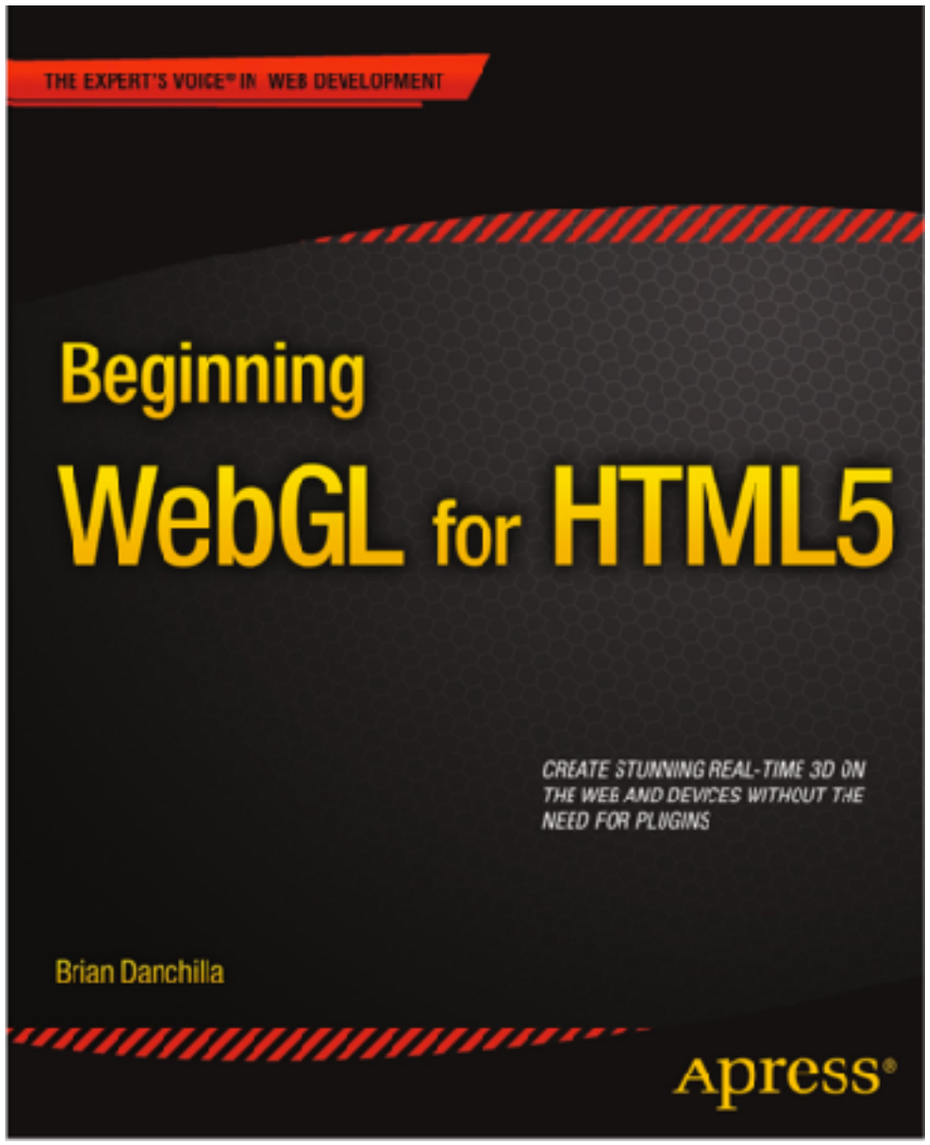
# BCA611 Video Oyunları için 3B Grafik

## WebGL’e giriş

Zümra Kavafoğlu

# Kaynaklar




- Brian Danchilla, Beginning WebGL for HTML5, Apress

- http://www.springer.com/br/book/9781430239963

**https://github.com/apress/beg-webgl-for-html5**

http://learningwebgl.com

# WebGL

- **WebGL** (**Web Graphics Library**) 3B grafiklerin, uyumlu web tarayıcılarında herhangi eklentiye(plug-in) ihtiyaç olmadan renderlanmasını sağlayan bir JavaScript API'sidir.

- Kullanımı OpenGL'e çok benzerdir.

- Herhangi bir kurulum gerektirmez.

- Sistem bağımlı değildir.

http://learningwebgl.com

# İlk WebGL programı: Canvas oluşturma ve renklendirme

```
<!doctype html>
<html>
    <head>
        <title>Background Color</title>
        <style>
            body{ background-color: grey; }
            canvas{ background-color: white; }
        </style>
        <script>
            var gl = null,
                canvas = null;

            function initWebGL()
            {
                canvas = document.getElementById("my-canvas");
                try{
                    gl = canvas.getContext("webgl");
                }catch(e){
                }

                if(gl)
                {
                    setupWebGL();
                }else{
                    alert(  "Error: Your browser does not appear to support WebGL.");
                }
            }

            function setupWebGL()
            {
                //set the clear color to a shade of green
                gl.clearColor(0.08, 0.74, 0.8, 1.0);
                gl.clear(gl.COLOR_BUFFER_BIT);

                gl.viewport(0, 0, canvas.width, canvas.height);
            }


        </script>
    </head>
    <body onload="initWebGL()">
        <canvas id="my-canvas" width="800" height="600">
        Your browser does not support the HTML5 canvas element.
        </canvas>
    </body>
</html>
```

