1 – Triangle → 3D square

To do

Add perspective projection and model matrices

Change vec2 to vec3

Change assignment to gl_Position

```
<script id="vertex" type="x-shader">
  mat4 uMVMatrix = mat4 (1.0, 0.0, 0.0, 0.0,
                         0.0, 1.0, 0.0, 0.0,
                         0.0, 0.0, 1.0, 0.0,
                         0.0, 0.0, -3.333, 1.0 );

  mat4 uPMatrix = mat4 (2.41421, 0.0, 0.0, 0.0,
                        0.0, 2.41421, 0.0, 0.0,
                        0.0, 0.0, -1.002002, -1.0,
                        0.0, 0.0, -0.2002002, 0.0 );

  attribute vec3 aVertexPosition;

  void main() {
    gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
  }
</script>
```

Make triangle 3d (add z ordinate)

Add second triangle to make a square

Add co-ordinates (initGeometry)

Increase itemsize (initGeometry)

```
function initGeometry() {
  var vertices = new Float32Array([
  //|------ 0 ------| |------ 1 ------| |------ 2 ------|
  //|  x    y    z  | |  x    y    z  | |  x    y    z  |
      0.5, 0.5, 0.5,    0.5,-0.5, 0.5,   -0.5,-0.5, 0.5,  // triangle 1
      0.5, 0.5, 0.5,   -0.5, 0.5, 0.5,   -0.5,-0.5, 0.5,  // triangle 2
      0.5, 0.5,-0.5,    0.5,-0.5,-0.5,   -0.5,-0.5,-0.5,  // triangle 3
    ]);

    cubeVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexPositionBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW);

    cubeVertexPositionBuffer.itemSize = 3;
```

2 – 3D Square → Coloured Cube

To do

Make cube (add faces)

(connect faces)

(preview with lines)

define vertices, faces

Add colour to faces

```
<script id="vertex" type="x-shader">
  ...
  attribute vec3 aVertexPosition;
  attribute vec4 aVertexColor;

  varying vec4 vColor;

  void main() {
    gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
    vColor = aVertexColor;
  }
</script>

<script id="fragment" type="x-shader">
```

```
    ...
    gl_FragColor = vColor;
  }
</script>

<script type="text/javascript">
  var cubeVertexPositionBuffer;
  var cubeVertexColorBuffer;
  var cubeVertexIndexBuffer;

  function initShaderProgram() {
    ...
    gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);
    shaderProgram.vertexColorAttribute = gl.getAttribLocation(shaderProgram, "aVertexColor");
    gl.enableVertexAttribArray(shaderProgram.vertexColorAttribute);
  }

  function initGeometry() {
    var vertices = new Float32Array([
            // Front face
            -0.5, -0.5,  0.5,
             0.5, -0.5,  0.5,
             0.5,  0.5,  0.5,
            -0.5,  0.5,  0.5,

            // Back face
            -0.5, -0.5, -0.5,
            -0.5,  0.5, -0.5,
             0.5,  0.5, -0.5,
             0.5, -0.5, -0.5,

            // Top face
            -0.5,  0.5, -0.5,
            -0.5,  0.5,  0.5,
             0.5,  0.5,  0.5,
             0.5,  0.5, -0.5,

            // Bottom face
            -0.5, -0.5, -0.5,
             0.5, -0.5, -0.5,
             0.5, -0.5,  0.5,
            -0.5, -0.5,  0.5,

            // Right face
             0.5, -0.5, -0.5,
             0.5,  0.5, -0.5,
             0.5,  0.5,  0.5,
             0.5, -0.5,  0.5,

            // Left face
            -0.5, -0.5, -0.5,
            -0.5, -0.5,  0.5,
            -0.5,  0.5,  0.5,
            -0.5,  0.5, -0.5
      ]);
    cubeVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexPositionBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW);
    cubeVertexPositionBuffer.itemSize = 3;
    cubeVertexPositionBuffer.numItems = vertices.length / cubeVertexPositionBuffer.itemSize;

    cubeVertexColorBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexColorBuffer);
    colors = [
        [0.0, 1.0, 1.0, 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, 0.0, 1.0, 1.0], // Bottom face
        [1.0, 0.0, 0.0, 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;

    var indices = new Uint16Array([
        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
    ]);
    cubeVertexIndexBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, cubeVertexIndexBuffer);
    gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, indices, gl.STATIC_DRAW);
    cubeVertexIndexBuffer.itemSize = 1;
    cubeVertexIndexBuffer.numItems = 36;
  }


  function draw() {
    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);

    gl.clearColor(0, 0.5, 0, 1);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    gl.enable(gl.DEPTH_TEST);

    gl.drawElements(gl.TRIANGLES, cubeVertexIndexBuffer.numItems, gl.UNSIGNED_SHORT, 0);
  }
```

3 – Coloured Cube → Rotated Cube

To do

Add Rotation

```
<script type="text/javascript">
  ...
  var mvMatrix = mat4.create();
  var mvMatrixStack = [];
  var pMatrix = mat4.create();

  var rotationMatrix = mat4.create();
  mat4.identity(rotationMatrix);


  function draw() {
    mat4.perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);
    mat4.identity(mvMatrix);
    mat4.translate(mvMatrix, [0.0, 0.0, -4.0]);
    mat4.rotate(mvMatrix, 3, [1, 1, 1]);
    mat4.multiply(mvMatrix, rotationMatrix);

    gl.uniformMatrix4fv(shaderProgram.pMatrixUniform, false, pMatrix);
    gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform, false, mvMatrix);
```

