1.0, -1.0, 0.0**

 **], 4);**

 **}**

 **function drawScene() {**

 **// gl.viewportWidth e Height inicializadas em initGl(canvas)**

 **gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);**

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

 **//vertical field of view 45**

 **//near=0.1 and far=100.0**

 **//pMatrix=perspectiva**

 **mat4.perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);**

 **//mvMatrix model view**

 **mat4.identity(mvMatrix);**

 **mat4.translate(mvMatrix, [-1.5, 0.0, -7.0]);**

 **//mat4.translate(mvMatrix, [-2.5, 0.0, -7.0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,**

 **triangleVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **// Copia mvMatrix e pMatrix para placa grafica**

 **setMatrixUniforms();**

 **gl.drawArrays(gl.TRIANGLES, 0, triangleVertexPositionBuffer.numItems);**

 **//mat4.translate(mvMatrix, [3.0, 0.0, 0.0]);**

 **mat4.translate(mvMatrix, [2.0, 0.0, 0.0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,**

 **squareVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **setMatrixUniforms();**

 **gl.drawArrays(gl.TRIANGLE\_STRIP, 0, squareVertexPositionBuffer.numItems);**

 **}**

 **function webGLStart() {**

 **var canvas = document.getElementById("lesson01-canvas");**

 **initGL(canvas);**

 **initShaders();**

 **initBuffers();**

 **gl.clearColor(0.3, 0.5, 0.7, 1.0); //cor de fundo**

 **gl.enable(gl.DEPTH\_TEST);**

 **drawScene();**

 **}**

**</script>**

**</head>**

**<body onload="webGLStart();">**

 **<canvas id="lesson01-canvas" width="300" height="300"></canvas>**

**</body>**

**</html>**

**// glMatrix v0.9.5**

**glMatrixArrayType=typeof Float32Array!="undefined"?Float32Array:typeof WebGLFloatArray!="undefined"?WebGLFloatArray:Array;**

**var vec3={};**

**vec3.create=function(a){var b=new glMatrixArrayType(3);if(a){b[0]=a[0];b[1]=a[1];b[2]=a[2]}return b};**

**vec3.set=function(a,b){b[0]=a[0];b[1]=a[1];b[2]=a[2];return b};vec3.add=function(a,b,c){if(!c||a==c){a[0]+=b[0];a[1]+=b[1];a[2]+=b[2];return a}c[0]=a[0]+b[0];c[1]=a[1]+b[1];c[2]=a[2]+b[2];return c};**

**vec3.subtract=function(a,b,c){if(!c||a==c){a[0]-=b[0];a[1]-=b[1];a[2]-=b[2];return a}c[0]=a[0]-b[0];c[1]=a[1]-b[1];c[2]=a[2]-b[2];return c};**

**vec3.negate=function(a,b){b||(b=a);b[0]=-a[0];b[1]=-a[1];b[2]=-a[2];return b};**

**vec3.scale=function(a,b,c){if(!c||a==c){a[0]\*=b;a[1]\*=b;a[2]\*=b;return a}c[0]=a[0]\*b;c[1]=a[1]\*b;c[2]=a[2]\*b;return c};**

**vec3.normalize=function(a,b){b||(b=a);var c=a[0],d=a[1],e=a[2],g=Math.sqrt(c\*c+d\*d+e\*e);if(g){if(g==1){b[0]=c;b[1]=d;b[2]=e;return b}}else{b[0]=0;b[1]=0;b[2]=0;return b}g=1/g;b[0]=c\*g;b[1]=d\*g;b[2]=e\*g;return b};**

**vec3.cross=function(a,b,c){c||(c=a);var d=a[0],e=a[1];a=a[2];var g=b[0],f=b[1];b=b[2];c[0]=e\*b-a\*f;c[1]=a\*g-d\*b;c[2]=d\*f-e\*g;return c};**

**vec3.length=function(a){var b=a[0],c=a[1];a=a[2];return Math.sqrt(b\*b+c\*c+a\*a)};**

**vec3.dot=function(a,b){return a[0]\*b[0]+a[1]\*b[1]+a[2]\*b[2]};**

**vec3.direction=function(a,b,c){c||(c=a);var d=a[0]-b[0],e=a[1]-b[1];a=a[2]-b[2];b=Math.sqrt(d\*d+e\*e+a\*a);if(!b){c[0]=0;c[1]=0;c[2]=0;return c}b=1/b;c[0]=d\*b;c[1]=e\*b;c[2]=a\*b;return c};**

**vec3.lerp=function(a,b,c,d){d||(d=a);d[0]=a[0]+c\*(b[0]-a[0]);d[1]=a[1]+c\*(b[1]-a[1]);d[2]=a[2]+c\*(b[2]-a[2]);return d};**

**vec3.str=function(a){return"["+a[0]+", "+a[1]+", "+a[2]+"]"};**

**var mat3={};**

**mat3.create=function(a){var b=new glMatrixArrayType(9);if(a){b[0]=a[0];b[1]=a[1];b[2]=a[2];b[3]=a[3];b[4]=a[4];b[5]=a[5];b[6]=a[6];b[7]=a[7];b[8]=a[8];b[9]=a[9]}return b};**

**mat3.set=function(a,b){b[0]=a[0];b[1]=a[1];b[2]=a[2];b[3]=a[3];b[4]=a[4];b[5]=a[5];b[6]=a[6];b[7]=a[7];b[8]=a[8];return b};**

**mat3.identity=function(a){a[0]=1;a[1]=0;a[2]=0;a[3]=0;a[4]=1;a[5]=0;a[6]=0;a[7]=0;a[8]=1;return a};**

**mat3.transpose=function(a,b){if(!b||a==b){var c=a[1],d=a[2],e=a[5];a[1]=a[3];a[2]=a[6];a[3]=c;a[5]=a[7];a[6]=d;a[7]=e;return a}b[0]=a[0];b[1]=a[3];b[2]=a[6];b[3]=a[1];b[4]=a[4];b[5]=a[7];b[6]=a[2];b[7]=a[5];b[8]=a[8];return b};**

**mat3.toMat4=function(a,b){b||(b=mat4.create());b[0]=a[0];b[1]=a[1];b[2]=a[2];b[3]=0;b[4]=a[3];b[5]=a[4];b[6]=a[5];b[7]=0;b[8]=a[6];b[9]=a[7];b[10]=a[8];b[11]=0;b[12]=0;b[13]=0;b[14]=0;b[15]=1;return b};**

**mat3.str=function(a){return"["+a[0]+", "+a[1]+", "+a[2]+", "+a[3]+", "+a[4]+", "+a[5]+", "+a[6]+", "+a[7]+", "+a[8]+"]"};**

**var mat4={};**

**mat4.create=function(a){var b=new glMatrixArrayType(16);if(a){b[0]=a[0];b[1]=a[1];b[2]=a[2];b[3]=a[3];b[4]=a[4];b[5]=a[5];b[6]=a[6];b[7]=a[7];b[8]=a[8];b[9]=a[9];b[10]=a[10];b[11]=a[11];b[12]=a[12];b[13]=a[13];b[14]=a[14];b[15]=a[15]}return b};**

**mat4.set=function(a,b){b[0]=a[0];b[1]=a[1];b[2]=a[2];b[3]=a[3];b[4]=a[4];b[5]=a[5];b[6]=a[6];b[7]=a[7];b[8]=a[8];b[9]=a[9];b[10]=a[10];b[11]=a[11];b[12]=a[12];b[13]=a[13];b[14]=a[14];b[15]=a[15];return b};**

**mat4.identity=function(a){a[0]=1;a[1]=0;a[2]=0;a[3]=0;a[4]=0;a[5]=1;a[6]=0;a[7]=0;a[8]=0;a[9]=0;a[10]=1;a[11]=0;a[12]=0;a[13]=0;a[14]=0;a[15]=1;return a};**