***HTML <canvas> elemanı bir web sayfasında grafik çizmek için kullanılan alandır.***

# İlk WebGL programı: Canvas oluşturma ve renklendirme

```
<!doctype html>
<html>
    <head>
        <title>Background Color</title>
        <style>
            body{ background-color: grey; }
            canvas{ background-color: white; }
        </style>
        <script>
            var gl = null,
                canvas = null;

function initWebGL()
{
    canvas = document.getElementById("my-canvas");
    try{
        gl = canvas.getContext("webgl");
    }catch(e){
    }

    if(gl)
    {
        setupWebGL();
    }else{
        alert(  "Error: Your browser does not appear to support WebGL.");
    }
}
```

*Tarayıcı webgl destekliyorsa null'dan farklı bir değer döner*

# İlk WebGL programı: Canvas oluşturma ve renklendirme

```
29          function setupWebGL()
30          {
31              //set the clear color to a shade of green
32              gl.clearColor(0.08, 0.74, 0.8, 1.0);
33              gl.clear(gl.COLOR_BUFFER_BIT);
34
35              gl.viewport(0, 0, canvas.width, canvas.height);
36          }
37
38
39          </script>
40      </head>
41      <body onload="initWebGL()">
42          <canvas id="my-canvas" width="800" height="600">
43          Your browser does not support the HTML5 canvas elemen
44          </canvas>
45      </body>
46  </html>
```

*Arkaplan rengini değiştirir.*

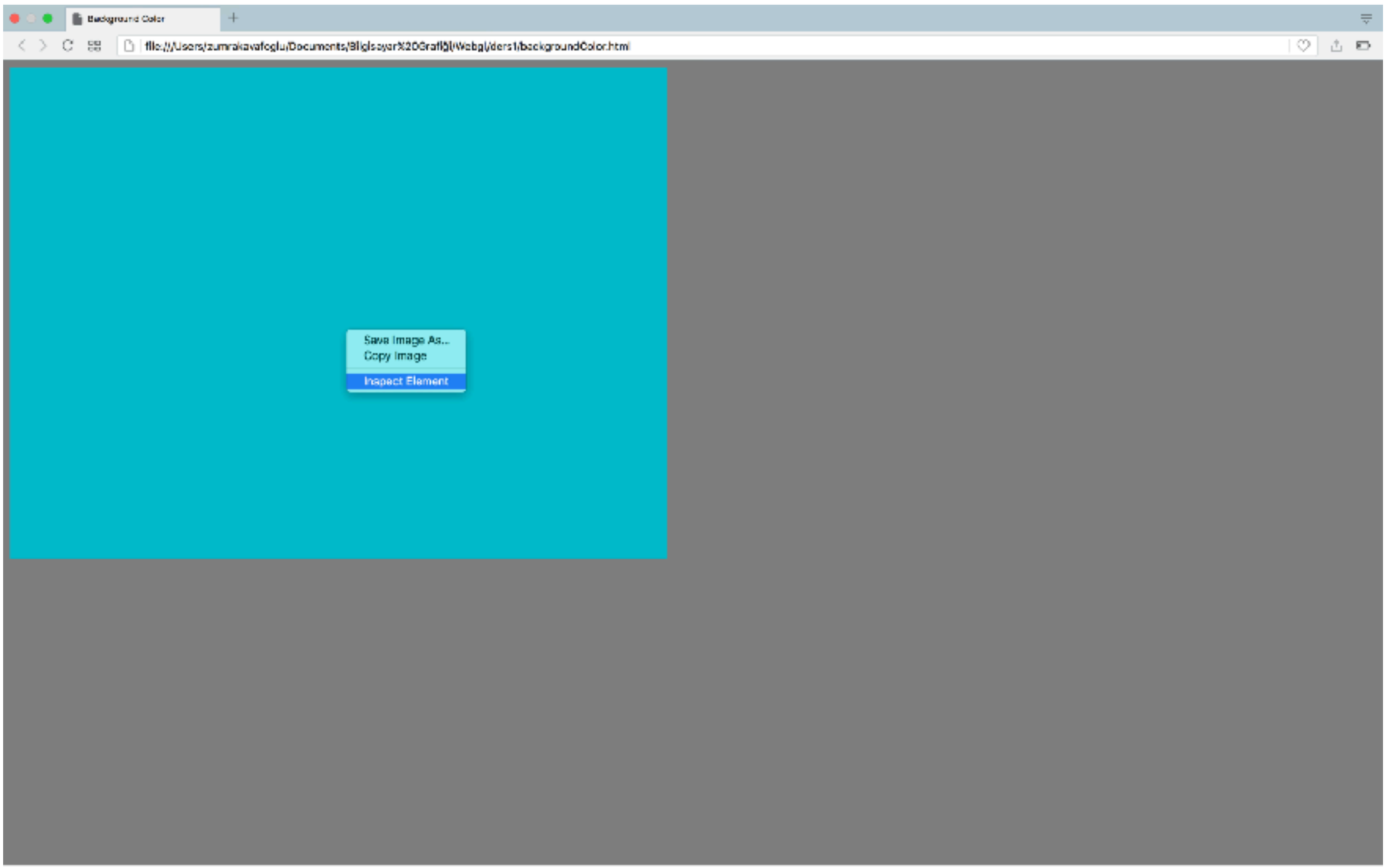
*Çizim yapılacak alanın boyutlarını belirler*

# İlk WebGL programı: Canvas oluşturma ve renklendirme

Dosyayı .html uzantısıyla kaydedip çalıştırdığımızda tarayıcımızda bu sayfa açılır.

