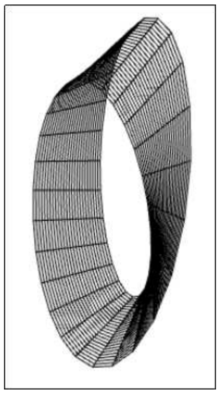
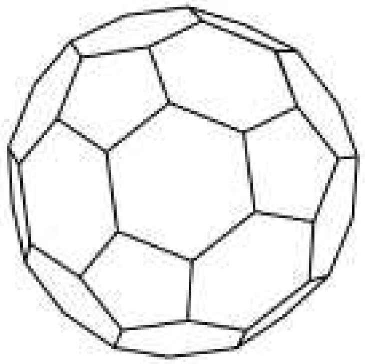
CS 371 – Assignment 3
A Bouncing BuckyBall Bumping Into a . . .
Due: By 11:59pm, Wednesday, October 25th

A "BuckyBall" is the nickname of a chemical molecule whose full name is a *Buckminsterfullerene.*
(see `https://reference.wolfram.com/legacy/v4/GettingStarted/BuckyballConstruction.html`)

Its structure includes 12 pentagon faces and n hexagon faces where n varies depending on the particulars of the molecule's structure. In broad terms, your goal is make a BuckyBall, such as that pictured on the right below, bounce from right to left across the canvas. On the left of side of the canvas looms an interesting surface defined using mathematical formulas such as those we described in class for the sombrero function or Moebius band. When the BuckyBall approaches the surface, it should bounce through or over it in an aesthetically pleasing way.





The particular BuckyBall model you will use for this assignment has 20 hexagonal faces. Its vertices are brought into your work as the array variable *buckyBall* by including the file *buckyball.js* in the *assignment3.html* that you will start your work from:

```
<!DOCTYPE html>
<html>

<p> </p>
<button id = "Button1">Increase Near and Far</button>
<button id = "Button2">Decrease Near and Far</button>
<button id = "Button3">Increase R</button>
<button id = "Button4">Decrease R</button>

<p> </p>
<button id = "Button5">Increase theta</button>
<button id = "Button6">Decrease theta</button>
<button id = "Button7">Increase phi</button>
<button id = "Button8">Decrease phi</button>
<p> </p>


<script id="vertex-shader" type="x-shader/x-vertex">

attribute  vec4 vPosition;

uniform mat4 modelViewMatrix;
uniform mat4 projectionMatrix;

void main()
{
    gl_Position = projectionMatrix*modelViewMatrix*vPosition;
}
</script>

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

precision mediump float;

uniform vec4 fColor;

void
main()
{
    gl_FragColor = fColor;
}
</script>

<script type="text/javascript" src="../Common/webgl-utils.js"></script>
<script type="text/javascript" src="../Common/initShaders.js"></script>
<script type="text/javascript" src="../Common/MV.js"></script>
<script type="text/javascript" src="../Common/webgl-debug.js"></script>
```

```
<script type="text/javascript" src="buckyball.js"></script>
<script type="text/javascript" src="assignment3.js"></script>

<body>
<canvas id="gl-canvas" width="800" height="500">
Oops ... your browser doesn't support the HTML5 canvas element
</canvas>

</body>
</html>
```

