Andrea Padula
OpenGL ES

What is OpenGL ES?

- OpenGL ES stands for OpenGL for Embedded System. It is a subset of the OpenGL application programming interface (API).
  - Designed for embedded systems
    - mobile phones.Used in a wide variety of devices, not just Android (iPad, iPhone, Blackberry, symbian, Nintendo3DS)
  - Lightweight interface compare to OpenGL
  - Differences from standard OpenGL
    - Programmers need to compute projection matrix & transformation matrix

Version of OpenGL ES

- In OpenGL ES 1.0/1.1
  - Use fixed function pipeline
    - No glBegin() and glEnd() for OpenGL ES 2.0 and above
- In OpenGL ES 2.0 and above
  - Use programmable function pipeline

OpenGL for Android

What platform? Android Studio

Android Studio

What language? Java

Java™

OpenGL for Android

There are two way to work with android openGL ES:

1.Framework APIandroid.opengl package

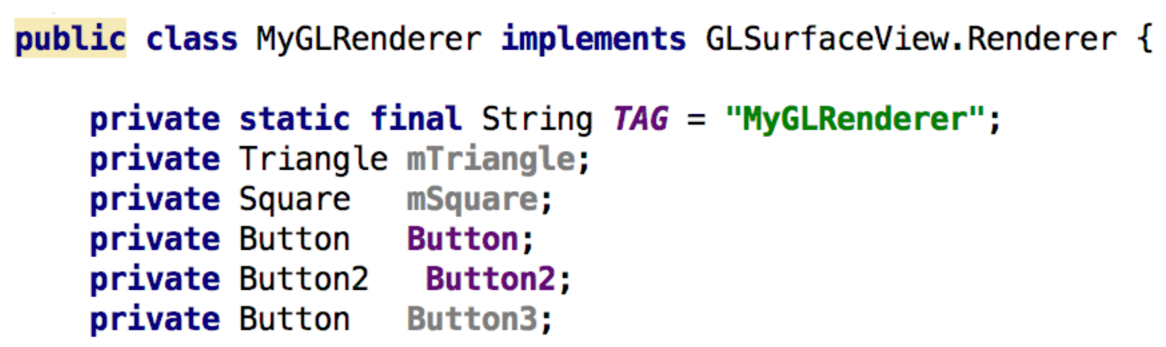
2. Android Native Development Kit used to build portions of apps in native code in C or C++

How to enable OpenGL?

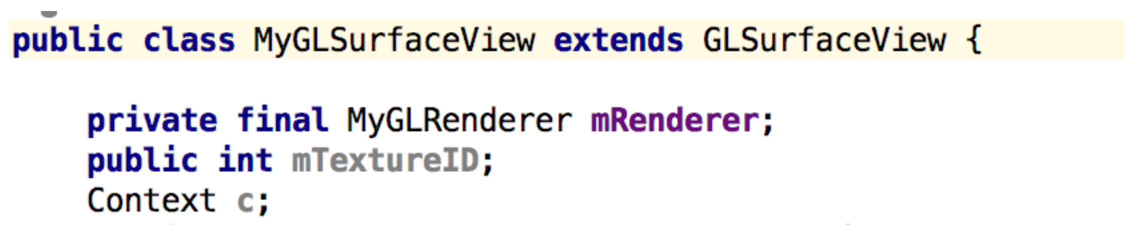
```xml
<!-- Tell the system this application requires OpenGL ES 3.1. -->
<uses-feature android:glEsVersion="0x00030002" android:required="true" />
```

Requires two classes

1.GLSurfaceView
2.GLSsurfaceview.Render



```
public class MyGLRenderer implements GLSurfaceView.Renderer {

    private static final String TAG = "MyGLRenderer";
    private Triangle mTriangle;
    private Square   mSquare;
    private Button   Button;
    private Button2   Button2;
    private Button   Button3;
```



```
public class MyGLSurfaceView extends GLSurfaceView {

    private final MyGLRenderer mRenderer;
    public int mTextureID;
    Context c;
```

GLSurfaceView

This is :

