```
<!--  ************************************************************
GF begins rationalization 27mar14
Leans heavily on the Jones-MacKenzie gl.wannabe.
But it looks like the graphics (labelled with the term "shader")
is either built here (to replace what? parts of GLUT? Xwindow? )
or merely extend what's in Jones-MacKenzie?
************************************************************ -->
<!-- Zach Reizner translated oc1 to .js using Jones-MacKenzie minified -->
<!doctype html>
<html>
  <head>
    <script type="text/javascript">
<title>Octahedron</title>
</head>
<body>
    <canvas id="the-canvas" style="border: 1px dotted black"></canvas>
    <script type="text/javascript">

//ZR: Some cross-browser issues. Kindly skip over this function.
(function() {
  var requestAnimationFrame = window.requestAnimationFrame
      || window.mozRequestAnimationFrame
      || window.webkitRequestAnimationFrame
      || window.msRequestAnimationFrame;
  window.requestAnimationFrame = requestAnimationFrame;
})();

var MouseX = 200;  //restore the variable size window
var MouseY = 200;  //and put the bullseye back where it belongs

//ZR: More cross browser rubish. Honestly not worth learning about.
function relativeMouse(e, element) {
  if (e.pageX || e.pageY) {
    MouseX = e.pageX;
    MouseY = e.pageY;
  }
  else {
    MouseX = e.clientX + document.body.scrollLeft +
document.documentElement.scrollLeft;
    MouseY = e.clientY + document.body.scrollTop +
document.documentElement.scrollTop;
  }
  MouseX -= element.offsetLeft;
  MouseY -= element.offsetTop;
}
```

```
//ZR: Some OpenGL utility functions
/***************** gl. uses Jones-MacKenzie library  */
//ZR: Creates a GPU BUFFER for the array of data given.

function createBuffer(data)
{
    var buffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(data), gl.STATIC_DRAW);
    gl.bindBuffer(gl.ARRAY_BUFFER, null);
    return buffer;
}

function createShader(source, type) {
    var shader = gl.createShader(type);
    gl.shaderSource(shader, source);
    gl.compileShader(shader);
    if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS))
    {
        var errorMessage = gl.getShaderInfoLog(shader);
        console.error( errorMessage);
        return null;
    }
    return shader;
}

function createShaderProgram(vertexShaderSource, fragmentShaderSource) {
  var vertexShader = createShader(vertexShaderSource, gl.VERTEX_SHADER)
  var fragmentShader = createShader(fragmentShaderSource,
gl.FRAGMENT_SHADER);
  var shaderProgram = gl.createProgram();
  gl.attachShader(shaderProgram, vertexShader);
  gl.attachShader(shaderProgram, fragmentShader);
  gl.linkProgram(shaderProgram);
  if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) {
    console.error("Shaders failed to link");
  }
  return shaderProgram;
}

//ZR:  vertices for the octahedron
var vv = [
  /* positions*/  0.0, 1.0, 0.0, /* colors */ 0, 1, 0, // 1
  /* positions*/  1.0, 0.0, 0.0, /* colors */ 0, 0, 1, // 0
  /* positions*/  0.0, 0.0,-1.0, /* colors */ 0, 1, 1, // 5
  /* positions*/ -1.0, 0.0, 0.0, /* colors */ 1, 1, 0, // 3
```

```
  /* positions*/  0.0, 0.0, 1.0, /* colors */ 1, 0, 0, // 2
  /* positions*/  1.0, 0.0, 0.0, /* colors */ 0, 0, 1, // 0
  /* positions*/  0.0,-1.0, 0.0, /* colors */ 1, 0, 1, // 4
  /* positions*/  1.0, 0.0, 0.0, /* colors */ 0, 0, 1, // 0
  /* positions*/  0.0, 0.0,-1.0, /* colors */ 0, 1, 1, // 5
  /* positions*/ -1.0, 0.0, 0.0, /* colors */ 1, 1, 0, // 3
  /* positions*/  0.0, 0.0, 1.0, /* colors */ 1, 0, 0, // 2
  /* positions*/  1.0, 0.0, 0.0, /* colors */ 0, 0, 1, // 0
  //This are the original vertex/color array from oc1.c
  // and ZR says "they are not necessary", meaning what?