NOTE

The corresponding *assignment3.js* code that serves as your starting point presently displays an obscured BuckyBall on the right of the canvas and a boring cube on the right of the canvas.

```
// assignment3.js -- A starting point for your work on Assignment 3

var canvas;
var gl;
var program;

var near = 0.3;
var far = 10.0;
var radius = 4.0;                // Used to establish eye point
var theta  = 0.0;                // Used to establish eye point
var phi    = 0.0;                // Used to establish eye point
var rotation_by_5_deg = 5.0 * Math.PI/180.0;

var  fovy = 45.0;  // Field-of-view in Y direction angle (in degrees)
var  aspect;       // Viewport aspect ratio

var modelViewMatrix, projectionMatrix;
var modelViewMatrixLoc, projectionMatrixLoc;
var eye;                         // Established by radius, theta, phi as we move
const at = vec3(0.0, 0.0, 0.0);
const up = vec3(0.0, 1.0, 0.0);

////////////////// Object 1 vertex information //////////////////

// numVerticesObj1, pointsArray1, vertices1, coordsForObj1 are all
// used to generate the vertex information for "Object 1".  In the
// assignment, you are required to make this object a more interesting
// mathematically defined object such as the sombrero surface or
// Moebius band

var numVerticesObj1  = 36;       // For the 12 triangles

var pointsArray1 = [];

var vertices1 = [
    vec4(-0.5, -0.5,  1.5, 1.0),
    vec4(-0.5,  0.5,  1.5, 1.0),
    vec4(0.5,  0.5,  1.5, 1.0),
    vec4(0.5, -0.5,  1.5, 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)
];

function coordsForObj1()
{
    function quad(a, b, c, d) {
        pointsArray1.push(vertices1[a]);
        pointsArray1.push(vertices1[b]);
        pointsArray1.push(vertices1[c]);
        pointsArray1.push(vertices1[a]);
        pointsArray1.push(vertices1[c]);
        pointsArray1.push(vertices1[d]);
    };

    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 );
}

///////// End of vertex information for Object 1  ////////

window.onload = function init() {

    canvas = document.getElementById( "gl-canvas" );
```

NOTE

```
    //    gl = WebGLUtils.setupWebGL( canvas );
    gl = WebGLDebugUtils.makeDebugContext( canvas.getContext("webgl") ); // For debugging
    if ( !gl ) { alert( "WebGL isn't available" ); }

    gl.viewport( 0, 0, canvas.width, canvas.height );
    aspect =  canvas.width/canvas.height;
    gl.clearColor( 0.0, 0.0, 0.0, 1.0 );
    gl.enable(gl.DEPTH_TEST);


    //
    //  Load shaders and initialize attribute buffers
    //
    program = initShaders( gl, "vertex-shader", "fragment-shader" );
    gl.useProgram( program );

    coordsForObj1();       // This will probably change once you finalize Object 1           NOTE

    var vBuffer = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, vBuffer );
    gl.bufferData( gl.ARRAY_BUFFER, flatten(pointsArray1.concat(buckyBall)),                   NOTE
                   gl.STATIC_DRAW );

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

    modelViewMatrixLoc = gl.getUniformLocation( program, "modelViewMatrix" );
    projectionMatrixLoc = gl.getUniformLocation( program, "projectionMatrix" );

    // buttons for viewing parameters

    document.getElementById("Button1").onclick = function(){near  *= 1.02; far *= 1.02;};
    document.getElementById("Button2").onclick = function(){near *= 0.98; far *= 0.98;};
    document.getElementById("Button3").onclick = function(){radius *= 1.1;};
    document.getElementById("Button4").onclick = function(){radius *= 0.9;};
    document.getElementById("Button5").onclick = function(){theta += rotation_by_5_deg;};
    document.getElementById("Button6").onclick = function(){theta -= rotation_by_5_deg;};
    document.getElementById("Button7").onclick = function(){phi += rotation_by_5_deg;};
    document.getElementById("Button8").onclick = function(){phi -= rotation_by_5_deg;};

    render();
};


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

    eye = vec3(radius*Math.sin(theta)*Math.cos(phi),
               radius*Math.sin(theta)*Math.sin(phi), radius*Math.cos(theta));

    // Object 1                                                                                 NOTE
    modelViewMatrix = lookAt(eye, at , up);
    modelViewMatrix = mult(modelViewMatrix, translate(-1.5,0.0,0.0));
    modelViewMatrix = mult(modelViewMatrix, scalem(0.5,0.5,0.5));
    projectionMatrix = perspective(fovy, aspect, near, far);

    gl.uniformMatrix4fv( modelViewMatrixLoc, false, flatten(modelViewMatrix) );
    gl.uniformMatrix4fv( projectionMatrixLoc, false, flatten(projectionMatrix) );

    gl.uniform4fv(gl.getUniformLocation(program, "fColor"),
                  flatten(vec4(1.0, 0.0, 0.0, 1.0)));
    gl.drawArrays( gl.TRIANGLES, 0, numVerticesObj1 );


    // The BuckyBall                                                                            NOTE
    modelViewMatrix = lookAt(eye, at , up);
    modelViewMatrix = mult(modelViewMatrix, translate(1.75,0.0,0.0));
    modelViewMatrix = mult(modelViewMatrix, scalem(0.03,0.03,0.03));
    projectionMatrix = perspective(fovy, aspect, near, far);

    gl.uniformMatrix4fv( modelViewMatrixLoc, false, flatten(modelViewMatrix) );
    gl.uniformMatrix4fv( projectionMatrixLoc, false, flatten(projectionMatrix) );

    gl.uniform4fv(gl.getUniformLocation(program, "fColor"),
                  flatten(vec4(0.0, .0, 1.0, 1.0)));
    gl.drawArrays( gl.TRIANGLES, numVerticesObj1, buckyBall.length );

    requestAnimFrame(render);
};
```