**mat4.transpose=function(a,b){if(!b||a==b){var c=a[1],d=a[2],e=a[3],g=a[6],f=a[7],h=a[11];a[1]=a[4];a[2]=a[8];a[3]=a[12];a[4]=c;a[6]=a[9];a[7]=a[13];a[8]=d;a[9]=g;a[11]=a[14];a[12]=e;a[13]=f;a[14]=h;return a}b[0]=a[0];b[1]=a[4];b[2]=a[8];b[3]=a[12];b[4]=a[1];b[5]=a[5];b[6]=a[9];b[7]=a[13];b[8]=a[2];b[9]=a[6];b[10]=a[10];b[11]=a[14];b[12]=a[3];b[13]=a[7];b[14]=a[11];b[15]=a[15];return b};**

**mat4.determinant=function(a){var b=a[0],c=a[1],d=a[2],e=a[3],g=a[4],f=a[5],h=a[6],i=a[7],j=a[8],k=a[9],l=a[10],o=a[11],m=a[12],n=a[13],p=a[14];a=a[15];return m\*k\*h\*e-j\*n\*h\*e-m\*f\*l\*e+g\*n\*l\*e+j\*f\*p\*e-g\*k\*p\*e-m\*k\*d\*i+j\*n\*d\*i+m\*c\*l\*i-b\*n\*l\*i-j\*c\*p\*i+b\*k\*p\*i+m\*f\*d\*o-g\*n\*d\*o-m\*c\*h\*o+b\*n\*h\*o+g\*c\*p\*o-b\*f\*p\*o-j\*f\*d\*a+g\*k\*d\*a+j\*c\*h\*a-b\*k\*h\*a-g\*c\*l\*a+b\*f\*l\*a};**

**mat4.inverse=function(a,b){b||(b=a);var c=a[0],d=a[1],e=a[2],g=a[3],f=a[4],h=a[5],i=a[6],j=a[7],k=a[8],l=a[9],o=a[10],m=a[11],n=a[12],p=a[13],r=a[14],s=a[15],A=c\*h-d\*f,B=c\*i-e\*f,t=c\*j-g\*f,u=d\*i-e\*h,v=d\*j-g\*h,w=e\*j-g\*i,x=k\*p-l\*n,y=k\*r-o\*n,z=k\*s-m\*n,C=l\*r-o\*p,D=l\*s-m\*p,E=o\*s-m\*r,q=1/(A\*E-B\*D+t\*C+u\*z-v\*y+w\*x);b[0]=(h\*E-i\*D+j\*C)\*q;b[1]=(-d\*E+e\*D-g\*C)\*q;b[2]=(p\*w-r\*v+s\*u)\*q;b[3]=(-l\*w+o\*v-m\*u)\*q;b[4]=(-f\*E+i\*z-j\*y)\*q;b[5]=(c\*E-e\*z+g\*y)\*q;b[6]=(-n\*w+r\*t-s\*B)\*q;b[7]=(k\*w-o\*t+m\*B)\*q;b[8]=(f\*D-h\*z+j\*x)\*q;b[9]=(-c\*D+d\*z-g\*x)\*q;b[10]=(n\*v-p\*t+s\*A)\*q;b[11]=(-k\*v+l\*t-m\*A)\*q;b[12]=(-f\*C+h\*y-i\*x)\*q;b[13]=(c\*C-d\*y+e\*x)\*q;b[14]=(-n\*u+p\*B-r\*A)\*q;b[15]=(k\*u-l\*B+o\*A)\*q;return b};**

**mat4.toRotationMat=function(a,b){b||(b=mat4.create());b[0]=a[0];b[1]=a[1];b[2]=a[2];b[3]=a[3];b[4]=a[4];b[5]=a[5];b[6]=a[6];b[7]=a[7];b[8]=a[8];b[9]=a[9];b[10]=a[10];b[11]=a[11];b[12]=0;b[13]=0;b[14]=0;b[15]=1;return b};**

**mat4.toMat3=function(a,b){b||(b=mat3.create());b[0]=a[0];b[1]=a[1];b[2]=a[2];b[3]=a[4];b[4]=a[5];b[5]=a[6];b[6]=a[8];b[7]=a[9];b[8]=a[10];return b};**

**mat4.toInverseMat3=function(a,b){var c=a[0],d=a[1],e=a[2],g=a[4],f=a[5],h=a[6],i=a[8],j=a[9],k=a[10],l=k\*f-h\*j,o=-k\*g+h\*i,m=j\*g-f\*i,n=c\*l+d\*o+e\*m;if(!n)return null;n=1/n;b||(b=mat3.create());b[0]=l\*n;b[1]=(-k\*d+e\*j)\*n;b[2]=(h\*d-e\*f)\*n;b[3]=o\*n;b[4]=(k\*c-e\*i)\*n;b[5]=(-h\*c+e\*g)\*n;b[6]=m\*n;b[7]=(-j\*c+d\*i)\*n;b[8]=(f\*c-d\*g)\*n;return b};**

**mat4.multiply=function(a,b,c){c||(c=a);var d=a[0],e=a[1],g=a[2],f=a[3],h=a[4],i=a[5],j=a[6],k=a[7],l=a[8],o=a[9],m=a[10],n=a[11],p=a[12],r=a[13],s=a[14];a=a[15];var A=b[0],B=b[1],t=b[2],u=b[3],v=b[4],w=b[5],x=b[6],y=b[7],z=b[8],C=b[9],D=b[10],E=b[11],q=b[12],F=b[13],G=b[14];b=b[15];c[0]=A\*d+B\*h+t\*l+u\*p;c[1]=A\*e+B\*i+t\*o+u\*r;c[2]=A\*g+B\*j+t\*m+u\*s;c[3]=A\*f+B\*k+t\*n+u\*a;c[4]=v\*d+w\*h+x\*l+y\*p;c[5]=v\*e+w\*i+x\*o+y\*r;c[6]=v\*g+w\*j+x\*m+y\*s;c[7]=v\*f+w\*k+x\*n+y\*a;c[8]=z\*d+C\*h+D\*l+E\*p;c[9]=z\*e+C\*i+D\*o+E\*r;c[10]=z\*g+C\*j+D\*m+E\*s;c[11]=z\*f+C\*k+D\*n+E\*a;c[12]=q\*d+F\*h+G\*l+b\*p;c[13]=q\*e+F\*i+G\*o+b\*r;c[14]=q\*g+F\*j+G\*m+b\*s;c[15]=q\*f+F\*k+G\*n+b\*a;return c};**

**mat4.multiplyVec3=function(a,b,c){c||(c=b);var d=b[0],e=b[1];b=b[2];c[0]=a[0]\*d+a[4]\*e+a[8]\*b+a[12];c[1]=a[1]\*d+a[5]\*e+a[9]\*b+a[13];c[2]=a[2]\*d+a[6]\*e+a[10]\*b+a[14];return c};**

**mat4.multiplyVec4=function(a,b,c){c||(c=b);var d=b[0],e=b[1],g=b[2];b=b[3];c[0]=a[0]\*d+a[4]\*e+a[8]\*g+a[12]\*b;c[1]=a[1]\*d+a[5]\*e+a[9]\*g+a[13]\*b;c[2]=a[2]\*d+a[6]\*e+a[10]\*g+a[14]\*b;c[3]=a[3]\*d+a[7]\*e+a[11]\*g+a[15]\*b;return c};**

**mat4.translate=function(a,b,c){var d=b[0],e=b[1];b=b[2];if(!c||a==c){a[12]=a[0]\*d+a[4]\*e+a[8]\*b+a[12];a[13]=a[1]\*d+a[5]\*e+a[9]\*b+a[13];a[14]=a[2]\*d+a[6]\*e+a[10]\*b+a[14];a[15]=a[3]\*d+a[7]\*e+a[11]\*b+a[15];return a}var g=a[0],f=a[1],h=a[2],i=a[3],j=a[4],k=a[5],l=a[6],o=a[7],m=a[8],n=a[9],p=a[10],r=a[11];c[0]=g;c[1]=f;c[2]=h;c[3]=i;c[4]=j;c[5]=k;c[6]=l;c[7]=o;c[8]=m;c[9]=n;c[10]=p;c[11]=r;c[12]=g\*d+j\*e+m\*b+a[12];c[13]=f\*d+k\*e+n\*b+a[13];c[14]=h\*d+l\*e+p\*b+a[14];c[15]=i\*d+o\*e+r\*b+a[15];return c};**

