Bilgisayar Grafikler, Ders - WebGL Dönem Ödevi Kod Dosyalarının Dökümantasyon Raporu

Giriş

Bu proje, WebGL kullanarak "Ercan Turan" metnini çizen basit bir 3D grafik uygulamasıdır. Proje, HTML, JavaScript ve WebGL ile oluşturulmuştur ve her bir harf farklı renklere sahip olacak şekilde çizilmiştir.

Kod Yapısı ve Açıklamalar

Kod yapısında tek HTML projesi olarak yazdım GitHub'a ama yaptığım tüm projeleri yükleyeceğim js kodunu yarı koymadım projenin son hali son.html'dir. Aşağıda Kod yapılarını ekran fotosu olarak ekledim şimdi html yapısını da ekleyeceğim buraya.

Bu HTML yapısı, projenin temelini oluşturur. Bir canvas elementi ve harici bir JavaScript dosyası (script.js) içerir.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>WebGL ERCAN TURAN</title>
    <style>
      canvas {
        width: 100%;
        height: 100%;
      }
    </style>
</head>
<body>
    <canvas id="glcanvas"></canvas>
    <script src="https://cdn.jsdelivr.net/npm/gl-matrix@2.8.1/dist/gl-matrix-
min.js"></script>
    <script src="main.js"></script> //bu script yok son porjede çünkü tek
proje olrak yaptım
</body>
</html>
```

JavaScript (script.js)

Geldiniz | index.html | index copy.html | index copy 2.html | main.js | a.html | son.html

son.html > html > body > script > initBuffers > colors

```html
<!DOCTYPE html>
<html lang="tr">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>WebGL Ercan Turan</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/gl-matrix/2.8.1/gl-matrix-min.js"></script>
  </head>
  <body>
    <canvas id="glcanvas" width="640" height="480"></canvas>
    <script>
      function main() {
        const canvas = document.querySelector("#glcanvas");
        const gl = canvas.getContext("webgl");

        // WebGL bağlamı kontrolü
        if (!gl) {
          console.error(
            "WebGL bağlamı başlatılamadı. Tarayıcınız WebGL'i desteklemiyor."
          );
          return;
        }

        // Vertex shader kaynağı
        const vsSource = `
          attribute vec4 aVertexPosition;
          attribute vec4 aVertexColor;
          uniform mat4 uModelViewMatrix;
          uniform mat4 uProjectionMatrix;
          varying lowp vec4 vColor;
          void main(void) {
            gl_Position = uProjectionMatrix * uModelViewMatrix * aVertexPosition;
            vColor = aVertexColor;
          }
        `;

        // Fragment shader kaynağı
        const fsSource = `
          varying lowp vec4 vColor;
          void main(void) {
            gl_FragColor = vColor;
          }
        `;

        // Shader programını başlat
        const shaderProgram = initShaderProgram(gl, vsSource, fsSource);
        const programInfo = {
          program: shaderProgram,
          attribLocations: {
            vertexPosition: gl.getAttribLocation(
              shaderProgram,
              "aVertexPosition"
            ),
            vertexColor: gl.getAttribLocation(shaderProgram, "aVertexColor"),
          },
          uniformLocations: {
            projectionMatrix: gl.getUniformLocation(
              shaderProgram,
              "uProjectionMatrix"
            ),
            modelViewMatrix: gl.getUniformLocation(
              shaderProgram,
```

```
            shaderProgram,
            "uModelViewMatrix"
          ),
        },
      };

      // Bufferları başlat
      const buffers = initBuffers(gl);

      // Sahneyi çiz
      drawScene(gl, programInfo, buffers);
    }

    // Shader programını başlatma fonksiyonu
    function initShaderProgram(gl, vsSource, fsSource) {
      const vertexShader = loadShader(gl, gl.VERTEX_SHADER, vsSource);
      const fragmentShader = loadShader(gl, gl.FRAGMENT_SHADER, fsSource);

      const shaderProgram = gl.createProgram();
      gl.attachShader(shaderProgram, vertexShader);
      gl.attachShader(shaderProgram, fragmentShader);
      gl.linkProgram(shaderProgram);

      // Shader programı bağlantı kontrolü
      if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) {
        console.error(
          "Shader programı başlatılamadı: " +
            gl.getProgramInfoLog(shaderProgram)
        );
        return null;
      }

      return shaderProgram;
    }

    // Shader yükleme fonksiyonu
    function loadShader(gl, type, source) {
      const shader = gl.createShader(type);
      gl.shaderSource(shader, source);
      gl.compileShader(shader);

      // Shader derleme kontrolü
      if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
        console.error("Shader derlenemedi: " + gl.getShaderInfoLog(shader));
        gl.deleteShader(shader);
        return null;
      }

      return shader;
    }

    // Bufferları başlatma fonksiyonu
    function initBuffers(gl) {
      // Pozisyon verileri
      const positions = [
        // E harfi
        -0.9,
        0.9,
        0.0,
```