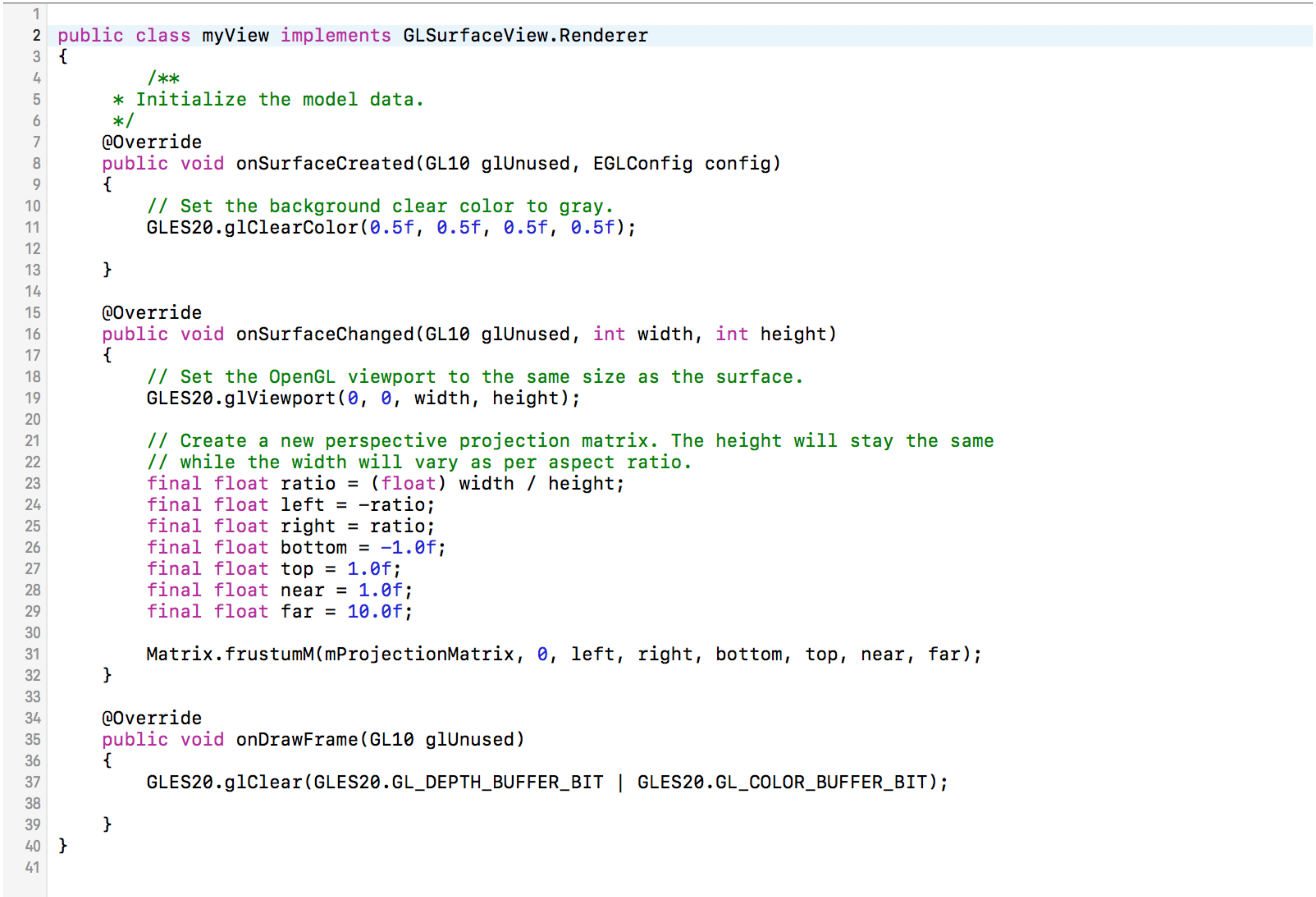
- where draw and manipulate objects. We can consider this class our draw function in our homework.
- where we implement touch listeners and respond to touch event.