**mat4.scale=function(a,b,c){var d=b[0],e=b[1];b=b[2];if(!c||a==c){a[0]\*=d;a[1]\*=d;a[2]\*=d;a[3]\*=d;a[4]\*=e;a[5]\*=e;a[6]\*=e;a[7]\*=e;a[8]\*=b;a[9]\*=b;a[10]\*=b;a[11]\*=b;return a}c[0]=a[0]\*d;c[1]=a[1]\*d;c[2]=a[2]\*d;c[3]=a[3]\*d;c[4]=a[4]\*e;c[5]=a[5]\*e;c[6]=a[6]\*e;c[7]=a[7]\*e;c[8]=a[8]\*b;c[9]=a[9]\*b;c[10]=a[10]\*b;c[11]=a[11]\*b;c[12]=a[12];c[13]=a[13];c[14]=a[14];c[15]=a[15];return c};**

**mat4.rotate=function(a,b,c,d){var e=c[0],g=c[1];c=c[2];var f=Math.sqrt(e\*e+g\*g+c\*c);if(!f)return null;if(f!=1){f=1/f;e\*=f;g\*=f;c\*=f}var h=Math.sin(b),i=Math.cos(b),j=1-i;b=a[0];f=a[1];var k=a[2],l=a[3],o=a[4],m=a[5],n=a[6],p=a[7],r=a[8],s=a[9],A=a[10],B=a[11],t=e\*e\*j+i,u=g\*e\*j+c\*h,v=c\*e\*j-g\*h,w=e\*g\*j-c\*h,x=g\*g\*j+i,y=c\*g\*j+e\*h,z=e\*c\*j+g\*h;e=g\*c\*j-e\*h;g=c\*c\*j+i;if(d){if(a!=d){d[12]=a[12];d[13]=a[13];d[14]=a[14];d[15]=a[15]}}else d=a;d[0]=b\*t+o\*u+r\*v;d[1]=f\*t+m\*u+s\*v;d[2]=k\*t+n\*u+A\*v;d[3]=l\*t+p\*u+B\*v;d[4]=b\*w+o\*x+r\*y;d[5]=f\*w+m\*x+s\*y;d[6]=k\*w+n\*x+A\*y;d[7]=l\*w+p\*x+B\*y;d[8]=b\*z+o\*e+r\*g;d[9]=f\*z+m\*e+s\*g;d[10]=k\*z+n\*e+A\*g;d[11]=l\*z+p\*e+B\*g;return d};**

**mat4.rotateX=function(a,b,c){var d=Math.sin(b);b=Math.cos(b);var e=a[4],g=a[5],f=a[6],h=a[7],i=a[8],j=a[9],k=a[10],l=a[11];if(c){if(a!=c){c[0]=a[0];c[1]=a[1];c[2]=a[2];c[3]=a[3];c[12]=a[12];c[13]=a[13];c[14]=a[14];c[15]=a[15]}}else c=a;c[4]=e\*b+i\*d;c[5]=g\*b+j\*d;c[6]=f\*b+k\*d;c[7]=h\*b+l\*d;c[8]=e\*-d+i\*b;c[9]=g\*-d+j\*b;c[10]=f\*-d+k\*b;c[11]=h\*-d+l\*b;return c};**

**mat4.rotateY=function(a,b,c){var d=Math.sin(b);b=Math.cos(b);var e=a[0],g=a[1],f=a[2],h=a[3],i=a[8],j=a[9],k=a[10],l=a[11];if(c){if(a!=c){c[4]=a[4];c[5]=a[5];c[6]=a[6];c[7]=a[7];c[12]=a[12];c[13]=a[13];c[14]=a[14];c[15]=a[15]}}else c=a;c[0]=e\*b+i\*-d;c[1]=g\*b+j\*-d;c[2]=f\*b+k\*-d;c[3]=h\*b+l\*-d;c[8]=e\*d+i\*b;c[9]=g\*d+j\*b;c[10]=f\*d+k\*b;c[11]=h\*d+l\*b;return c};**

**mat4.rotateZ=function(a,b,c){var d=Math.sin(b);b=Math.cos(b);var e=a[0],g=a[1],f=a[2],h=a[3],i=a[4],j=a[5],k=a[6],l=a[7];if(c){if(a!=c){c[8]=a[8];c[9]=a[9];c[10]=a[10];c[11]=a[11];c[12]=a[12];c[13]=a[13];c[14]=a[14];c[15]=a[15]}}else c=a;c[0]=e\*b+i\*d;c[1]=g\*b+j\*d;c[2]=f\*b+k\*d;c[3]=h\*b+l\*d;c[4]=e\*-d+i\*b;c[5]=g\*-d+j\*b;c[6]=f\*-d+k\*b;c[7]=h\*-d+l\*b;return c};**

**mat4.frustum=function(a,b,c,d,e,g,f){f||(f=mat4.create());var h=b-a,i=d-c,j=g-e;f[0]=e\*2/h;f[1]=0;f[2]=0;f[3]=0;f[4]=0;f[5]=e\*2/i;f[6]=0;f[7]=0;f[8]=(b+a)/h;f[9]=(d+c)/i;f[10]=-(g+e)/j;f[11]=-1;f[12]=0;f[13]=0;f[14]=-(g\*e\*2)/j;f[15]=0;return f};**

**mat4.perspective=function(a,b,c,d,e){a=c\*Math.tan(a\*Math.PI/360);b=a\*b;return mat4.frustum(-b,b,-a,a,c,d,e)};**

**mat4.ortho=function(a,b,c,d,e,g,f){f||(f=mat4.create());var h=b-a,i=d-c,j=g-e;f[0]=2/h;f[1]=0;f[2]=0;f[3]=0;f[4]=0;f[5]=2/i;f[6]=0;f[7]=0;f[8]=0;f[9]=0;f[10]=-2/j;f[11]=0;f[12]=-(a+b)/h;f[13]=-(d+c)/i;f[14]=-(g+e)/j;f[15]=1;return f};**

**mat4.lookAt=function(a,b,c,d){d||(d=mat4.create());var e=a[0],g=a[1];a=a[2];var f=c[0],h=c[1],i=c[2];c=b[1];var j=b[2];if(e==b[0]&&g==c&&a==j)return mat4.identity(d);var k,l,o,m;c=e-b[0];j=g-b[1];b=a-b[2];m=1/Math.sqrt(c\*c+j\*j+b\*b);c\*=m;j\*=m;b\*=m;k=h\*b-i\*j;i=i\*c-f\*b;f=f\*j-h\*c;if(m=Math.sqrt(k\*k+i\*i+f\*f)){m=1/m;k\*=m;i\*=m;f\*=m}else f=i=k=0;h=j\*f-b\*i;l=b\*k-c\*f;o=c\*i-j\*k;if(m=Math.sqrt(h\*h+l\*l+o\*o)){m=1/m;h\*=m;l\*=m;o\*=m}else o=l=h=0;d[0]=k;d[1]=h;d[2]=c;d[3]=0;d[4]=i;d[5]=l;d[6]=j;d[7]=0;d[8]=f;d[9]=o;d[10]=b;d[11]=0;d[12]=-(k\*e+i\*g+f\*a);d[13]=-(h\*e+l\*g+o\*a);d[14]=-(c\*e+j\*g+b\*a);d[15]=1;return d};**

**mat4.str=function(a){return"["+a[0]+", "+a[1]+", "+a[2]+", "+a[3]+", "+a[4]+", "+a[5]+", "+a[6]+", "+a[7]+", "+a[8]+", "+a[9]+", "+a[10]+", "+a[11]+", "+a[12]+", "+a[13]+", "+a[14]+", "+a[15]+"]"};**