  /* positions*/  1.0, 0.0, 0.0, /* colors */ 0, 0, 1,
  /* positions*/  0.0, 1.0, 0.0, /* colors */ 0, 1, 0,
  /* positions*/  0.0, 0.0, 1.0, /* colors */ 1, 0, 0,
  /* positions*/ -1.0, 0.0, 0.0, /* colors */ 1, 1, 0,
  /* positions*/  0.0,-1.0, 0.0, /* colors */ 1, 0, 1,
  /* positions*/  0.0, 0.0,-1.0, /* colors */ 0, 1, 1
]

// ZR: Shader source code //what to the backslashes do?

var octVertexShaderSource = "\
uniform mat4 Affine;                                    \
                                                        \
attribute vec3 iPosition;                               \
attribute vec3 iColor;                                  \
                                                        \
varying vec3 oColor;                                    \
                                                        \
void main(void) {                                       \
    gl_Position = Affine * vec4(iPosition, 1);          \
    oColor = iColor;                                    \
}                                                       \
"

var octFragmentShaderSource = "\
precision mediump float;            \
                                    \
varying vec3 oColor;                \
                                    \
void main(void) {                   \
    gl_FragColor = vec4(oColor, 1); \
}                                   \
"

/************************RETURN TO FAMILIAR ********************/
//ZR:  Grab the canvas and context from the document. We draw with the
```

```
context.
var theCanvas = document.getElementById("the-canvas");

theCanvas.addEventListener("mousemove",
          function(event){ relativeMouse(event, theCanvas);});
theCanvas.width = 400;
theCanvas.height = 400;
var gl = theCanvas.getContext("experimental-webgl");

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

/***************** first use of preparations above  *******/
//ZR: Create a GPU buffer of vertices for the octahedron
var octVertexBuffer = createBuffer(vv);

//ZR: Get the shader and the indexes for setting variables
var shaderProgram = createShaderProgram(octVertexShaderSource,
                    octFragmentShaderSource);
var affIndex = gl.getUniformLocation(shaderProgram, "Affine");
var positionIndex = gl.getAttribLocation(shaderProgram, "iPosition");
var colorIndex = gl.getAttribLocation(shaderProgram, "iColor");

function draw() {
  //ZR: Ask the browser to call this function next chance it gets
  requestAnimationFrame(draw);

  //ZR: Update the affine matrix
  var dx = (MouseX - 200) / 1024;
  var dy = (MouseY - 200) / 1024;
  mat4.rotate(aff, aff, dx, [ 0.0, 1.0, 0.0]); /* (angle,axis) of rotation
*/
  mat4.rotate(aff, aff, dy, [-1.0, 0.0, 0.0]); /* (angle,axis) of rotation
*/

//Wy is there no gl. in front of mat4?

  //ZR: This part is unchanged from OpenGL
  gl.viewport(0, 0, theCanvas.width, theCanvas.height);
  gl.clearColor(0, 0, 0, 1);
  gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
  gl.enable(gl.DEPTH_TEST);

  // Tell OpenGL which shader to use
  gl.useProgram(shaderProgram);
```

```
  // Upload the View and Projection Matrix to the shader
  gl.uniformMatrix4fv(affIndex, false, aff);

  gl.bindBuffer(gl.ARRAY_BUFFER, octVertexBuffer);
  // Describe to OpenGL how the vertices are laid out.
  gl.enableVertexAttribArray(positionIndex);
  gl.vertexAttribPointer(positionIndex, 3, gl.FLOAT, false, 24, 0);

  gl.enableVertexAttribArray(colorIndex);
  gl.vertexAttribPointer(colorIndex, 3, gl.FLOAT, false, 24, 12);

  // These draw two triangle fans that form the octahedron
  gl.drawArrays(gl.TRIANGLE_FAN, 0, 6);
  gl.drawArrays(gl.TRIANGLE_FAN, 6, 6);

  // Cleanup after ourselves
  gl.bindBuffer(gl.ARRAY_BUFFER, null);
  gl.disableVertexAttribArray(positionIndex);
  gl.disableVertexAttribArray(colorIndex);
}

// Now that everything is defined, start the rendering loop.
draw();

    </script>
  </body>
</html>
```