

CSE520
Samuel Marrujo
Professor Yu
Lab 02

Draw a square with Android

In this lab, we are to take triangles and create a yellow square using the Android libraries and using the Eclipse IDE. I have successfully completed this task, and here is a picture of the following:



5554:newavd



3G 9:05



HelloES1a



S1a/src/o...

MainActivity.java [Rea...

5554:newavd

Main Activity Code:

```
package opengl.helloesla;
import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.opengl.GLSurfaceView;
public class MainActivity extends Activity {
    private GLSurfaceView mGLView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Create a GLSurfaceView instance and set it
        // as the ContentView for this Activity.
        mGLView = new HelloESSurfaceView(this);
        setContentView(mGLView);
    }
    @Override
    protected void onPause() {
        super.onPause();
        // The following call pauses the rendering thread.
        mGLView.onPause();
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The following call resumes a paused rendering thread.
        mGLView.onResume();
    }
    class HelloESSurfaceView extends GLSurfaceView {
        public HelloESSurfaceView(Context context){
            super(context);
            // Set the Renderer for drawing on the GLSurfaceView
            setRenderer(new HelloESRenderer());
        }
    }
}
```

Renderer Code:

```
package opengl.helloesla;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.opengl.GLSurfaceView;

public class HelloESRenderer implements GLSurfaceView.Renderer {

    private FloatBuffer triangle;
    private FloatBuffer triangle2;
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
```

```

        // Set the background frame color to blue
        gl.glClearColor(0.0f, 0.0f, 0.9f, 1.0f);
        // initialize the triangle vertex array
        initShapes();
        initShapes2();
        // Enable use of vertex arrays
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
    }
    public void onDrawFrame(GL10 gl) {
        // Redraw background color
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        // Draw the triangle using green color
        gl.glColor4f(1.0f, 1.0f, 0.0f, 0.0f);
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, triangle);
        gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, triangle2);
        gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);
    }
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        gl.glViewport(0, 0, width, height);
    }
    private void initShapes(){
        float vertices[] = {
            // (x, y, z) of triangle
            -0.3f, -0.3f, 0,
            0.3f, 0.3f, 0,
            -0.3f, 0.3f, 0
        };

        // initialize vertex Buffer for triangle
        // argument=(# of coordinate values * 4 bytes per float)
        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
        // use the device hardware's native byte order
        vbb.order(ByteOrder.nativeOrder());
        // create a floating point buffer from the ByteBuffer
        triangle = vbb.asFloatBuffer();
        // add the coordinates to the FloatBuffer
        triangle.put(vertices);
        // set the buffer to read the first vertex coordinates
        triangle.position(0);
    }
    private void initShapes2(){
        float vertices[] = {
            // (x, y, z) of triangle 2
            -0.3f, -0.3f, 0,
            0.3f, -0.3f, 0,
            0.3f, 0.3f, 0
        };

        // initialize vertex Buffer for triangle
        // argument=(# of coordinate values * 4 bytes per float)
        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
        // use the device hardware's native byte order
        vbb.order(ByteOrder.nativeOrder());
        // create a floating point buffer from the ByteBuffer
        triangle2 = vbb.asFloatBuffer();
        // add the coordinates to the FloatBuffer
        triangle2.put(vertices);
        // set the buffer to read the first vertex coordinates
    }

```

```
    triangle2.position(0);  
  }  
}
```