```
    // Pozisyon buffer'ı oluşturma ve verileri yükleme
    const positionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
    gl.bufferData(
      gl.ARRAY_BUFFER,
      new Float32Array(positions),
      gl.STATIC_DRAW
    );
    // Renk buffer'ı oluşturma ve verileri yükleme
    const colorBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, colorBuffer);
    gl.bufferData(
      gl.ARRAY_BUFFER,
      new Float32Array(colors),
      gl.STATIC_DRAW
    );

    return {
      position: positionBuffer,
      color: colorBuffer,
    };
  }
  // Sahneyi çizme fonksiyonu
  function drawScene(gl, programInfo, buffers) {
    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.clearDepth(1.0);
```

```
611       }
612       // Sahneyi çizme fonksiyonu
613       function drawScene(gl, programInfo, buffers) {
614         gl.clearColor(0.0, 0.0, 0.0, 1.0);
615         gl.clearDepth(1.0);
616         gl.enable(gl.DEPTH_TEST);
617         gl.depthFunc(gl.LEQUAL);
618         gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
619         // Perspektif matrisi ayarlama
620         const fieldOfView = (45 * Math.PI) / 180;
621         const aspect = gl.canvas.clientWidth / gl.canvas.clientHeight;
622         const zNear = 0.1;
623         const zFar = 100.0;
624         const projectionMatrix = mat4.create();
625         mat4.perspective(projectionMatrix, fieldOfView, aspect, zNear, zFar);
626         // Model-görünüm matrisi ayarlama
627         const modelViewMatrix = mat4.create();
628         mat4.translate(modelViewMatrix, modelViewMatrix, [-0.0, 0.0, -6.0]);
629         // Pozisyon buffer'ını bağlama ve vertex attribute'u aktifleştirme
630         {
631           const numComponents = 3; // X ve Y ve Z bileşenleri
632           const type = gl.FLOAT;
633           const normalize = false;
634           const stride = 0;
635           const offset = 0;
636           gl.bindBuffer(gl.ARRAY_BUFFER, buffers.position);
637           gl.vertexAttribPointer(
638             programInfo.attribLocations.vertexPosition,
639             numComponents,
640             type,
```

```
611         }
612         // Sahneyi çizme fonksiyonu
613         function drawScene(gl, programInfo, buffers) {
614           gl.clearColor(0.0, 0.0, 0.0, 1.0);
615           gl.clearDepth(1.0);
616           gl.enable(gl.DEPTH_TEST);
617           gl.depthFunc(gl.LEQUAL);
618           gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
619           // Perspektif matrisi ayarlama
620           const fieldOfView = (45 * Math.PI) / 180;
621           const aspect = gl.canvas.clientWidth / gl.canvas.clientHeight;
622           const zNear = 0.1;
623           const zFar = 100.0;
624           const projectionMatrix = mat4.create();
625           mat4.perspective(projectionMatrix, fieldOfView, aspect, zNear, zFar);
626           // Model-görünüm matrisi ayarlama
627           const modelViewMatrix = mat4.create();
628           mat4.translate(modelViewMatrix, modelViewMatrix, [-0.0, 0.0, -6.0]);
629           // Pozisyon buffer'ını bağlama ve vertex attribute'u aktifleştirme
630           {
631             const numComponents = 3; // X ve Y ve Z bileşenleri
632             const type = gl.FLOAT;
633             const normalize = false;
634             const stride = 0;
635             const offset = 0;
636             gl.bindBuffer(gl.ARRAY_BUFFER, buffers.position);
637             gl.vertexAttribPointer(
638               programInfo.attribLocations.vertexPosition,
639               numComponents,
640               type,
```

```
              stride,
              offset
            );
            gl.enableVertexAttribArray(programInfo.attribLocations.vertexColor);
          }
          // Shader programını kullanma
          gl.useProgram(programInfo.program);
          gl.uniformMatrix4fv(
            programInfo.uniformLocations.projectionMatrix,
            false,
            projectionMatrix
          ); // Uniformları ayarlama
          gl.uniformMatrix4fv(
            programInfo.uniformLocations.modelViewMatrix,
            false,
            modelViewMatrix
          );

          {
            // Şekilleri çizme
            const offset = 0;
            const vertexCount = 66; // Toplam 66 vertex
            gl.drawArrays(gl.LINES, offset, vertexCount); // Çizgiler olarak çizme
          }
        }

        window.onload = main;
      </script>
    </body>
  </html>
```

Açıklamalar

Ana Fonksiyon (main): Canvas ve WebGL bağlamını başlatır, shader programını oluşturur ve sahneyi çizer.

Shader Programı Başlatma (initShaderProgram): Vertex ve fragment shader'larını oluşturur ve bağlar.

Shader Yükleme (loadShader): Shader kaynak kodunu yükler ve derler.

Tamponları Başlatma (initBuffers): Pozisyon ve renk verilerini tanımlar ve WebGL tamponlarına yükler.

Sahneyi Çizme (drawScene): Sahneyi temizler, dönüşüm matrislerini hesaplar ve çizim işlemini gerçekleştirir.

Sonuç

Bu proje, WebGL kullanarak basit bir metin çizimi örneğidir. Daha karmaşık şekiller ve animasyonlar için temel bir yapı sağlar.