**quat4={};**

**quat4.create=function(a){var b=new glMatrixArrayType(4);if(a){b[0]=a[0];b[1]=a[1];b[2]=a[2];b[3]=a[3]}return b};**

**quat4.set=function(a,b){b[0]=a[0];b[1]=a[1];b[2]=a[2];b[3]=a[3];return b};**

**quat4.calculateW=function(a,b){var c=a[0],d=a[1],e=a[2];if(!b||a==b){a[3]=-Math.sqrt(Math.abs(1-c\*c-d\*d-e\*e));return a}b[0]=c;b[1]=d;b[2]=e;b[3]=-Math.sqrt(Math.abs(1-c\*c-d\*d-e\*e));return b};**

**quat4.inverse=function(a,b){if(!b||a==b){a[0]\*=1;a[1]\*=1;a[2]\*=1;return a}b[0]=-a[0];b[1]=-a[1];b[2]=-a[2];b[3]=a[3];return b};**

**quat4.length=function(a){var b=a[0],c=a[1],d=a[2];a=a[3];return Math.sqrt(b\*b+c\*c+d\*d+a\*a)};**

**quat4.normalize=function(a,b){b||(b=a);var c=a[0],d=a[1],e=a[2],g=a[3],f=Math.sqrt(c\*c+d\*d+e\*e+g\*g);if(f==0){b[0]=0;b[1]=0;b[2]=0;b[3]=0;return b}f=1/f;b[0]=c\*f;b[1]=d\*f;b[2]=e\*f;b[3]=g\*f;return b};**

**quat4.multiply=function(a,b,c){c||(c=a);var d=a[0],e=a[1],g=a[2];a=a[3];var f=b[0],h=b[1],i=b[2];b=b[3];c[0]=d\*b+a\*f+e\*i-g\*h;c[1]=e\*b+a\*h+g\*f-d\*i;c[2]=g\*b+a\*i+d\*h-e\*f;c[3]=a\*b-d\*f-e\*h-g\*i;return c};**

**quat4.multiplyVec3=function(a,b,c){c||(c=b);var d=b[0],e=b[1],g=b[2];b=a[0];var f=a[1],h=a[2];a=a[3];var i=a\*d+f\*g-h\*e,j=a\*e+h\*d-b\*g,k=a\*g+b\*e-f\*d;d=-b\*d-f\*e-h\*g;c[0]=i\*a+d\*-b+j\*-h-k\*-f;c[1]=j\*a+d\*-f+k\*-b-i\*-h;c[2]=k\*a+d\*-h+i\*-f-j\*-b;return c};**

**quat4.toMat3=function(a,b){b||(b=mat3.create());var c=a[0],d=a[1],e=a[2],g=a[3],f=c+c,h=d+d,i=e+e,j=c\*f,k=c\*h;c=c\*i;var l=d\*h;d=d\*i;e=e\*i;f=g\*f;h=g\*h;g=g\*i;b[0]=1-(l+e);b[1]=k-g;b[2]=c+h;b[3]=k+g;b[4]=1-(j+e);b[5]=d-f;b[6]=c-h;b[7]=d+f;b[8]=1-(j+l);return b};**

**quat4.toMat4=function(a,b){b||(b=mat4.create());var c=a[0],d=a[1],e=a[2],g=a[3],f=c+c,h=d+d,i=e+e,j=c\*f,k=c\*h;c=c\*i;var l=d\*h;d=d\*i;e=e\*i;f=g\*f;h=g\*h;g=g\*i;b[0]=1-(l+e);b[1]=k-g;b[2]=c+h;b[3]=0;b[4]=k+g;b[5]=1-(j+e);b[6]=d-f;b[7]=0;b[8]=c-h;b[9]=d+f;b[10]=1-(j+l);b[11]=0;b[12]=0;b[13]=0;b[14]=0;b[15]=1;return b};**

**quat4.slerp=function(a,b,c,d){d||(d=a);var e=c;if(a[0]\*b[0]+a[1]\*b[1]+a[2]\*b[2]+a[3]\*b[3]<0)e=-1\*c;d[0]=1-c\*a[0]+e\*b[0];d[1]=1-c\*a[1]+e\*b[1];d[2]=1-c\*a[2]+e\*b[2];d[3]=1-c\*a[3]+e\*b[3];return d};**

