Following slides provide an animation for two objects.

```
var gl;
var myShaderProgramTri;
var myShaderProgramSquare;
var thetaTri;
var thetaSquare;
var count;
var arrayOfPointsForTri;
var arrayOfPointsForSquare;
var bufferIdTri;
var bufferIdSquare;

function initGL() {
    var canvas = document.getElementById("gl-canvas");
    gl = WebGLUtils.setupWebGL( canvas );
    if (!gl) { alert( "WebGL is not available" ); }

    gl.viewport( 0, 0, 512, 512 );
    gl.clearColor( 1.0, 0.0, 0.0, 1.0 );

    thetaTri=.0;
    thetaSquare=.0;
    count=.0;

    // Initialize programs
    myShaderProgramTri =
    initShaders( gl, "vertex-shader", "fragment-shader" );

    myShaderProgramSquare =
    initShaders( gl, "vertex-shader2", "fragment-shader2" );
```

Set up two shaders programs, one for the triangle, and one for the square (vertex and fragment shaders not shown here).

```
    // Initialize Triangle
    var point0 = vec2( 0.0, 0.0 );
    var point1 = vec2( 1.0, 0.0 );
    var point2 = vec2( 0.0, 1.0 );

    arrayOfPointsForTri = [ point0, point1, point2 ];


    // Set up the buffer for the triangle
    bufferIdTri = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdTri );
    gl.bufferData( gl.ARRAY_BUFFER, flatten( arrayOfPointsForTri ),gl.STATIC_DRAW );

    // Initialize Square
    var point0 = vec2( -1.0, 0.0 );
    var point1 = vec2( -1.0, -1.0 );
    var point2 = vec2( 0.0, -1.0 );
    var point3 = vec2( 0.0, 0.0 );

    arrayOfPointsForSquare = [ point0, point1, point2, point3 ];

    // Set up the buffer for the square
    bufferIdSquare = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdSquare );
    gl.bufferData( gl.ARRAY_BUFFER, flatten( arrayOfPointsForSquare ), gl.STATIC_DRAW );
```

Initialize the triangle and square points, and send their data to the GPU.

```
    render();
}
```

Call the render() function.

```
function render() {
    gl.clear( gl.COLOR_BUFFER_BIT );

    // Use the shader program for the triangle
    gl.useProgram( myShaderProgramTri );

    // Update the animation angle for the triangle
    thetaTri=thetaTri+.1;
    var thetalocTri=gl.getUniformLocation(myShaderProgramTri,"theta");
    gl.uniform1f(thetalocTri,thetaTri);

    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdTri );

    // Set up the attributes for the triangle based on the current buffer
    var myPositionTri = gl.getAttribLocation(myShaderProgramTri, "myPosition");
    gl.vertexAttribPointer(myPositionTri,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray( myPositionTri );

    // Draw the triangle
    gl.drawArrays( gl.TRIANGLES, 0, 3 );


    // Use the shader program for the square
    gl.useProgram( myShaderProgramSquare );

    // Update the animation angle for the square
    thetaSquare=thetaSquare+.2;
    var thetalocSquare=gl.getUniformLocation(myShaderProgramSquare,"theta");
    gl.uniform1f(thetalocSquare,thetaSquare);

    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdSquare );

    // Set up the attributes for the square based on the current buffer
    var myPositionSquare = gl.getAttribLocation(myShaderProgramSquare, "myPosition");
    gl.vertexAttribPointer(myPositionSquare,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray( myPositionSquare );

    // Draw the square
    gl.drawArrays( gl.TRIANGLE_FAN, 0, 4 );

    // Request the next render frame
    requestAnimFrame(render);
}
```

First set up the uniforms, buffer, and attributes for the triangle, and draw it.

Then set up the uniforms, buffer, and attributes for the square, and draw it.