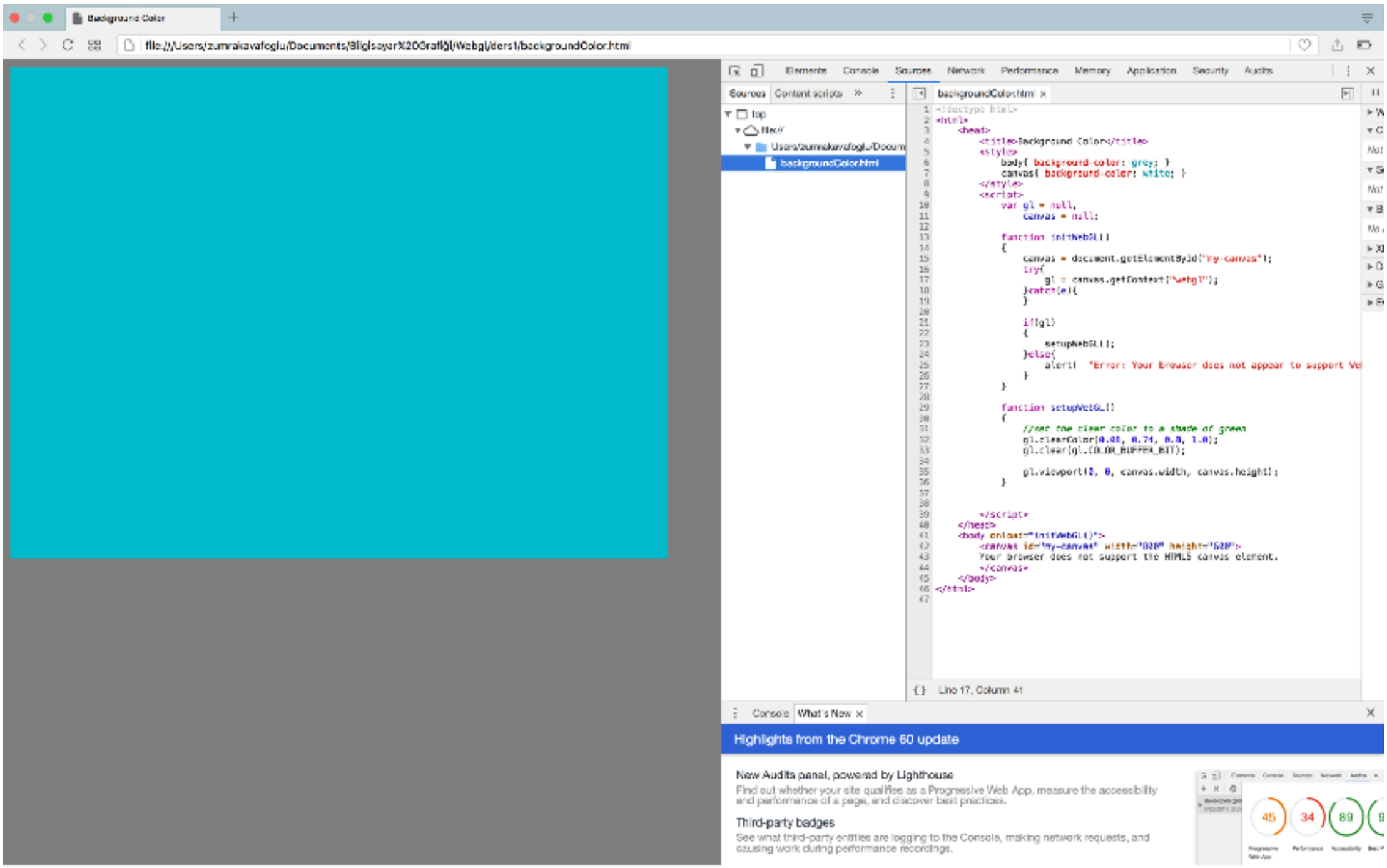
# İlk WebGL programı: Canvas oluşturma ve renklendirme

Bu sayfa üzerinde kodumuzu ve varsa hatalarını görebiliriz.