GLSurfaceView.Render

This is an interface where we need to implements 3 methods:

1. onSurfaceCreated() for initializing GL graphics objects. Set the background color.
2. onDrawFrame this creates movement and animation. We clear the buffer color and depth
3. onSurfaceChanged called when size of view changes. Set coordinate System to normalized device coordinates

Simple Renderer



```java

public class myView implements GLSurfaceView.Renderer
{
        /**
     * Initialize the model data.
     */
    @Override
    public void onSurfaceCreated(GL10 glUnused, EGLConfig config)
    {
        // Set the background clear color to gray.
        GLES20.glClearColor(0.5f, 0.5f, 0.5f, 0.5f);

    }

    @Override
    public void onSurfaceChanged(GL10 glUnused, int width, int height)
    {
        // Set the OpenGL viewport to the same size as the surface.
        GLES20.glViewport(0, 0, width, height);

        // Create a new perspective projection matrix. The height will stay the same
        // while the width will vary as per aspect ratio.
        final float ratio = (float) width / height;
        final float left = -ratio;
        final float right = ratio;
        final float bottom = -1.0f;
        final float top = 1.0f;
        final float near = 1.0f;
        final float far = 10.0f;

        Matrix.frustumM(mProjectionMatrix, 0, left, right, bottom, top, near, far);
    }

    @Override
    public void onDrawFrame(GL10 glUnused)
    {
        GLES20.glClear(GLES20.GL_DEPTH_BUFFER_BIT | GLES20.GL_COLOR_BUFFER_BIT);

    }
}

```

Vertex and Fragment Shader

```
private final String vertexShaderCode =
        // This matrix member variable provides a hook to manipulate
        // the coordinates of the objects that use this vertex shader
        "uniform mat4 uMVPMatrix;" +
        "attribute vec4 vPosition;" +
        "void main() {" +
        // the matrix must be included as a modifier of gl_Position
        // Note that the uMVPMatrix factor *must be first* in order
        // for the matrix multiplication product to be correct.
        "  gl_Position = uMVPMatrix * vPosition;" +
        "}";

private final String fragmentShaderCode =
        "precision mediump float;" +
        "uniform vec4 vColor;" +
        "void main() {" +
        "  gl_FragColor = vColor;" +
        “}";
```

How to link everything together?

```
int vertexShader = MyGLRenderer.loadShader(
                GLES30.GL_VERTEX_SHADER, vertexShaderCode);
        int fragmentShader = MyGLRenderer.loadShader(
                GLES30.GL_FRAGMENT_SHADER,
fragmentShaderCode);

        mProgram = GLES30.glCreateProgram();
// create empty OpenGL Program
        GLES30.glAttachShader(mProgram, vertexShader);
// add the vertex shader to program
        GLES30.glAttachShader(mProgram, fragmentShader);
// add the fragment shader to program
        GLES30.glLinkProgram(mProgram);
// create OpenGL program executables
```

