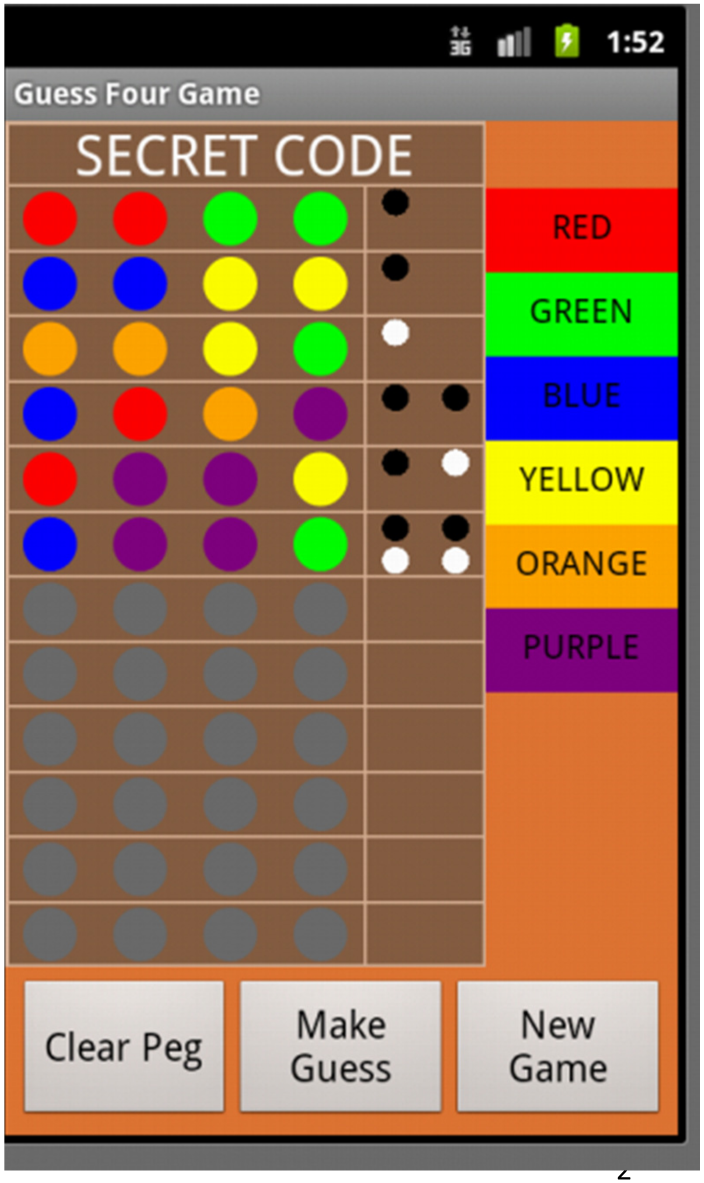
CS378 - Mobile Computing

3D Graphics

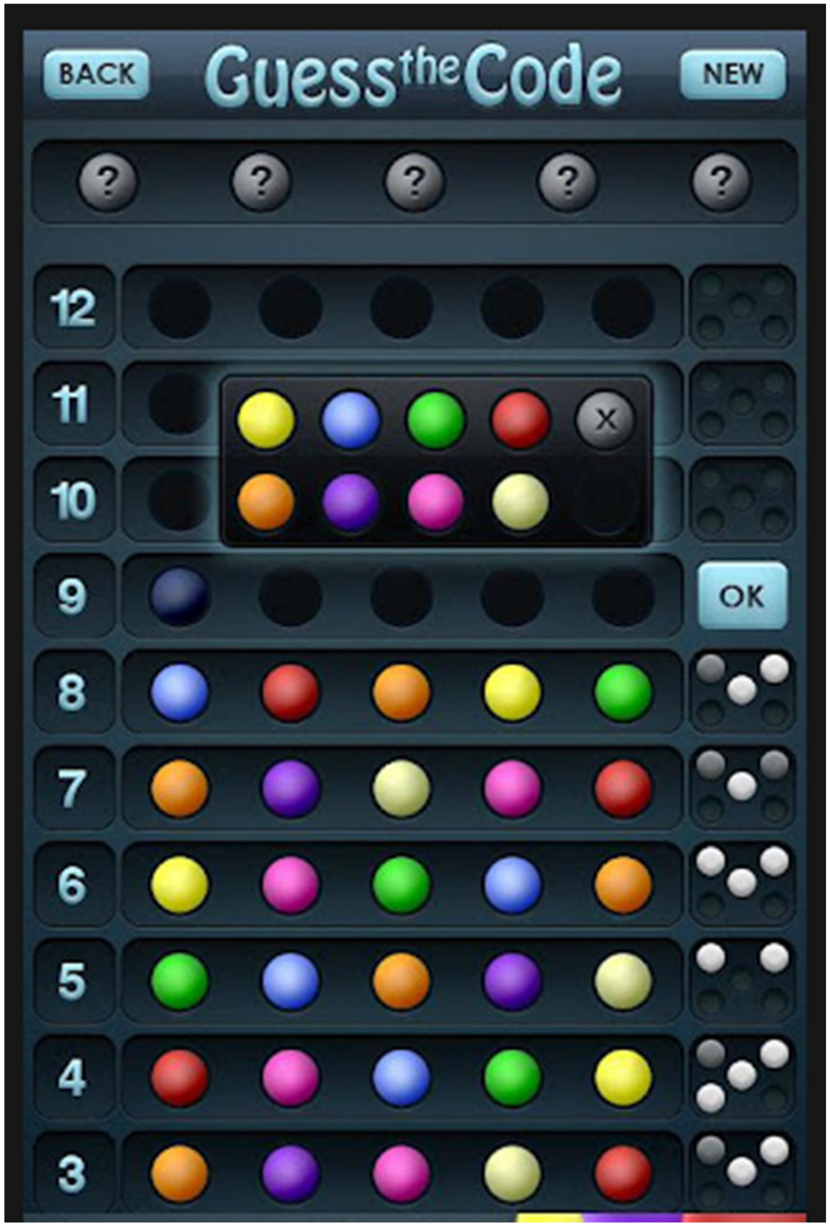
2D Graphics

- android.graphics library for 2D graphics (not Java AWT and Swing)