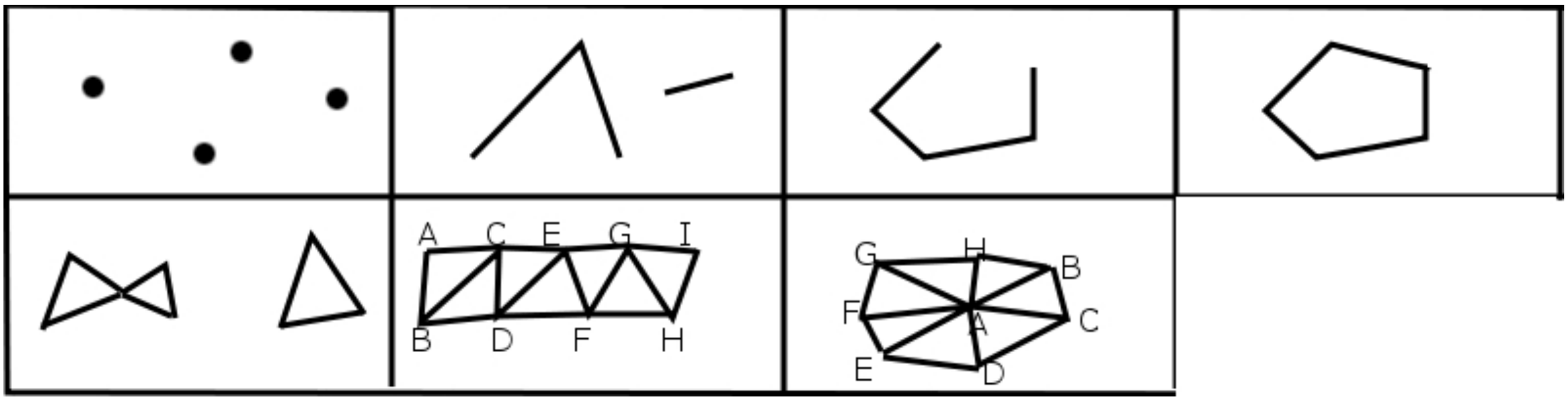
# İlk WebGL programı: Canvas oluşturma ve renklendirme

Bu sayfa üzerinde kodumuzu ve varsa hatalarını görebiliriz.



# WebGL primitif tipleri

Primitifler bir grafik API'siyle oluşturulan tüm modellerin yapıtaşlarını oluştururlar.
Primitflerin yapıtaşları ise köşelerdir.




Üst satır: POINTS, LINES, LINE_STRIP, LINE_LOOP
Alt satır: TRIANGLES, TRIANGLE_STRIP, TRIANGLE_FAN

OpenGL'in aksine WebGL QUAD primitifine sahip değildir.

# VBO (Vertex Buffer Object / Köşe Belleği Nesnesi)

Bir köşe ile ilişkili tüm veri JavaScript API'sinden GPU'ya aktarılmalıdır. WebGL'de köşenin özelliklerini tutan VBO'lar oluşturulmalıdır. Bu VBO'lar daha sonra köşe ile ilgili veriyi işleyen shader programlarına gönderirler. Shaderlarla ilgili detaylı bilgiyi ileriki derslerde göreceğiz.

Her VBO, köşenin bir özelliğini tutar, bu özellik pozisyon, renk, normal vektörü, doku koordinatları vs. olabilir.

# Tek bir üçgenin köşe pozisyonları için VBO oluşturma

(-0.5,-0.5) , (0.5,-0.5) ve (0,0) köşe pozisyonlarına sahip bir üçgen çizdirmek için bu pozisyonları tutan bir VBO oluşturmalıyız.

```
function setupBuffers()
{
    var triangleVertices = [
        -0.5, -0.5, 0.0,
         0.5, -0.5, 0.0,
         0.0, 0.0, 0.0,
    ];

    triangleVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVertices), gl.STATIC_DRAW);
}
```