A partial listing of the *buckyball.js* file:

```
var buckyBall = [
    // Hexagon
    vec4(   3.741182, 0, -19.9988,1.0),
    vec4(   -3.741183, 0, -19.9988,1.0),
    vec4(   -8.050561, 6.972719, -17.335458,1.0),
    vec4(   3.741182, 0, -19.9988,1.0),
    vec4(   -8.050561, 6.972719, -17.335458,1.0),
    vec4(   8.05056, 6.97272, -17.335458,1.0),
    vec4(   8.05056, 6.97272, -17.335458,1.0),
    vec4(   -8.050561, 6.972719, -17.335458,1.0),
    vec4(   4.309377, 13.026081, -15.02328,1.0),
    vec4(   -8.050561, 6.972719, -17.335458,1.0),
    vec4(   -4.309377, 13.026081, -15.02328,1.0),
    vec4(   4.309377, 13.026081, -15.02328,1.0),

    // Hexagon
    vec4(   17.335458, -8.050561, -6.972719,1.0),
    vec4(   15.02328, -4.309377, -13.026081,1.0),
    vec4(   15.02328, 4.309377, -13.026082,1.0),
    vec4(   17.335458, -8.050561, -6.972719,1.0),
    vec4(   15.02328, 4.309377, -13.026082,1.0),
    vec4(   19.9988, -3.741184, 0,1.0),
    vec4(   19.9988, -3.741184, 0,1.0),
    vec4(   15.02328, 4.309377, -13.026082,1.0),
    vec4(   19.9988, 3.741184, 0,1.0),
    vec4(   15.02328, 4.309377, -13.026082,1.0),
    vec4(   17.335458, 8.050561, -6.972718,1.0),
    vec4(   19.9988, 3.741184, 0,1.0),

    // Hexagon
    vec4(   15.02328, -4.309377, -13.026081,1.0),
    vec4(   17.335458, -8.050561, -6.972719,1.0),
    vec4(   13.026083, -15.023278, -4.309375,1.0),
    vec4(   15.02328, -4.309377, -13.026081,1.0),
    vec4(   13.026083, -15.023278, -4.309375,1.0),
    vec4(   8.05056, -6.97272, -17.335458,1.0),
    vec4(   8.05056, -6.97272, -17.335458,1.0),
    vec4(   13.026083, -15.023278, -4.309375,1.0),
    vec4(   4.309377, -13.026082, -15.02328,1.0),
    vec4(   13.026083, -15.023278, -4.309375,1.0),
    vec4(   6.972717, -17.335458, -8.050561,1.0),
    vec4(   4.309377, -13.026082, -15.02328,1.0),

    .
    .
    .

    // Pentagon
    vec4(   -13.026083, 15.023278, 4.309375,1.0),
    vec4(   -13.026083, 15.023278, -4.309376,1.0),
    vec4(   -17.335458, 8.050561, -6.972717,1.0), ];
    vec4(   -13.026083, 15.023278, 4.309375,1.0),
    vec4(   -17.335458, 8.050561, -6.972717,1.0),
    vec4(   -17.335458, 8.050562, 6.972716,1.0),
    vec4(   -17.335458, 8.050561, -6.972717,1.0),
    vec4(   -19.998799, 3.741185, 0,1.0),
    vec4(   -17.335458, 8.050562, 6.972716,1.0),

    // Pentagon
    vec4(   -17.335458, -8.050562, 6.972717,1.0),
    vec4(   -19.998799, -3.741185, 0,1.0),
    vec4(   -17.335458, -8.050561, -6.972718,1.0),
    vec4(   -17.335458, -8.050562, 6.972717,1.0),
    vec4(   -17.335458, -8.050561, -6.972718,1.0),
    vec4(   -13.026082, -15.023278, 4.309376,1.0),
    vec4(   -17.335458, -8.050561, -6.972718,1.0),
    vec4(   -13.026081, -15.02328, -4.309376,1.0),
    vec4(   -13.026082, -15.023278, 4.309376,1.0)
```

