```
// modified demo to illustrate a single buffer containing objects of
// mixed types:
//      from index 0 to 35: total 36 vertices representing 12 triangles
//      from index 36 to 59: total 24 vertices representing 12 line segments
// and how to draw them in the render function.
//
// The modified parts are highlighted in yellow.
//
// Xue Dong Yang
// 12/02/2019

var canvas;
var gl;

var NumVertices1  = 36;     // 12 triangles
var NumVertices2  = 24;     // 12 edges
var NumVertices  = NumVertices1 + NumVertices2;
                            // total number of vertices


var points = [];
var colors = [];

var xAxis = 0;
var yAxis = 1;
var zAxis = 2;

var axis = 2;
var theta = [ 0, 0, 0 ];

var thetaLoc;

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

    gl = WebGLUtils.setupWebGL( canvas );
    if ( !gl ) { alert( "WebGL isn't available" ); }

    colorCube(); // generating triangles
    cubeEdges(); // generating cube edges

    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
    //
    var program = initShaders( gl, "vertex-shader", "fragment-shader" );
```

```
    gl.useProgram( program );

    var cBuffer = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, cBuffer );
    gl.bufferData( gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW );

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

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

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

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

    //event listeners for buttons

    document.getElementById( "xButton" ).onclick = function () {
        axis = xAxis;
    };
    document.getElementById( "yButton" ).onclick = function () {
        axis = yAxis;
    };
    document.getElementById( "zButton" ).onclick = function () {
        axis = zAxis;
    };

    render();
}

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 );
}

function quad(a, b, c, d)
{
    var vertices = [
        vec3( -0.5, -0.5,  0.5 ),
        vec3( -0.5,  0.5,  0.5 ),
        vec3(  0.5,  0.5,  0.5 ),
```

```
        vec3(  0.5, -0.5,  0.5 ),
        vec3( -0.5, -0.5, -0.5 ),
        vec3( -0.5,  0.5, -0.5 ),
        vec3(  0.5,  0.5, -0.5 ),
        vec3(  0.5, -0.5, -0.5 )
    ];

    var vertexColors = [
        [ 0.0, 0.0, 0.0, 1.0 ],  // black
        [ 1.0, 0.0, 0.0, 1.0 ],  // red
        [ 1.0, 1.0, 0.0, 1.0 ],  // yellow
        [ 0.0, 1.0, 0.0, 1.0 ],  // green
        [ 0.0, 0.0, 1.0, 1.0 ],  // blue
        [ 1.0, 0.0, 1.0, 1.0 ],  // magenta
        [ 1.0, 1.0, 1.0, 1.0 ],  // white
        [ 0.0, 1.0, 1.0, 1.0 ]   // cyan
    ];

    // We need to parition the quad into two triangles in order for
    // WebGL to be able to render it.  In this case, we create two
    // triangles from the quad indices

    //vertex color assigned by the index of the vertex

    var indices = [ a, b, c, a, c, d ];

    for ( var i = 0; i < indices.length; ++i ) {
        points.push( vertices[indices[i]] );
        colors.push( vertexColors[indices[i]] );

        // for solid colored faces use
        //colors.push(vertexColors[a]);

    }
}

function cubeEdges()
{
    edge( 0, 1);       // create 12 edges of the cube
    edge( 1, 2);
    edge( 2, 3);
    edge( 3, 0);
    edge( 4, 5);
    edge( 5, 6);
    edge( 6, 7);
    edge( 7, 4);
    edge( 0, 4);
    edge( 1, 5);
    edge( 2, 6);
    edge( 3, 7);
}
```

```
function edge(a, b)
{
    var vertices = [
        vec3( -0.5, -0.5,  0.5 ),
        vec3( -0.5,  0.5,  0.5 ),
        vec3(  0.5,  0.5,  0.5 ),
        vec3(  0.5, -0.5,  0.5 ),
        vec3( -0.5, -0.5, -0.5 ),
        vec3( -0.5,  0.5, -0.5 ),
        vec3(  0.5,  0.5, -0.5 ),
        vec3(  0.5, -0.5, -0.5 )
    ];

    var vertexColors = [
        [ 0.0, 0.0, 0.0, 1.0 ],  // black
        [ 1.0, 0.0, 0.0, 1.0 ],  // red
        [ 1.0, 1.0, 0.0, 1.0 ],  // yellow
        [ 0.0, 1.0, 0.0, 1.0 ],  // green
        [ 0.0, 0.0, 1.0, 1.0 ],  // blue
        [ 1.0, 0.0, 1.0, 1.0 ],  // magenta
        [ 1.0, 1.0, 1.0, 1.0 ],  // white
        [ 0.0, 1.0, 1.0, 1.0 ]   // cyan
    ];

    points.push( vertices[a] );         // append 2 vertices to the vertex
    points.push( vertices[b] );         // array
    colors.push( vertexColors[0] );     // draw edge in black
    colors.push( vertexColors[0] );

}


function render()
{
    gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    theta[axis] += 0.50;
    gl.uniform3fv(thetaLoc, theta);

    gl.drawArrays( gl.TRIANGLES, 0, NumVertices1 );

    // After the color cube is drawn, the edges of the cube will be drawn
    // next. In order to show that a different transformation can be applied
    // this part, I give another increment to the rotation angle such that
    // the edges will not be aligned with the cube, but a little bit ahead.

    theta[axis] += 6.0;
    gl.uniform3fv(thetaLoc, theta);

    gl.drawArrays( gl.LINES, NumVertices1, NumVertices2 );
```

```
    requestAnimFrame( render );

}
```