# Tek bir üçgen çizdirme

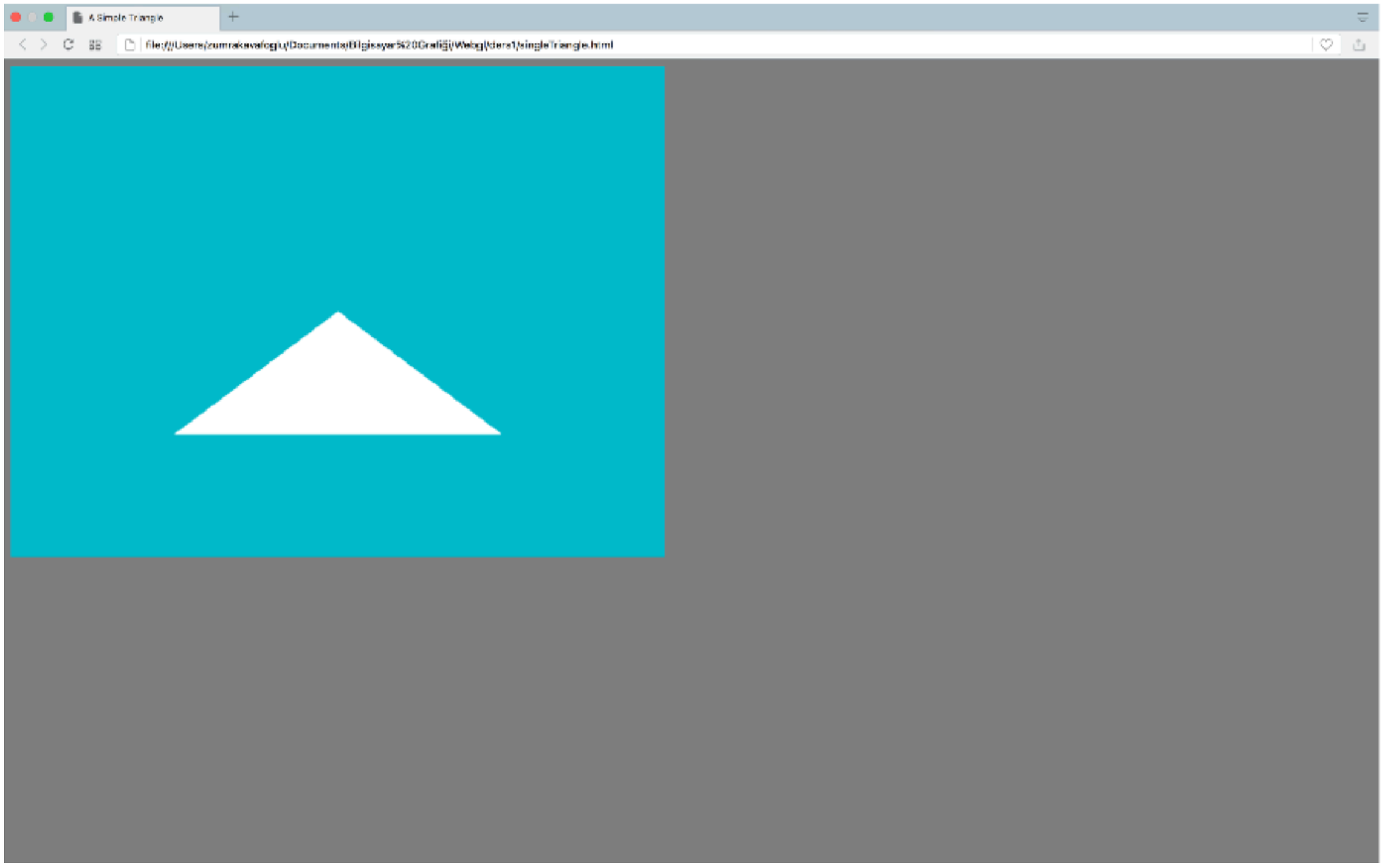
```
function drawScene()
{
    vertexPositionAttribute = gl.getAttribLocation(glProgram, "aVertexPosition");
    gl.enableVertexAttribArray(vertexPositionAttribute);

    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0, 0);
    gl.drawArrays(gl.TRIANGLES, 0, 3);
}
```

