```
var canvas;
var gl;

var numVertices  = 36;

var pointsArray = [];
var normalsArray = [];

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 )
    ];

var lightPosition = vec4(1.0, 1.0, 1.0, 0.0 );
var lightAmbient = vec4(0.2, 0.2, 0.2, 1.0 );
var lightDiffuse = vec4( 1.0, 1.0, 1.0, 1.0 );
var lightSpecular = vec4( 1.0, 1.0, 1.0, 1.0 );

var materialAmbient = vec4( 1.0, 0.0, 1.0, 1.0 );
var materialDiffuse = vec4( 1.0, 0.8, 0.0, 1.0);
var materialSpecular = vec4( 1.0, 0.8, 0.0, 1.0 );
var materialShininess = 100.0;

var ctm;
var ambientColor, diffuseColor, specularColor;
var modelView, projection;
var viewerPos;
var program;

var xAxis = 0;
var yAxis = 1;
var zAxis = 2;
var axis = 0;
var theta =[0, 0, 0];

var thetaLoc;

var flag = true;

function quad(a, b, c, d) {

     // Compute the normal vector of the polygon
     var t1 = subtract(vertices[b], vertices[a]);  // vector from a to b
     var t2 = subtract(vertices[c], vertices[b]);  // vector from a to c
     var normal = cross(t1, t2);         // the cross-product of the two vectors
     var normal = vec3(normal);          // conversion from vec4 to vec3
     normal = normalize(normal);         // normalize to unit length

     pointsArray.push(vertices[a]);      // add a vertex to the points array
     normalsArray.push(normal);          // add the normal to the associated array
     pointsArray.push(vertices[b]);
     normalsArray.push(normal);
```

```
     pointsArray.push(vertices[c]);
     normalsArray.push(normal);
     pointsArray.push(vertices[a]);
     normalsArray.push(normal);
     pointsArray.push(vertices[c]);
     normalsArray.push(normal);
     pointsArray.push(vertices[d]);
     normalsArray.push(normal);
}


function colorCube()
{
    quad( 1, 0, 3, 2 );
    quad( 2, 3, 7, 6 );
    quad( 3, 0, 4, 7 );
    quad( 6, 5, 1, 2 );
    quad( 4, 5, 6, 7 );
    quad( 5, 4, 0, 1 );
}


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

    gl = WebGLUtils.setupWebGL( canvas );
    if ( !gl ) { alert( "WebGL 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();       //  create data

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

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

    var vBuffer = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, vBuffer );
    gl.bufferData( gl.ARRAY_BUFFER, flatten(pointsArray), gl.STATIC_DRAW );

    var vPosition = gl.getAttribLocation(program, "vPosition");
    gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

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

    viewerPos = vec3(0.0, 0.0, -20.0 );
```

```
    projection = ortho(-1, 1, -1, 1, -100, 100);

    // Compute the ambient term - Ia*ka.
    ambientProduct = mult(lightAmbient, materialAmbient);

    // Pre-Compute the constant part in the diffuse term - Id*kd(L•N)
    diffuseProduct = mult(lightDiffuse, materialDiffuse);

    // Pre-Compute the constant part in the specular term - Is*ks(H•N)^α
    specularProduct = mult(lightSpecular, materialSpecular);

    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;};

    gl.uniform4fv(gl.getUniformLocation(program, "ambientProduct"),
       flatten(ambientProduct));
    gl.uniform4fv(gl.getUniformLocation(program, "diffuseProduct"),
       flatten(diffuseProduct) );
    gl.uniform4fv(gl.getUniformLocation(program, "specularProduct"),
       flatten(specularProduct) );
    gl.uniform4fv(gl.getUniformLocation(program, "lightPosition"),
       flatten(lightPosition) );

    gl.uniform1f(gl.getUniformLocation(program,
       "shininess"),materialShininess);

    gl.uniformMatrix4fv( gl.getUniformLocation(program, "projectionMatrix"),
       false, flatten(projection));

    render();
}

var render = function(){

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

    if(flag) theta[axis] += 2.0;

    modelView = mat4();
    modelView = mult(modelView, rotate(theta[xAxis], [1, 0, 0] ));
    modelView = mult(modelView, rotate(theta[yAxis], [0, 1, 0] ));
    modelView = mult(modelView, rotate(theta[zAxis], [0, 0, 1] ));

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

    gl.drawArrays( gl.TRIANGLES, 0, numVertices );


    requestAnimFrame(render);
}
```