At the end, request the next frame.

```
function render() {
    gl.clear( gl.COLOR_BUFFER_BIT );

    // Use the shader program for the triangle
    gl.useProgram( myShaderProgramTri );

    // Update the animation angle for the triangle
    thetaTri=thetaTri+.1;
    var thetalocTri=gl.getUniformLocation(myShaderProgramTri,"theta");
    gl.uniform1f(thetalocTri,thetaTri);

    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdTri );

    // Set up the attributes for the triangle based on the current buffer
    var myPositionTri = gl.getAttribLocation(myShaderProgramTri, "myPosition");
    gl.vertexAttribPointer(myPositionTri,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray( myPositionTri );

    // Draw the triangle
    gl.drawArrays( gl.TRIANGLES, 0, 3 );


    // Use the shader program for the square
    gl.useProgram( myShaderProgramSquare );

    // Update the animation angle for the square
    thetaSquare=thetaSquare+.2;
    var thetalocSquare=gl.getUniformLocation(myShaderProgramSquare,"theta");
    gl.uniform1f(thetalocSquare,thetaSquare);

    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdSquare );

    // Set up the attributes for the square based on the current buffer
    var myPositionSquare = gl.getAttribLocation(myShaderProgramSquare, "myPosition");
    gl.vertexAttribPointer(myPositionSquare,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray( myPositionSquare );

    // Draw the square
    gl.drawArrays( gl.TRIANGLE_FAN, 0, 4 );

    // Request the next render frame
    requestAnimFrame(render);
}
```

The boxed portions are responsible for performing the animation by updating the uniforms at every frame.

```
function render() {
    gl.clear( gl.COLOR_BUFFER_BIT );

    // Use the shader program for the triangle
    gl.useProgram( myShaderProgramTri );

    // Update the animation angle for the triangle
    thetaTri=thetaTri+.1;
    var thetalocTri=gl.getUniformLocation(myShaderProgramTri,"theta");
    gl.uniform1f(thetalocTri,thetaTri);

    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdTri );

    // Set up the attributes for the triangle based on the current buffer
    var myPositionTri = gl.getAttribLocation(myShaderProgramTri, "myPosition");
    gl.vertexAttribPointer(myPositionTri,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray( myPositionTri );

    // Draw the triangle
    gl.drawArrays( gl.TRIANGLES, 0, 3 );
```

```
    // Use the shader program for the square
    gl.useProgram( myShaderProgramSquare );

    // Update the animation angle for the square
    thetaSquare=thetaSquare+.2;
    var thetalocSquare=gl.getUniformLocation(myShaderProgramSquare,"theta");
    gl.uniform1f(thetalocSquare,thetaSquare);

    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdSquare );

    // Set up the attributes for the square based on the current buffer
    var myPositionSquare = gl.getAttribLocation(myShaderProgramSquare, "myPosition");
    gl.vertexAttribPointer(myPositionSquare,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray( myPositionSquare );

    // Draw the square
    gl.drawArrays( gl.TRIANGLE_FAN, 0, 4 );

    // Request the next render frame
    requestAnimFrame(render);
}
```

If you use multiple shaders, one per shape, make sure you call the correct program for each shape before drawing the shape.

```
function render() {
    gl.clear( gl.COLOR_BUFFER_BIT );

    // Use the shader program for the triangle
    gl.useProgram( myShaderProgramTri );

    // Update the animation angle for the triangle
    thetaTri=thetaTri+.1;
    var thetalocTri=gl.getUniformLocation(myShaderProgramTri,"theta");
    gl.uniform1f(thetalocTri,thetaTri);

    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdTri );

    // Set up the attributes for the triangle based on the current buffer
    var myPositionTri = gl.getAttribLocation(myShaderProgramTri, "myPosition");
    gl.vertexAttribPointer(myPositionTri,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray( myPositionTri );

    // Draw the triangle
    gl.drawArrays( gl.TRIANGLES, 0, 3 );


    // Use the shader program for the square
    gl.useProgram( myShaderProgramSquare );

    // Update the animation angle for the square
    thetaSquare=thetaSquare+.2;
    var thetalocSquare=gl.getUniformLocation(myShaderProgramSquare,"theta");
    gl.uniform1f(thetalocSquare,thetaSquare);

    gl.bindBuffer( gl.ARRAY_BUFFER, bufferIdSquare );

    // Set up the attributes for the square based on the current buffer
    var myPositionSquare = gl.getAttribLocation(myShaderProgramSquare, "myPosition");
    gl.vertexAttribPointer(myPositionSquare,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray( myPositionSquare );

    // Draw the square
    gl.drawArrays( gl.TRIANGLE_FAN, 0, 4 );

    // Request the next render frame
    requestAnimFrame(render);
}
```

BEFORE DRAWING A SHAPE, BIND THE ARRAY BUFFER TO THE CORRECT BUFFER.