var texture = gl.createTexture();

    gl.bindTexture(gl.TEXTURE\_2D, texture);

    // Fill the texture with a 1x1 red pixel.

    const texData = new Uint8Array([255, 0, 0, 255, 0, 255, 0, 255, 0, 0, 255, 255, 255, 255, 0, 255]);

    gl.texImage2D(gl.TEXTURE\_2D, 0, gl.RGBA, 2, 2, 0, gl.RGBA, gl.UNSIGNED\_BYTE, texData);

    //gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_MIN\_FILTER, gl.NEAREST\_MIPMAP\_NEAREST); // It is default

    gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_MIN\_FILTER, gl.LINEAR);

    gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_MAG\_FILTER, gl.LINEAR);

    gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_WRAP\_S, gl.MIRRORED\_REPEAT);

    gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_WRAP\_T, gl.MIRRORED\_REPEAT);

    // Asynchronously load an image

    /\*

    var image = new Image();

    image.src = "hylee\_128.png";

    image.addEventListener('load', function() {

        // Now that the image has loaded make copy it to the texture.

        gl.bindTexture(gl.TEXTURE\_2D, texture);

        gl.texImage2D(gl.TEXTURE\_2D, 0, gl.RGBA, gl.RGBA,gl.UNSIGNED\_BYTE, image);

        gl.generateMipmap(gl.TEXTURE\_2D);

        });

    \*/

var fragmentShaderSource = '\

            varying highp vec4 color; \

            varying mediump vec2 texCoord;\

            uniform sampler2D sampler2d;\

            void main(void) \

            { \

                gl\_FragColor = 0.0 \* color + 1.0 \* texture2D(sampler2d, texCoord); \

            }';

var vertexShaderSource = '\

            attribute highp vec4 myVertex; \

            attribute highp vec4 myColor; \

            attribute highp vec2 myUV; \

            uniform mediump mat4 mMat; \

            uniform mediump mat4 vMat; \

            uniform mediump mat4 pMat; \

            varying  highp vec4 color;\

            varying mediump vec2 texCoord;\

            void main(void)  \

            { \

                gl\_Position = pMat \* vMat \* mMat \* myVertex; \

                gl\_PointSize = 8.0; \

                color = myColor; \

                texCoord = myUV\*3.0; \

            }';

 gl.bindAttribLocation(gl.programObject, 0, "myVertex");

    gl.bindAttribLocation(gl.programObject, 1, "myColor");

    gl.bindAttribLocation(gl.programObject, 2, "myUV");

gl.vertexAttribPointer(0, 3, gl.FLOAT, gl.FALSE, 36, 0);

    gl.enableVertexAttribArray(1);

    gl.vertexAttribPointer(1, 4, gl.FLOAT, gl.FALSE, 36, 12);

    gl.enableVertexAttribArray(2);

    gl.vertexAttribPointer(2, 2, gl.FLOAT, gl.FALSE, 36, 28);

 var vertexShaderSource = '\

            attribute highp vec4 myVertex; \

            attribute highp vec4 myColor; \

            attribute highp vec2 myUV; \

            attribute highp vec3 myNormal; \

            uniform mediump mat4 mMat; \

            uniform mediump mat4 vMat; \

            uniform mediump mat4 pMat; \

            uniform mediump mat4 normalMat; \

            varying  highp vec4 color;\

            varying mediump vec2 texCoord;\

            varying highp vec3 v; \

            varying highp vec3 n; \

            void main(void)  \

            { \

                vec3 light; \

                vec4 light\_color; \

                light = vec3(1.0, 1.0, 1.0); \

                light\_color = vec4 (1.0, 1.0, 1.0, 1.0); \

                normalize(light); \

                n = vec3(normalMat \* vec4(myNormal, 1.0)); \

                normalize(n); \

                gl\_Position = pMat \* vMat \* mMat \* myVertex; \

                color = light\_color \* myColor \* max(dot(light, n), 0.3);  \

                texCoord = myUV\*2.0; \

            }';

gl.bindAttribLocation(gl.programObject, 3, "myNormal");

var mMatLocation = gl.getUniformLocation(gl.programObject, "mMat");

    var vMatLocation = gl.getUniformLocation(gl.programObject, "vMat");

    var pMatLocation = gl.getUniformLocation(gl.programObject, "pMat");

    var normalMatLocation = gl.getUniformLocation(gl.programObject, "normalMat");

 gl.uniformMatrix4fv(mMatLocation, gl.FALSE, mMat );

    gl.uniformMatrix4fv(vMatLocation, gl.FALSE, vMat );

    gl.uniformMatrix4fv(pMatLocation, gl.FALSE, pMat );

    gl.uniformMatrix4fv(normalMatLocation, gl.FALSE, normalMat );

  gl.bindBuffer(gl.ARRAY\_BUFFER, gl.vertexBuffer);

    gl.enableVertexAttribArray(0);

    gl.vertexAttribPointer(0, 3, gl.FLOAT, gl.FALSE, 48, 0);

    gl.enableVertexAttribArray(1);

    gl.vertexAttribPointer(1, 4, gl.FLOAT, gl.FALSE, 48, 12);

    gl.enableVertexAttribArray(2);

    gl.vertexAttribPointer(2, 2, gl.FLOAT, gl.FALSE, 48, 28);

    gl.enableVertexAttribArray(3);

    gl.vertexAttribPointer(3, 3, gl.FLOAT, gl.FALSE, 48, 36);