Lab 18

In this lab, we will start with code that is based on the Key for Lab 17, then we will render a 3D tetrahedron, including the 4 buttons to choose the axis of rotation and toggling rotation on/off.

Toggle variable:

```
var flag = true;
```

Need to add vertex definitions for the 4 triangles used to form a tetrahedron.

```
// triangles to form a tetrahedron
    points=[
    vec4(  0.00 ,  0.50 ,  0.00 , 1.0 ),
    vec4( -0.50 , -0.50 ,  0.50 , 1.0 ),
    vec4(  0.50 , -0.50 ,  0.50 , 1.0 ),
    vec4(  0.50 , -0.50 , -0.50 , 1.0 )
    ];
```

```
Keep the triangular tex coordinates from Lab 17 KEY
```

```
Create a vertexColors array of 4 different colors, one for each face
of the tetrahedron
```

```
var vertexColors = [

    vec4(0.0, 0.0, 1.0, 1.0),  // blue
    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
];
```

Create two functions, triangle and colorTetra, based on the quad and colorCube functions used in previous labs, to define the four triangular faces of the tetrahedron. Be careful with the order of the vertex definitions for each triangle, and note that the triangle number determines the color of the face.

```
function triangle (a,b,c,triNum)
{
   positionsArray.push(points[a]);
   colorsArray.push(vertexColors[triNum]);
   texCoordsArray.push(texCoord[0]);

   positionsArray.push(points[b]);
   colorsArray.push(vertexColors[triNum]);
   texCoordsArray.push(texCoord[1]);

   positionsArray.push(points[c]);
   colorsArray.push(vertexColors[triNum]);
   texCoordsArray.push(texCoord[2]);
}

function colorTetra()
{
    triangle(0,1,2 ,0);
    triangle(0,2,3 ,1);
    triangle(0,3,1 ,2);
    triangle(1,3,2 ,3);
}
```

Add the code for rotations, again similar to the previous Color Cube labs, and theta can be all 45 degrees initiall:

```
var xAxis = 0;
var yAxis = 1;
var zAxis = 2;
var axis = xAxis;
var theta = vec3(45.0, 45.0, 45.0);
var thetaLoc;
```

Add the code for the uniform variable for theta, the angle of rotation and for the button listeners to toggle rotation and choose an axis of rotation.

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

Add the code that increments the angle theta before rendering:

```
    if(flag) theta[axis] += 2.0;
    gl.uniform3fv(thetaLoc, theta);
```

===

Add the code for the four buttons:

```
<button id = "ButtonX">Rotate X</button>
<button id = "ButtonY">Rotate Y</button>
<button id = "ButtonZ">Rotate Z</button>
<button id = "ButtonT">Toggle Rotation</button>
```

Add the code for the uniform variable uTheta in the vertex shader:

```
uniform vec3 uTheta;
```

Add the code for the rotation matrices and calculating position:

```
    // Compute the sines and cosines of theta for each of
    //   the three axes in one computation.
    vec3 angles = radians(uTheta);
    vec3 c = cos(angles);
    vec3 s = sin(angles);
    // Remember: these matrices are column-major
    mat4 rx = mat4(1.0, 0.0, 0.0, 0.0,
               0.0, c.x, s.x, 0.0,
               0.0, -s.x, c.x, 0.0,
               0.0, 0.0, 0.0, 1.0 );
    mat4 ry = mat4(c.y, 0.0, -s.y, 0.0,
               0.0, 1.0, 0.0, 0.0,
               s.y, 0.0, c.y, 0.0,
               0.0, 0.0, 0.0, 1.0);
    mat4 rz = mat4(c.z, s.z, 0.0, 0.0,
               -s.z, c.z, 0.0, 0.0,
               0.0,  0.0, 1.0, 0.0,
```

```
                0.0,  0.0, 0.0, 1.0);
  gl_Position = rz * ry * rx * aPosition;
  gl_Position.z = -gl_Position.z;
```

And lastly, don't forget to turn on depth testing:

```
    gl.enable(gl.DEPTH_TEST);

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