**quat4.str=function(a){return"["+a[0]+", "+a[1]+", "+a[2]+", "+a[3]+"]"};**

**<!- hae02b.html -->**

**<html>**

**<head>**

**<title>Learning WebGL - lesson 2</title>**

**<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>**

**<script id="shader-fs" type="x-shader/x-fragment">**

 **precision mediump float;**

 **varying vec4 vColor;**

 **void main(void) {**

 **// interpolacao linear**

 **gl\_FragColor = vColor;**

 **}**

**</script>**

**<script id="shader-vs" type="x-shader/x-vertex">**

 **attribute vec3 aVertexPosition;**

 **attribute vec4 aVertexColor;**

 **uniform mat4 uMVMatrix;**

 **uniform mat4 uPMatrix;**

 **varying vec4 vColor;**

 **void main(void) {**

 **gl\_Position = uPMatrix \* uMVMatrix \* vec4(aVertexPosition, 1.0);**

 **vColor = aVertexColor;**

 **}**

**</script>**

**<script type="text/javascript">**

 **var gl;**

 **function initGL(canvas) {**

 **try {**

 **gl = canvas.getContext("experimental-webgl");**

 **gl.viewportWidth = canvas.width;**

 **gl.viewportHeight = canvas.height;**

 **} catch (e) {**

 **}**

 **if (!gl) {**

 **alert("Could not initialise WebGL, sorry :-(");**

 **}**

 **}**

 **function getShader(gl, id) {**

 **var shaderScript = document.getElementById(id);**

 **if (!shaderScript) {**

 **return null;**

 **}**

 **var str = "";**

 **var k = shaderScript.firstChild;**

 **while (k) {**

 **if (k.nodeType == 3) {**

 **str += k.textContent;**

 **}**

 **k = k.nextSibling;**

 **}**

 **var shader;**

 **if (shaderScript.type == "x-shader/x-fragment") {**

 **shader = gl.createShader(gl.FRAGMENT\_SHADER);**

 **} else if (shaderScript.type == "x-shader/x-vertex") {**

 **shader = gl.createShader(gl.VERTEX\_SHADER);**

 **} else {**

 **return null;**

 **}**

 **gl.shaderSource(shader, str);**

 **gl.compileShader(shader);**

 **if (!gl.getShaderParameter(shader, gl.COMPILE\_STATUS)) {**

 **alert(gl.getShaderInfoLog(shader));**

 **return null;**

 **}**

 **return shader;**

 **}**

 **var shaderProgram;**

 **function initShaders() {**

 **var fragmentShader = getShader(gl, "shader-fs");**

 **var vertexShader = getShader(gl, "shader-vs");**

 **shaderProgram = gl.createProgram();**

 **gl.attachShader(shaderProgram, vertexShader);**

 **gl.attachShader(shaderProgram, fragmentShader);**

 **gl.linkProgram(shaderProgram);**

 **if (!gl.getProgramParameter(shaderProgram, gl.LINK\_STATUS)) {**

 **alert("Could not initialise shaders");**

 **}**

 **gl.useProgram(shaderProgram);**

 **shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram, "aVertexPosition");**

 **gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);**

 **shaderProgram.vertexColorAttribute = gl.getAttribLocation(shaderProgram, "aVertexColor");**

 **gl.enableVertexAttribArray(shaderProgram.vertexColorAttribute);**

 **shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");**

 **shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");**

 **}**

 **var mvMatrix = mat4.create();**

 **var pMatrix = mat4.create();**

 **function setMatrixUniforms() {**

 **gl.uniformMatrix4fv(shaderProgram.pMatrixUniform, false, pMatrix);**

 **gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform, false, mvMatrix);**

 **}**

 **var triangleVertexPositionBuffer;**

 **var triangleVertexColorBuffer;**

 **var squareVertexPositionBuffer;**

 **var squareVertexColorBuffer;**

 **function initBuffers() {**

 **triangleVertexPositionBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexPositionBuffer);**

 **var vertices = [**

 **0.0, 2.0, 0.0,**

 **-1.0, -2.0, 0.0,**

 **1.0, -2.0, 0.0**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);**

 **triangleVertexPositionBuffer.itemSize = 3;**

 **triangleVertexPositionBuffer.numItems = 3;**

 **triangleVertexColorBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexColorBuffer);**

 **var colors = [**

 **0.0, 1.0, 1.0, 1.0,**

 **1.0, 0.0, 1.0, 1.0,**

 **1.0, 1.0, 0.0, 1.0**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(colors), gl.STATIC\_DRAW);**

 **triangleVertexColorBuffer.itemSize = 4;**

 **triangleVertexColorBuffer.numItems = 3;**

 **squareVertexPositionBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexPositionBuffer);**

 **vertices = [**

 **1.0, 2.0, 0.0,**

 **-1.0, 2.0, 0.0,**

 **1.0, -2.0, 0.0,**

 **-1.0, -2.0, 0.0**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);**

 **squareVertexPositionBuffer.itemSize = 3;**

 **squareVertexPositionBuffer.numItems = 4;**

 **squareVertexColorBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexColorBuffer);**

 **//colors = [];**

 **//for (var i=0; i < 4; i++) {**

 **// colors = colors.concat([0.5, 0.5, 1.0, 1.0]);**

 **//}**

 **colors = [**

 **0.0, 1.0, 1.0, 1.0,**

 **1.0, 0.0, 1.0, 1.0,**

 **1.0, 1.0, 0.0, 1.0,**

 **1.0, 0.5, 0.5, 1.0**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(colors), gl.STATIC\_DRAW);**

 **squareVertexColorBuffer.itemSize = 4;**

 **squareVertexColorBuffer.numItems = 4;**

 **}**

 **function drawScene() {**

 **gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);**

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

 **mat4.perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);**

 **mat4.identity(mvMatrix);**

 **mat4.translate(mvMatrix, [-1.5, 0.0, -7.0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute, triangleVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexColorBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexColorAttribute, triangleVertexColorBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **setMatrixUniforms();**

 **gl.drawArrays(gl.TRIANGLES, 0, triangleVertexPositionBuffer.numItems);**

 **mat4.translate(mvMatrix, [3.0, 0.0, 0.0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute, squareVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexColorBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexColorAttribute, squareVertexColorBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **setMatrixUniforms();**