It may help to know that, in any group of 12 vertices defining the four triangles of a hexagon, the uniquely defined vertices can be found at offsets 0, 1, 2, 5, 8, and 10 within that group of 12. Similarly, in any group of 9 vertices defining the three triangles of a pentagon, the uniquely defined vertices can be found at offsets 0, 1, 2, 5, and 7 within that group of 9.

Here are the minimal requirements that your program must meet for the 90%-level.

- The Buckyball is obscured and aesthetically unpleasing because all of its faces are rendered in blue. You must change this so that the faces are distinguishable. This will necessitate using at least two colors for the faces – one for the hexagons and one for the pentagons – and also using line loops to outline each of the faces. Because the present program renders the Buckyball using gl.TRIANGLES, you will no doubt have to make some fairly significant changes to the way that you work with the BuckyBall vertices. A careful examination on your part of the way in which the vertices for the triangles comprising a hexagon and pentagon are ordered should trigger some ideas about how you can economize on the duplication of vertices that is presently required for the gl.TRIANGLES rendering technique that is used.
- The bouncing motion of the BuckyBall from right to left should be achieved by using some form of a sin or cos curve in the transformation accumulated in the *modelViewMatrix* and consequently applied to the BuckyBall coordinates in the vertex shader. When the BuckyBall bounces off the left edge of the screen, it should re-appear on the right, thereby creating motion that is continuous until the viewer leaves the program.
- The object on the left side of the canvas that looms in Bucky's bouncing path is merely an unimaginative cube in your starter code. You must replace the cube with an interesting mathematical surface such as a sombrero, a Moebius band, or some other parametrically-defined surface that you discover. If the surface is not a sombrero or Moebius band, be sure to cite the source where you learned about the algebraic formulas that define the surface.

What should you submit?

In one zip file that you upload to the D2L dropbox for Assignment 3, I should find:

- One HTML file named *assignment3.html*
- One JavaScript file named *assignment3.js*.
- The *buckyball.js* file that defines the BuckyBall vertices. It is acceptable to submit a version of this file different from the original if your doing so makes it easier to achieve the rendering requirements for the BuckyBall. However, if you do edit the *buckyball.js* file, make sure that the version you render still has the same 20 hexagons and 12 pentagons as the original.

To load everything correctly, the following must be unchanged in the HTML file you submit:

```
<script type="text/javascript" src="../Common/webgl-utils.js"></script>
<script type="text/javascript" src="../Common/initShaders.js"></script>
<script type="text/javascript" src="../Common/MV.js"></script>
<script type="text/javascript" src="../Common/webgl-debug.js"></script>
<script type="text/javascript" src="buckyball.js"></script>
<script type="text/javascript" src="assignment3.js"></script>
```

If I have to edit any of these lines in order to run your program (for instance, because you didn't follow our conventions as to where the A/S Common utilities are located), a 10% penalty will be imposed. As always, any GGW efforts should be documented to ensure you receive credit for them. That documentation can appear in the introductory documentation block of your *assignment3.js* file or, even better, add it directly into your HTML page so that viewers of your efforts are appropriately informed regarding all the neat stuff you have done.