*Primitif tipi*

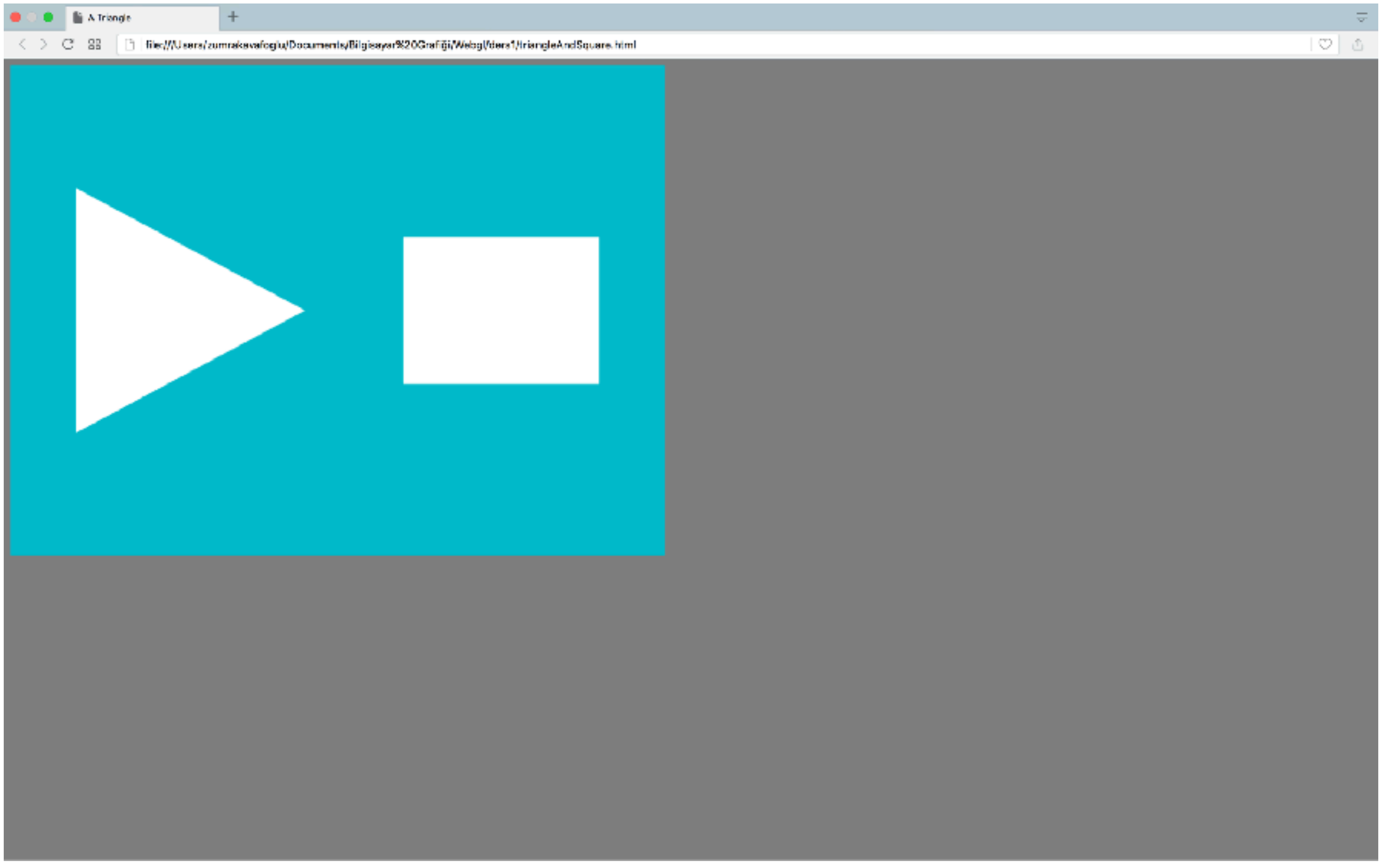
*Özelliğin (köşe pozisyonu) sayısı*

*Özelliğin (köşe pozisyonu) boyutu*

# Üçgeni çizdirme



# Bir üçgen ve bir kare çizdirme



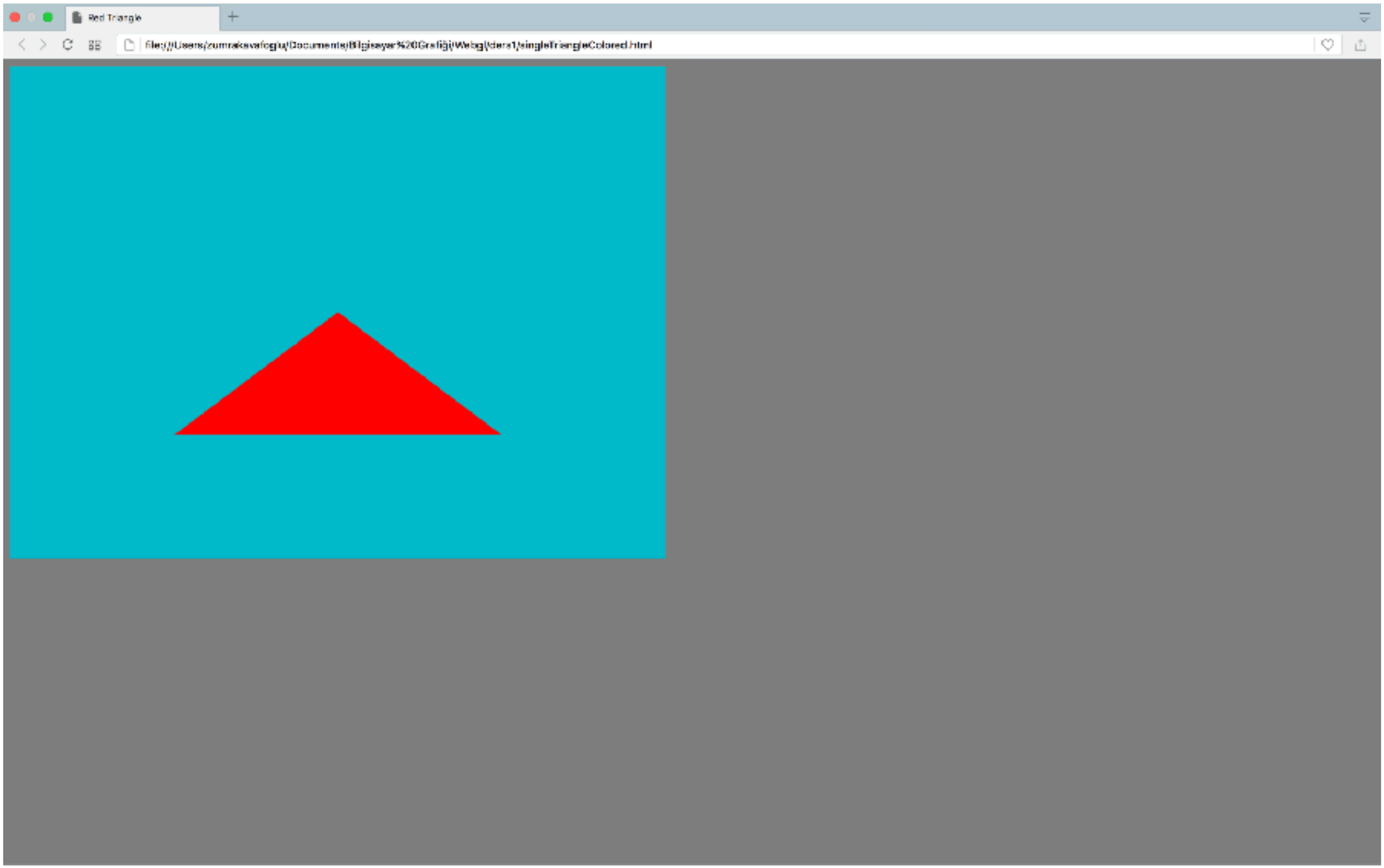
# Bir üçgen ve bir karenin köşe pozisyonları için VBO oluşturma

```
function setupBuffers()
{
    var triangleVertices = [
        -0.8, 0.5, 0.0,
         -0.8, -0.5, 0.0,
        -0.1, 0.0, 0.0,
    ];

    triangleVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVertices), gl.STATIC_DRAW);

    var squareVertices = [
         0.2,  0.3,  0.0,
         0.2,  -0.3,  0.0,
         0.8, 0.3,  0.0,
         0.8, -0.3,  0.0
    ];
    squareVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, squareVertexPositionBuffer);

    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(squareVertices), gl.STATIC_DRAW);

}
```