 **gl.drawArrays(gl.TRIANGLE\_STRIP, 0, squareVertexPositionBuffer.numItems);**

 **}**

 **function webGLStart() {**

 **var canvas = document.getElementById("lesson02-canvas");**

 **initGL(canvas);**

 **initShaders();**

 **initBuffers();**

 **gl.clearColor(0.0, 0.0, 0.0, 1.0);**

 **gl.enable(gl.DEPTH\_TEST);**

 **drawScene();**

 **}**

**</script>**

**</head>**

**<body onload="webGLStart();">**

 **<canvas id="lesson02-canvas" style="border: none;" width="500" height="500"></canvas>**

**</body>**

**</html>**

****

**<!- hae03b.html -->**

**<html>**

**<head>**

**<title>Learning WebGL - lesson 3</title>**

**<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>**

**<script type="text/javascript" src="webgl-utils.js"></script>**

**<script id="shader-fs" type="x-shader/x-fragment">**

 **precision mediump float;**

 **varying vec4 vColor;**

 **void main(void) {**

 **gl\_FragColor = vColor;**

 **}**

**</script>**

**<script id="shader-vs" type="x-shader/x-vertex">**

 **attribute vec3 aVertexPosition;**

 **attribute vec4 aVertexColor;**

 **uniform mat4 uMVMatrix;**

 **uniform mat4 uPMatrix;**

 **varying vec4 vColor;**

 **void main(void) {**

 **gl\_Position = uPMatrix \* uMVMatrix \* vec4(aVertexPosition, 1.0);**

 **vColor = aVertexColor;**

 **}**

**</script>**

**<script type="text/javascript">**

 **var gl;**

 **function initGL(canvas) {**

 **try {**

 **gl = canvas.getContext("experimental-webgl");**

 **gl.viewportWidth = canvas.width;**

 **gl.viewportHeight = canvas.height;**

 **} catch (e) {**

 **}**

 **if (!gl) {**

 **alert("Could not initialise WebGL, sorry :-(");**

 **}**

 **}**

 **function getShader(gl, id) {**

 **var shaderScript = document.getElementById(id);**

 **if (!shaderScript) {**

 **return null;**

 **}**

 **var str = "";**

 **var k = shaderScript.firstChild;**

 **while (k) {**

 **if (k.nodeType == 3) {**

 **str += k.textContent;**

 **}**

 **k = k.nextSibling;**

 **}**

 **var shader;**

 **if (shaderScript.type == "x-shader/x-fragment") {**

 **shader = gl.createShader(gl.FRAGMENT\_SHADER);**

 **} else if (shaderScript.type == "x-shader/x-vertex") {**

 **shader = gl.createShader(gl.VERTEX\_SHADER);**

 **} else {**

 **return null;**

 **}**

 **gl.shaderSource(shader, str);**

 **gl.compileShader(shader);**

 **if (!gl.getShaderParameter(shader, gl.COMPILE\_STATUS)) {**

 **alert(gl.getShaderInfoLog(shader));**

 **return null;**

 **}**

 **return shader;**

 **}**

 **var shaderProgram;**

 **function initShaders() {**

 **var fragmentShader = getShader(gl, "shader-fs");**

 **var vertexShader = getShader(gl, "shader-vs");**

 **shaderProgram = gl.createProgram();**

 **gl.attachShader(shaderProgram, vertexShader);**

 **gl.attachShader(shaderProgram, fragmentShader);**

 **gl.linkProgram(shaderProgram);**

 **if (!gl.getProgramParameter(shaderProgram, gl.LINK\_STATUS)) {**

 **alert("Could not initialise shaders");**

 **}**

 **gl.useProgram(shaderProgram);**

 **shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram,**

 **"aVertexPosition");**

 **gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);**

 **shaderProgram.vertexColorAttribute = gl.getAttribLocation(shaderProgram,**

 **"aVertexColor");**

 **gl.enableVertexAttribArray(shaderProgram.vertexColorAttribute);**

 **shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");**

 **shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");**

 **}**

 **var mvMatrix = mat4.create();**

 **var mvMatrixStack = [];**

 **var pMatrix = mat4.create();**

 **function mvPushMatrix() {**

 **var copy = mat4.create();**

 **mat4.set(mvMatrix, copy);**

 **mvMatrixStack.push(copy);**

 **}**

 **function mvPopMatrix() {**

 **if (mvMatrixStack.length == 0) {**

 **throw "Invalid popMatrix!";**

 **}**

 **mvMatrix = mvMatrixStack.pop();**

 **}**

 **function setMatrixUniforms() {**

 **gl.uniformMatrix4fv(shaderProgram.pMatrixUniform, false, pMatrix);**

 **gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform, false, mvMatrix);**

 **}**

 **function degToRad(degrees) {**

 **return degrees \* Math.PI / 180;**

 **}**

 **var triangleVertexPositionBuffer;**

 **var triangleVertexColorBuffer;**

 **var squareVertexPositionBuffer;**

 **var squareVertexColorBuffer;**

 **function initBuffers() {**

 **triangleVertexPositionBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexPositionBuffer);**

 **var vertices = [**

 **0.0, 1.0, 0.0,**

 **-1.0, -1.0, 0.0,**

 **1.0, -1.0, 0.0**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);**

 **triangleVertexPositionBuffer.itemSize = 3;**

 **triangleVertexPositionBuffer.numItems = 3;**

 **triangleVertexColorBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexColorBuffer);**

 **var colors = [**

 **1.0, 0.0, 0.0, 1.0,**

 **0.0, 1.0, 0.0, 1.0,**

 **0.0, 0.0, 1.0, 1.0,**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(colors), gl.STATIC\_DRAW);**

 **triangleVertexColorBuffer.itemSize = 4;**

 **triangleVertexColorBuffer.numItems = 3;**

 **squareVertexPositionBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexPositionBuffer);**

 **vertices = [**

 **1.0, 1.0, 0.0,**

 **-1.0, 1.0, 0.0,**

 **1.0, -1.0, 0.0,**

 **-1.0, -1.0, 0.0**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);**

 **squareVertexPositionBuffer.itemSize = 3;**

 **squareVertexPositionBuffer.numItems = 4;**

 **squareVertexColorBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexColorBuffer);**

 **var colors = [**

 **1.0, 0.0, 0.0, 1.0,**

 **0.0, 1.0, 0.0, 1.0,**

 **0.0, 0.0, 1.0, 1.0,**

 **0.5, 0.0, 0.5, 1.0,**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(colors), gl.STATIC\_DRAW);**

 **squareVertexColorBuffer.itemSize = 4;**

 **squareVertexColorBuffer.numItems = 4;**

 **}**

 **var rTri = 0;**

 **var rSquare = 0;**

 **function drawScene() {**

 **gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);**

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

 **mat4.perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);**

 **mat4.identity(mvMatrix);**

 **//<<<<<<<<<<<<<<<<<**

 **mvPushMatrix();**

 **mat4.translate(mvMatrix, [-1.5, 0.0, -7.0]);**

 **mat4.rotate(mvMatrix, degToRad(rTri), [0, 1, 0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,**

 **triangleVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, triangleVertexColorBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexColorAttribute,**

 **triangleVertexColorBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **setMatrixUniforms();**

 **gl.drawArrays(gl.TRIANGLES, 0, triangleVertexPositionBuffer.numItems);**

 **mvPopMatrix();**

 **//<<<<<<<<<<<<<<<<<<<<<**

 **mvPushMatrix();**

 **mat4.translate(mvMatrix, [1.5, 0.0, -7.0]);**

 **mat4.rotate(mvMatrix, degToRad(rSquare), [0, -1, 0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,**

 **squareVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexColorBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexColorAttribute,**

 **squareVertexColorBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **setMatrixUniforms();**

 **gl.drawArrays(gl.TRIANGLE\_STRIP, 0, squareVertexPositionBuffer.numItems);**

 **mvPopMatrix();**

 **//<<<<<<<<<<<<<<<<<<<<<**

 **mvPushMatrix();**

 **mat4.translate(mvMatrix, [0.0, 0.0, -7.0]);**

 **mat4.rotate(mvMatrix, degToRad(1.4\*rSquare), [1, 0, 0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,**

 **squareVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, squareVertexColorBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexColorAttribute,**