Draw a shape

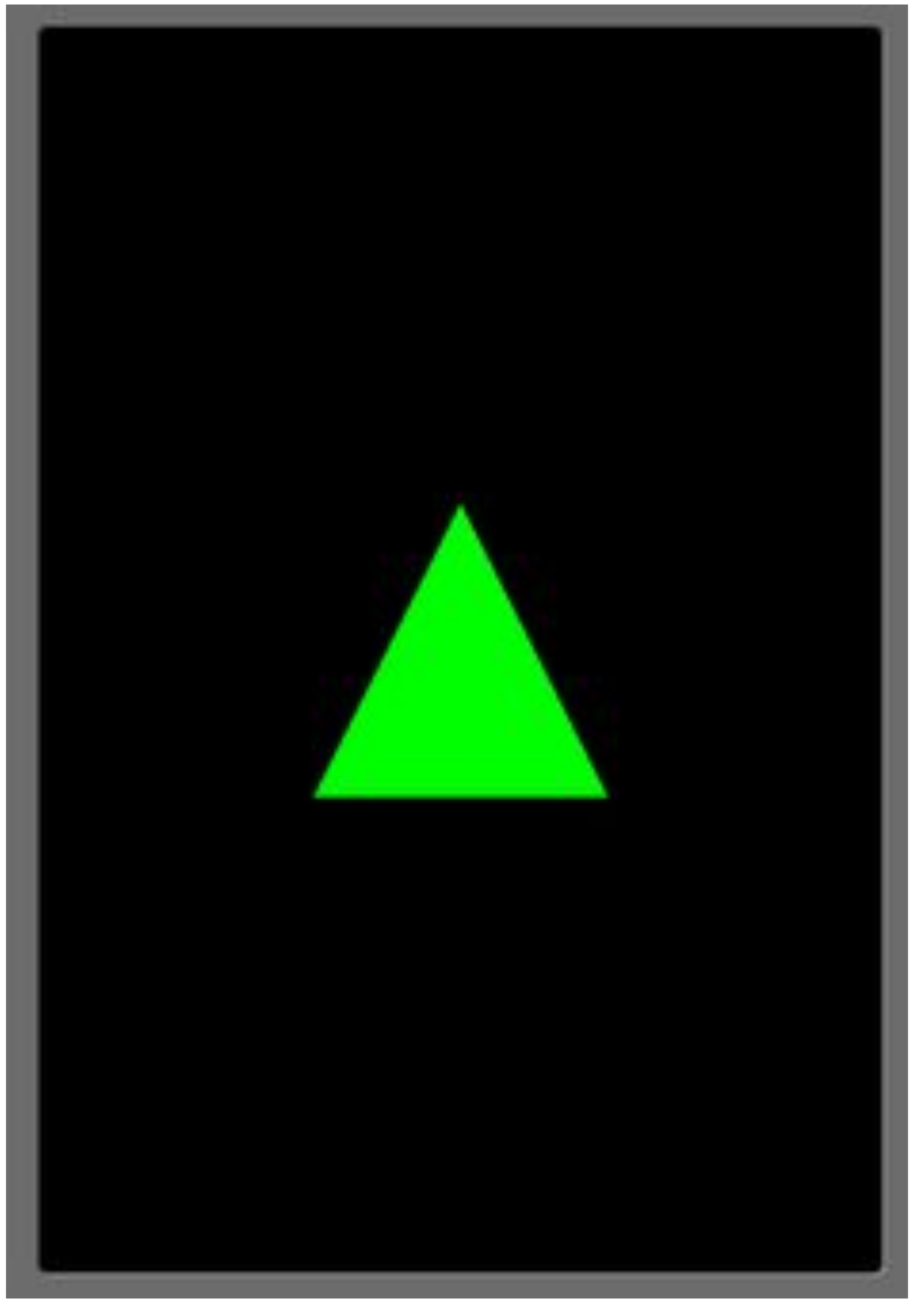
- Draw a shape using a OpenGL call like

```
GLES32.glDrawArrays(GLES32.GL_TRIANGLES, 0, vertexCount);
```

- We must define the vertices of our shape
- (X,Y,Z) coordinate system
- (0,0,0) center of out screen
- Normalize coordinates system so (-1,-1,0) is bottom

```
public void draw(float[] mvpMatrix) {
        // Add program to OpenGL environment
        GLES30.glUseProgram(mProgram);

        // get handle to vertex shader's vPosition member
        mPositionHandle = GLES30.glGetAttribLocation(mProgram, "vPosition");

        // Enable a handle to the triangle vertices
        GLES30.glEnableVertexAttribArray(mPositionHandle);

        // Prepare the triangle coordinate data
        GLES30.glVertexAttribPointer(
                mPositionHandle, COORDS_PER_VERTEX,
                GLES30.GL_FLOAT, false,
                vertexStride, vertexBuffer);

        // get handle to fragment shader's vColor member
        mColorHandle = GLES30.glGetUniformLocation(mProgram, "vColor");

        // Set color for drawing the triangle
        GLES30.glUniform4fv(mColorHandle, 1, color, 0);

        // get handle to shape's transformation matrix
        mMVPMatrixHandle = GLES30.glGetUniformLocation(mProgram, "uMVPMatrix");
        MyGLRenderer.checkGlError("glGetUniformLocation");

        // Apply the projection and view transformation
        GLES30.glUniformMatrix4fv(mMVPMatrixHandle, 1, false, mvpMatrix, 0);
        MyGLRenderer.checkGlError("glUniformMatrix4fv");

        // Draw the triangle
        GLES30.glDrawArrays(GLES30.GL_TRIANGLES, 0, vertexCount);

        // Disable vertex array
        GLES30.glDisableVertexAttribArray(mPositionHandle);
    }
```

RESULT



DEMO TIME !!!!!