# Üçgen ve kareyi çizdirme

```
function drawScene()
{
    vertexPositionAttribute = gl.getAttribLocation(glProgram, "aVertexPosition");
    gl.enableVertexAttribArray(vertexPositionAttribute);

    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0, 0);
    gl.drawArrays(gl.TRIANGLES, 0, 3);

    gl.bindBuffer(gl.ARRAY_BUFFER, squareVertexPositionBuffer);
    gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0, 0);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, 4);
}
```

# Üçgeni renklendirme



# Üçgeni renklendirme

```
</style>
<script id="shader-vs" type="x-shader/x-vertex">
    attribute vec3 aVertexPosition;
    attribute vec3 aVertexColor;
    varying highp vec4 vColor;
    void main(void) {
        gl_Position = vec4(aVertexPosition, 1.0);
        vColor = vec4(aVertexColor, 1.0);
    }
</script>
<script id="shader-fs" type="x-shader/x-fragment">
    varying highp vec4 vColor;
    void main(void) {
       gl_FragColor = vColor;
    }
</script>
```

# Üçgeni renklendirme

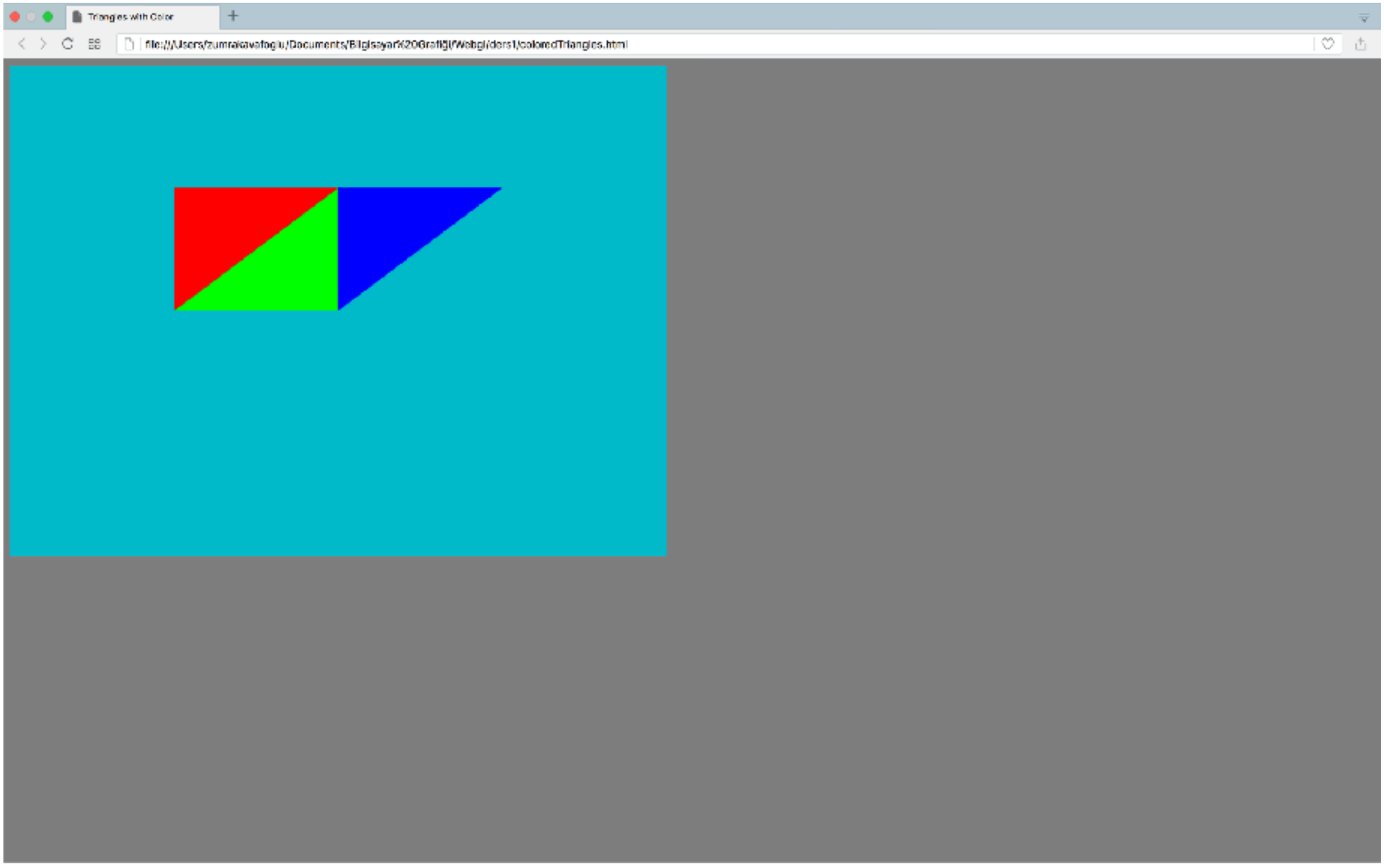
```
function setupBuffers()
{
    var triangleVertices = [
        -0.5, -0.5, 0.0,
         0.5, -0.5, 0.0,
        0.0, 0.0, 0.0,
    ];

    triangleVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVertices), gl.STATIC_DRAW);

    var triangleVertexColors = [
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,
    ];
    triangleColorBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleColorBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVertexColors), gl.STATIC_DRAW);

}
```