 **squareVertexColorBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **setMatrixUniforms();**

 **gl.drawArrays(gl.TRIANGLE\_STRIP, 0, squareVertexPositionBuffer.numItems);**

 **mvPopMatrix();**

 **}**

 **var lastTime = 0;**

 **function animate() {**

 **var timeNow = new Date().getTime();**

 **if (lastTime != 0) {**

 **var elapsed = timeNow - lastTime;**

 **rTri += (90 \* elapsed) / 1000.0;**

 **rSquare += (75 \* elapsed) / 1000.0;**

 **}**

 **lastTime = timeNow;**

 **}**

 **function tick() {**

 **// Forma de fazer animacao independente do browser**

 **requestAnimFrame(tick); //chama tick novamente quando necessario**

 **drawScene();**

 **animate();**

 **}**

 **function webGLStart() {**

 **var canvas = document.getElementById("lesson03-canvas");**

 **initGL(canvas);**

 **initShaders()**

 **initBuffers();**

 **gl.clearColor(0.0, 0.0, 0.0, 1.0);**

 **gl.enable(gl.DEPTH\_TEST);**

 **tick(); //**

 **}**

**</script>**

**</head>**

**<body onload="webGLStart();">**

 **<canvas id="lesson03-canvas" width="500" height="500"></canvas>**

**</body>**

**</html>**

****

**<!-- hae04a.html -->**

**<html>**

**<head>**

**<title>Learning WebGL - lesson 4</title>**

**<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>**

**<script type="text/javascript" src="webgl-utils.js"></script>**

**<script id="shader-fs" type="x-shader/x-fragment">**

 **precision mediump float;**

 **varying vec4 vColor;**

 **void main(void) {**

 **gl\_FragColor = vColor;**

 **}**

**</script>**

**<script id="shader-vs" type="x-shader/x-vertex">**

 **attribute vec3 aVertexPosition;**

 **attribute vec4 aVertexColor;**

 **uniform mat4 uMVMatrix;**

 **uniform mat4 uPMatrix;**

 **varying vec4 vColor;**

 **void main(void) {**

 **gl\_Position = uPMatrix \* uMVMatrix \* vec4(aVertexPosition, 1.0);**

 **vColor = aVertexColor;**

 **}**

**</script>**

**<script type="text/javascript">**

 **var gl;**

 **function initGL(canvas) {**

 **try {**

 **gl = canvas.getContext("experimental-webgl");**

 **gl.viewportWidth = canvas.width;**

 **gl.viewportHeight = canvas.height;**

 **} catch (e) {**

 **}**

 **if (!gl) {**

 **alert("Could not initialise WebGL, sorry :-(");**

 **}**

 **}**

 **function getShader(gl, id) {**

 **var shaderScript = document.getElementById(id);**

 **if (!shaderScript) {**

 **return null;**

 **}**

 **var str = "";**

 **var k = shaderScript.firstChild;**

 **while (k) {**

 **if (k.nodeType == 3) {**

 **str += k.textContent;**

 **}**

 **k = k.nextSibling;**

 **}**

 **var shader;**

 **if (shaderScript.type == "x-shader/x-fragment") {**

 **shader = gl.createShader(gl.FRAGMENT\_SHADER);**

 **} else if (shaderScript.type == "x-shader/x-vertex") {**

 **shader = gl.createShader(gl.VERTEX\_SHADER);**

 **} else {**

 **return null;**

 **}**

 **gl.shaderSource(shader, str);**

 **gl.compileShader(shader);**

 **if (!gl.getShaderParameter(shader, gl.COMPILE\_STATUS)) {**

 **alert(gl.getShaderInfoLog(shader));**

 **return null;**

 **}**

 **return shader;**

 **}**

 **var shaderProgram;**

 **function initShaders() {**

 **var fragmentShader = getShader(gl, "shader-fs");**

 **var vertexShader = getShader(gl, "shader-vs");**

 **shaderProgram = gl.createProgram();**

 **gl.attachShader(shaderProgram, vertexShader);**

 **gl.attachShader(shaderProgram, fragmentShader);**

 **gl.linkProgram(shaderProgram);**

 **if (!gl.getProgramParameter(shaderProgram, gl.LINK\_STATUS)) {**

 **alert("Could not initialise shaders");**

 **}**

 **gl.useProgram(shaderProgram);**

 **shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram, "aVertexPosition");**

 **gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);**

 **shaderProgram.vertexColorAttribute = gl.getAttribLocation(shaderProgram, "aVertexColor");**

 **gl.enableVertexAttribArray(shaderProgram.vertexColorAttribute);**

 **shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");**

 **shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");**

 **}**

 **var mvMatrix = mat4.create();**

 **var mvMatrixStack = [];**

 **var pMatrix = mat4.create();**

 **function mvPushMatrix() {**

 **var copy = mat4.create();**

 **mat4.set(mvMatrix, copy);**

 **mvMatrixStack.push(copy);**

 **}**

 **function mvPopMatrix() {**

 **if (mvMatrixStack.length == 0) {**

 **throw "Invalid popMatrix!";**

 **}**

 **mvMatrix = mvMatrixStack.pop();**

 **}**

 **function setMatrixUniforms() {**

 **gl.uniformMatrix4fv(shaderProgram.pMatrixUniform, false, pMatrix);**

 **gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform, false, mvMatrix);**

 **}**

 **function degToRad(degrees) {**

 **return degrees \* Math.PI / 180;**

 **}**

 **var cubeVertexPositionBuffer;**

 **var cubeVertexColorBuffer;**

 **var cubeVertexIndexBuffer;**

 **function initBuffers() {**

 **cubeVertexPositionBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, cubeVertexPositionBuffer);**

 **vertices = [**

 **// Front face**

 **-1.0, -1.0, 1.0,**

 **1.0, -1.0, 1.0,**

 **1.0, 1.0, 1.0,**

 **-1.0, 1.0, 1.0,**

 **// Back face**

 **-1.0, -1.0, -1.0,**

 **-1.0, 1.0, -1.0,**

 **1.0, 1.0, -1.0,**

 **1.0, -1.0, -1.0,**

 **// Top face**

 **-1.0, 1.0, -1.0,**

 **-1.0, 1.0, 1.0,**

 **1.0, 1.0, 1.0,**

 **1.0, 1.0, -1.0,**

 **// Bottom face**

 **-1.0, -1.0, -1.0,**

 **1.0, -1.0, -1.0,**

 **1.0, -1.0, 1.0,**

 **-1.0, -1.0, 1.0,**

 **// Right face**

 **1.0, -1.0, -1.0,**

 **1.0, 1.0, -1.0,**

 **1.0, 1.0, 1.0,**

 **1.0, -1.0, 1.0,**

 **// Left face**

 **-1.0, -1.0, -1.0,**

 **-1.0, -1.0, 1.0,**

 **-1.0, 1.0, 1.0,**

 **-1.0, 1.0, -1.0**

 **];**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);**

 **cubeVertexPositionBuffer.itemSize = 3;**

 **cubeVertexPositionBuffer.numItems = 24;**

 **cubeVertexColorBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, cubeVertexColorBuffer);**

 **colors = [**

 **[1.0, 0.0, 0.0, 1.0], // Front face**

 **[1.0, 1.0, 0.0, 1.0], // Back face**

 **[0.0, 1.0, 0.0, 1.0], // Top face**

 **[1.0, 0.5, 0.5, 1.0], // Bottom face**

 **[1.0, 0.0, 1.0, 1.0], // Right face**

 **[0.0, 0.0, 1.0, 1.0] // Left face**

 **];**

 **var unpackedColors = [];**

 **for (var i in colors) {**

 **var color = colors[i];**

 **for (var j=0; j < 4; j++) {**

 **unpackedColors = unpackedColors.concat(color);**

 **}**

 **}**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(unpackedColors), gl.STATIC\_DRAW);**

 **cubeVertexColorBuffer.itemSize = 4;**

 **cubeVertexColorBuffer.numItems = 24;**

 **cubeVertexIndexBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ELEMENT\_ARRAY\_BUFFER, cubeVertexIndexBuffer);**

 **var cubeVertexIndices = [**

 **0, 1, 2, 0, 2, 3, // Front face**

 **4, 5, 6, 4, 6, 7, // Back face**

 **8, 9, 10, 8, 10, 11, // Top face**

 **12, 13, 14, 12, 14, 15, // Bottom face**

 **16, 17, 18, 16, 18, 19, // Right face**

 **20, 21, 22, 20, 22, 23 // Left face**

 **];**

 **gl.bufferData(gl.ELEMENT\_ARRAY\_BUFFER, new Uint16Array(cubeVertexIndices),**

 **gl.STATIC\_DRAW);**

 **cubeVertexIndexBuffer.itemSize = 1;**

 **cubeVertexIndexBuffer.numItems = 36;**

 **}**

 **var rCube = 0;**

 **function drawScene() {**

 **gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);**

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

 **mat4.perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);**

 **mat4.identity(mvMatrix);**

 **mat4.translate(mvMatrix, [0.0, 0.0, -8.0]);**

 **mvPushMatrix();**

 **mat4.rotate(mvMatrix, degToRad(rCube), [1, 1, 1]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, cubeVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,**

 **cubeVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, cubeVertexColorBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexColorAttribute,**

 **cubeVertexColorBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **gl.bindBuffer(gl.ELEMENT\_ARRAY\_BUFFER, cubeVertexIndexBuffer);**