4 – Rotated Cube → Rotatable Cube

To do

Input keyboard

mouse

touch

Add flick

Add texture

```
<script type="text/javascript">
  ...
  var mouseDown = false;
  var lastMouseX = null;
  var lastMouseY = null;
  var rvelX = 0;
  var rvelY = 0;

  function degToRad(degrees) {
    return degrees * Math.PI / 180;
  }

  function handleMouseDown(event) {
    mouseDown = true;
    lastMouseX = event.clientX;
    lastMouseY = event.clientY;
  }

  function handleMouseUp(event) {
    mouseDown = false;
  }

  function handleMouseMove(event) {
    if (!mouseDown) {
      return;
    }
    var newX = event.clientX;
    var newY = event.clientY;

    var fudgefactor = 2;
    var deltaX = newX - lastMouseX;
    var deltaY = newY - lastMouseY;
    lastMouseX = newX
    lastMouseY = newY;

    rvelX = deltaX / fudgefactor;
    rvelY = deltaY / fudgefactor;
  }


  function animate() {
    var newRotationMatrix = mat4.create();

    mat4.identity(newRotationMatrix);
    mat4.rotate(newRotationMatrix, degToRad(rvelX), [0, 1, 0]);
    mat4.rotate(newRotationMatrix, degToRad(rvelY), [1, 0, 0]);
    mat4.multiply(newRotationMatrix, rotationMatrix, rotationMatrix);

    rvelX = rvelX / 1.08;
    if(Math.abs(rvelX) < 0.001) rvelX = 0;
    rvelY = rvelY / 1.1;
    if(Math.abs(rvelY) < 0.001) rvelY = 0;
  }
```

5 – Rotatable Cube → Dice ???

```
<script id="shader-fs" type="x-shader/x-fragment">
  precision mediump float;

  varying vec2 vTextureCoord;

  uniform sampler2D uSampler;

  void main(void) {
      gl_FragColor = texture2D(uSampler, vec2(vTextureCoord.s, vTextureCoord.t));
  }
</script>

<script id="shader-vs" type="x-shader/x-vertex">
  attribute vec3 aVertexPosition;
  attribute vec2 aTextureCoord;

  uniform mat4 uMVMatrix;
  uniform mat4 uPMatrix;

  varying vec2 vTextureCoord;

  void main(void) {
      gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
      vTextureCoord = aTextureCoord;
  }
</script>

<script type="text/javascript">
  var neheTexture;
  ...
  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.textureCoordAttribute = gl.getAttribLocation(shaderProgram, "aTextureCoord");
      gl.enableVertexAttribArray(shaderProgram.textureCoordAttribute);

      shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");
      shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");
      shaderProgram.samplerUniform = gl.getUniformLocation(shaderProgram, "uSampler");
  }

  function handleLoadedTexture(texture) {
      gl.bindTexture(gl.TEXTURE_2D, texture);
      gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);
      gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, texture.image);
      gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
      gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
      gl.bindTexture(gl.TEXTURE_2D, null);
  }

  function initTexture() {
      neheTexture = gl.createTexture();
      neheTexture.image = new Image();
      neheTexture.image.onload = function () {
```

```
            handleLoadedTexture(neheTexture)
        }

        neheTexture.image.src = "nehe.gif";
    }

    function initBuffers() {
        ...
        cubeVertexTextureCoordBuffer = gl.createBuffer();
        gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexTextureCoordBuffer);
        var textureCoords = [
          // Front face
          0.0, 0.0,
          1.0, 0.0,
          1.0, 1.0,
          0.0, 1.0,

          // Back face
          1.0, 0.0,
          1.0, 1.0,
          0.0, 1.0,
          0.0, 0.0,

          // Top face
          0.0, 1.0,
          0.0, 0.0,
          1.0, 0.0,
          1.0, 1.0,

          // Bottom face
          1.0, 1.0,
          0.0, 1.0,
          0.0, 0.0,
          1.0, 0.0,

          // Right face
          1.0, 0.0,
          1.0, 1.0,
          0.0, 1.0,
          0.0, 0.0,

          // Left face
          0.0, 0.0,
          1.0, 0.0,
          1.0, 1.0,
          0.0, 1.0,
        ];
        gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(textureCoords), gl.STATIC_DRAW);
        cubeVertexTextureCoordBuffer.itemSize = 2;
        cubeVertexTextureCoordBuffer.numItems = 24;
        ...
    }

    function drawScene() {
        ...
        gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexTextureCoordBuffer);
        gl.vertexAttribPointer(shaderProgram.textureCoordAttribute, cubeVertexTextureCoordBuffer.itemSize,
gl.FLOAT, false, 0, 0);

        gl.activeTexture(gl.TEXTURE0);
        gl.bindTexture(gl.TEXTURE_2D, neheTexture);
        gl.uniform1i(shaderProgram.samplerUniform, 0);
        ...
    }

    function webGLStart() {
        ...
        initTexture();
        ...
    }
</script>
```