*Birinci köşe için RGB değerleri*

# Üçgeni renklendirme

```
function drawScene()
{
    vertexPositionAttribute = gl.getAttribLocation(glProgram, "aVertexPosition");
    gl.enableVertexAttribArray(vertexPositionAttribute);

    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0, 0);

    vertexColorAttribute = gl.getAttribLocation(glProgram, "aVertexColor");
    gl.enableVertexAttribArray(vertexColorAttribute);
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleColorBuffer);
    gl.vertexAttribPointer(vertexColorAttribute, 3, gl.FLOAT, false, 0, 0);

    gl.drawArrays(gl.TRIANGLES, 0, 3);
}
```

# Farklı renkte bitişik üçgenler oluşturma



# Farklı renkte bitişik üçgenler oluşturma

```
function setupBuffers()
{
    var triangleVertices = [
        -0.5, 0.5, 0.0,
         -0.5, 0.0, 0.0,
        0.0, 0.5, 0.0,

        -0.5, 0.0, 0.0,
        0.0, 0.5, 0.0,
        0.0, 0.0, 0.0,

        0.0, 0.5, 0.0,
        0.0, 0.0, 0.0,
        0.5, 0.5, 0.0,
    ];

    triangleVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVertices), gl.STATIC_DRAW);

    var triangleVertexColors = [
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,

        0.0, 1.0, 0.0,
        0.0, 1.0, 0.0,
        0.0, 1.0, 0.0,

        0.0, 0.0, 1.0,
        0.0, 0.0, 1.0,
        0.0, 0.0, 1.0,
    ];
    triangleColorBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleColorBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVertexColors), gl.STATIC_DRAW);

}
```

# Farklı renkte bitişik üçgenler oluşturma

```
function drawScene()
{
    vertexPositionAttribute = gl.getAttribLocation(glProgram, "aVertexPosition");
    gl.enableVertexAttribArray(vertexPositionAttribute);
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0, 0);

    vertexColorAttribute = gl.getAttribLocation(glProgram, "aVertexColor");
    gl.enableVertexAttribArray(vertexColorAttribute);
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleColorBuffer);
    gl.vertexAttribPointer(vertexColorAttribute, 3, gl.FLOAT, false, 0, 0);

    gl.drawArrays(gl.TRIANGLES, 0, 9);
}
```

# Index Buffer kullanarak bitişik üçgenler çizdirme

```
function setupBuffers()
{
    var triangleVertices = [
        -0.5, 0.5, 0.0,
         -0.5, 0.0, 0.0,
        0.0, 0.5, 0.0,
        0.0, 0.0, 0.0,
        0.5, 0.5, 0.0,
    ];

    triangleVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVertices), gl.STATIC_DRAW);

    var triangleVertexColors = [
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,
        0.0, 1.0, 0.0,
        0.0, 0.0, 1.0,
    ];
    triangleColorBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleColorBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVertexColors), gl.STATIC_DRAW);

    var triangleVertexIndices = [
        //front face
        0,1,2,
        1,3,2,
        2,3,4
    ];

    triangleVerticesIndexBuffer = gl.createBuffer();
    triangleVerticesIndexBuffer.number_vertex_points = triangleVertexIndices.length;
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, triangleVerticesIndexBuffer);
    gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Uint16Array(triangleVertexIndices), gl.STATIC_DRAW);

}
```

***Ortak köşelerin pozisyon ve renk değerleri yalnızca bir kere yazılıyor***

# Index Buffer kullanarak bitişik üçgenler çizdirme

```
function drawScene()
{
    vertexPositionAttribute = gl.getAttribLocation(glProgram, "aVertexPosition");
    gl.enableVertexAttribArray(vertexPositionAttribute);
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleVertexPositionBuffer);
    gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0, 0);

    vertexColorAttribute = gl.getAttribLocation(glProgram, "aVertexColor");
    gl.enableVertexAttribArray(vertexColorAttribute);
    gl.bindBuffer(gl.ARRAY_BUFFER, triangleColorBuffer);
    gl.vertexAttribPointer(vertexColorAttribute, 3, gl.FLOAT, false, 0, 0);

    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, triangleVerticesIndexBuffer);
    gl.drawElements(gl.TRIANGLES, triangleVerticesIndexBuffer.number_vertex_points, gl.UNSIGNED_SHORT, 0);

    gl.drawArrays(gl.TRIANGLES, 0, 9);
}
```

# Index Buffer kullanarak bitişik üçgenler çizdirme