 **setMatrixUniforms();**

 **gl.drawElements(gl.TRIANGLES, cubeVertexIndexBuffer.numItems, gl.UNSIGNED\_SHORT, 0);**

 **mvPopMatrix();**

 **}**

 **var lastTime = 0;**

 **function animate() {**

 **var timeNow = new Date().getTime();**

 **if (lastTime != 0) {**

 **var elapsed = timeNow - lastTime;**

 **rCube -= (75 \* elapsed) / 1000.0;**

 **}**

 **lastTime = timeNow;**

 **}**

 **function tick() {**

 **requestAnimFrame(tick);**

 **drawScene();**

 **animate();**

 **}**

 **function webGLStart() {**

 **var canvas = document.getElementById("lesson04-canvas");**

 **initGL(canvas);**

 **initShaders()**

 **initBuffers();**

 **gl.clearColor(0.0, 0.0, 0.0, 1.0);**

 **gl.enable(gl.DEPTH\_TEST);**

 **tick();**

 **}**

**</script>**

**</head>**

**<body onload="webGLStart();">**

 **<canvas id="lesson04-canvas" width="500" height="500"></canvas>**

**</body>**

**</html>**

****

**<!-- hae04d.html -->**

**<html>**

**<head>**

**<title>Learning WebGL - lesson 4</title>**

**<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>**

**<script type="text/javascript" src="webgl-utils.js"></script>**

**<script id="shader-fs" type="x-shader/x-fragment">**

 **precision mediump float;**

 **varying vec4 vColor;**

 **void main(void) {**

 **gl\_FragColor = vColor;**

 **}**

**</script>**

**<script id="shader-vs" type="x-shader/x-vertex">**

 **attribute vec3 aVertexPosition;**

 **attribute vec4 aVertexColor;**

 **uniform mat4 uMVMatrix;**

 **uniform mat4 uPMatrix;**

 **varying vec4 vColor;**

 **void main(void) {**

 **gl\_Position = uPMatrix \* uMVMatrix \* vec4(aVertexPosition, 1.0);**

 **vColor = aVertexColor;**

 **}**

**</script>**

**<script type="text/javascript">**

 **var gl;**

 **function initGL(canvas) {**

 **try {**

 **gl = canvas.getContext("experimental-webgl");**

 **gl.viewportWidth = canvas.width;**

 **gl.viewportHeight = canvas.height;**

 **} catch (e) {**

 **}**

 **if (!gl) {**

 **alert("Could not initialise WebGL, sorry :-(");**

 **}**

 **}**

 **function getShader(gl, id) {**

 **var shaderScript = document.getElementById(id);**

 **if (!shaderScript) {**

 **return null;**

 **}**

 **var str = "";**

 **var k = shaderScript.firstChild;**

 **while (k) {**

 **if (k.nodeType == 3) {**

 **str += k.textContent;**

 **}**

 **k = k.nextSibling;**

 **}**

 **var shader;**

 **if (shaderScript.type == "x-shader/x-fragment") {**

 **shader = gl.createShader(gl.FRAGMENT\_SHADER);**

 **} else if (shaderScript.type == "x-shader/x-vertex") {**

 **shader = gl.createShader(gl.VERTEX\_SHADER);**

 **} else {**

 **return null;**

 **}**

 **gl.shaderSource(shader, str);**

 **gl.compileShader(shader);**

 **if (!gl.getShaderParameter(shader, gl.COMPILE\_STATUS)) {**

 **alert(gl.getShaderInfoLog(shader));**

 **return null;**

 **}**

 **return shader;**

 **}**

 **var shaderProgram;**

 **function initShaders() {**

 **var fragmentShader = getShader(gl, "shader-fs");**

 **var vertexShader = getShader(gl, "shader-vs");**

 **shaderProgram = gl.createProgram();**

 **gl.attachShader(shaderProgram, vertexShader);**

 **gl.attachShader(shaderProgram, fragmentShader);**

 **gl.linkProgram(shaderProgram);**

 **if (!gl.getProgramParameter(shaderProgram, gl.LINK\_STATUS)) {**

 **alert("Could not initialise shaders");**

 **}**

 **gl.useProgram(shaderProgram);**

 **shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram,**

 **"aVertexPosition");**

 **gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);**

 **shaderProgram.vertexColorAttribute = gl.getAttribLocation(shaderProgram,**

 **"aVertexColor");**

 **gl.enableVertexAttribArray(shaderProgram.vertexColorAttribute);**

 **shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");**

 **shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");**

 **}**

 **var mvMatrix = mat4.create();**

 **var mvMatrixStack = [];**

 **var pMatrix = mat4.create();**

 **function mvPushMatrix() {**

 **var copy = mat4.create();**

 **mat4.set(mvMatrix, copy);**

 **mvMatrixStack.push(copy);**

 **}**

 **function mvPopMatrix() {**

 **if (mvMatrixStack.length == 0) {**

 **throw "Invalid popMatrix!";**

 **}**

 **mvMatrix = mvMatrixStack.pop();**

 **}**

 **function setMatrixUniforms() {**

 **gl.uniformMatrix4fv(shaderProgram.pMatrixUniform, false, pMatrix);**

 **gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform, false, mvMatrix);**

 **}**

 **function degToRad(degrees) {**

 **return degrees \* Math.PI / 180;**

 **}**

 **var cubeVertexPositionBuffer;**

 **var cubeVertexColorBuffer;**

 **var cubeVertexIndexBuffer;**

 **function initBuffers() {**

 **cubeVertexPositionBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, cubeVertexPositionBuffer);**

 **var M\_PI=3.1415926535897932384626433832795;**

 **vertices=[];**

 **var n=30; var r=2; var rr=1;**

 **var delta=2\*M\_PI/n;**

 **for (var i=0; i<n; i++) {**

 **for (var j=0; j<n; j++) {**

 **var alpha=i\*delta;**

 **var cosa=Math.cos(alpha);**

 **var sina=Math.sin(alpha);**

 **var beta=j\*delta;**

 **var x=r+rr\*Math.cos(beta); var x1=cosa\*x;**

 **var y1=sina\*x; var z1=rr\*Math.sin(beta);**

 **vertices = vertices.concat([x1,y1,z1]);**

 **var alpha=(i+1)\*delta;**

 **var cosa=Math.cos(alpha);**

 **var sina=Math.sin(alpha);**

 **var beta=j\*delta;**

 **var x=r+rr\*Math.cos(beta); var x1=cosa\*x;**

 **var y1=sina\*x; var z1=rr\*Math.sin(beta);**

 **vertices = vertices.concat([x1,y1,z1]);**

 **var alpha=(i+1)\*delta;**

 **var cosa=Math.cos(alpha);**

 **var sina=Math.sin(alpha);**

 **var beta=(j+1)\*delta;**

 **var x=r+rr\*Math.cos(beta); var x1=cosa\*x;**

 **var y1=sina\*x; var z1=rr\*Math.sin(beta);**

 **vertices = vertices.concat([x1,y1,z1]);**

 **var alpha=i\*delta;**

 **var cosa=Math.cos(alpha);**

 **var sina=Math.sin(alpha);**

 **var beta=(j+1)\*delta;**

 **var x=r+rr\*Math.cos(beta); var x1=cosa\*x;**

 **var y1=sina\*x; var z1=rr\*Math.sin(beta);**

 **vertices = vertices.concat([x1,y1,z1]);**

 **}**

 **}**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices), gl.STATIC\_DRAW);**

 **cubeVertexPositionBuffer.itemSize = 3;**

 **cubeVertexPositionBuffer.numItems = 4\*n\*n;**

 **cubeVertexColorBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, cubeVertexColorBuffer);**

 **var unpackedColors = [];**

 **var cor1=[0.0, 1.0, 1.0, 1.0];**

 **var cor2=[1.0, 1.0, 0.0, 1.0];**

 **var temp;**

 **for (var i=0; i<n; i++) {**

 **temp=cor1; cor1=cor2; cor2=temp;**

 **for (var j=0; j<n/2; j++) {**

 **unpackedColors = unpackedColors.concat(cor1);**

 **unpackedColors = unpackedColors.concat(cor1);**

 **unpackedColors = unpackedColors.concat(cor1);**

 **unpackedColors = unpackedColors.concat(cor1);**

 **unpackedColors = unpackedColors.concat(cor2);**

 **unpackedColors = unpackedColors.concat(cor2);**

 **unpackedColors = unpackedColors.concat(cor2);**

 **unpackedColors = unpackedColors.concat(cor2);**

 **}**

 **}**

 **gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(unpackedColors), gl.STATIC\_DRAW);**

 **cubeVertexColorBuffer.itemSize = 4;**

 **cubeVertexColorBuffer.numItems = 24;**

 **cubeVertexIndexBuffer = gl.createBuffer();**

 **gl.bindBuffer(gl.ELEMENT\_ARRAY\_BUFFER, cubeVertexIndexBuffer);**

 **var cubeVertexIndices = [];**

 **for (var i=0; i<cubeVertexPositionBuffer.numItems; i=i+4) {**

 **cubeVertexIndices = cubeVertexIndices.concat([i,i+1,i+2]);**

 **cubeVertexIndices = cubeVertexIndices.concat([i,i+2,i+3]);**

 **}**

 **gl.bufferData(gl.ELEMENT\_ARRAY\_BUFFER, new Uint16Array(cubeVertexIndices),**

 **gl.STATIC\_DRAW);**

 **cubeVertexIndexBuffer.itemSize = 1;**

 **cubeVertexIndexBuffer.numItems = 6\*cubeVertexPositionBuffer.numItems/4;**

 **}**

 **var rCube = 0;**

 **function drawScene() {**

 **gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);**

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

 **mat4.perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);**

 **mat4.identity(mvMatrix);**

 **mat4.translate(mvMatrix, [0.0, 0.0, -8.0]);**

 **mvPushMatrix();**

 **mat4.rotate(mvMatrix, degToRad(rCube), [1, 0.5, 0]);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, cubeVertexPositionBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute, cubeVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **gl.bindBuffer(gl.ARRAY\_BUFFER, cubeVertexColorBuffer);**

 **gl.vertexAttribPointer(shaderProgram.vertexColorAttribute, cubeVertexColorBuffer.itemSize, gl.FLOAT, false, 0, 0);**

 **gl.bindBuffer(gl.ELEMENT\_ARRAY\_BUFFER, cubeVertexIndexBuffer);**

 **setMatrixUniforms();**

 **gl.drawElements(gl.TRIANGLES, cubeVertexIndexBuffer.numItems, gl.UNSIGNED\_SHORT, 0);**

 **mvPopMatrix();**

 **}**

 **var lastTime = 0;**

 **function animate() {**

 **var timeNow = new Date().getTime();**

 **if (lastTime != 0) {**

 **var elapsed = timeNow - lastTime;**

 **rCube -= (75 \* elapsed) / 1000.0;**

 **}**

 **lastTime = timeNow;**

 **}**

 **function tick() {**

 **requestAnimFrame(tick);**

 **drawScene();**

 **animate();**

 **}**

 **function webGLStart() {**

 **var canvas = document.getElementById("lesson04-canvas");**

 **initGL(canvas);**

 **initShaders()**

 **initBuffers();**

 **gl.clearColor(0.0, 0.0, 0.0, 1.0);**

 **gl.enable(gl.DEPTH\_TEST);**

 **tick();**

 **}**

**</script>**

**</head>**

**<body onload="webGLStart();">**

 **<canvas id="lesson04-canvas" style="border: none;" width="500" height="500"></canvas>**

**</body>**

**</html>**

****