- classes such as Canvas, Drawable, Bitmap, and others to create 2D graphics
- Various attempts to make two d graphics appear more "lifelike" and 3 dimensional

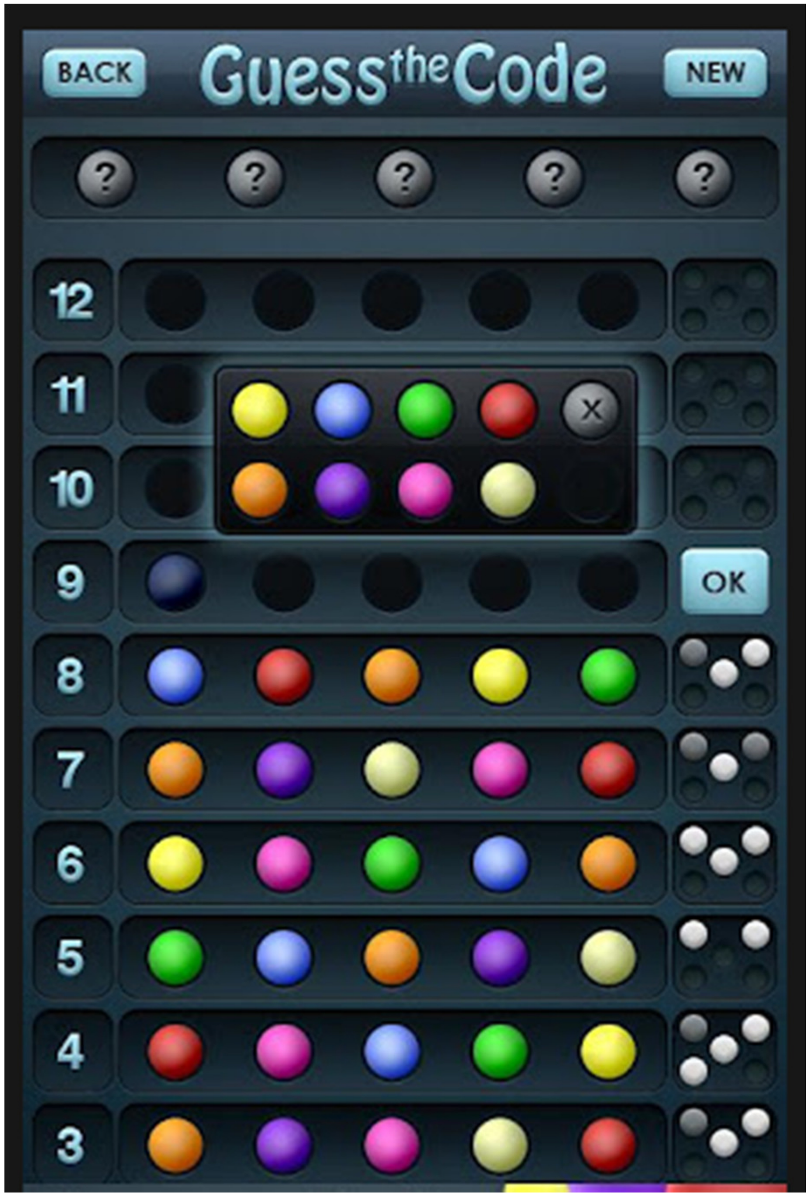


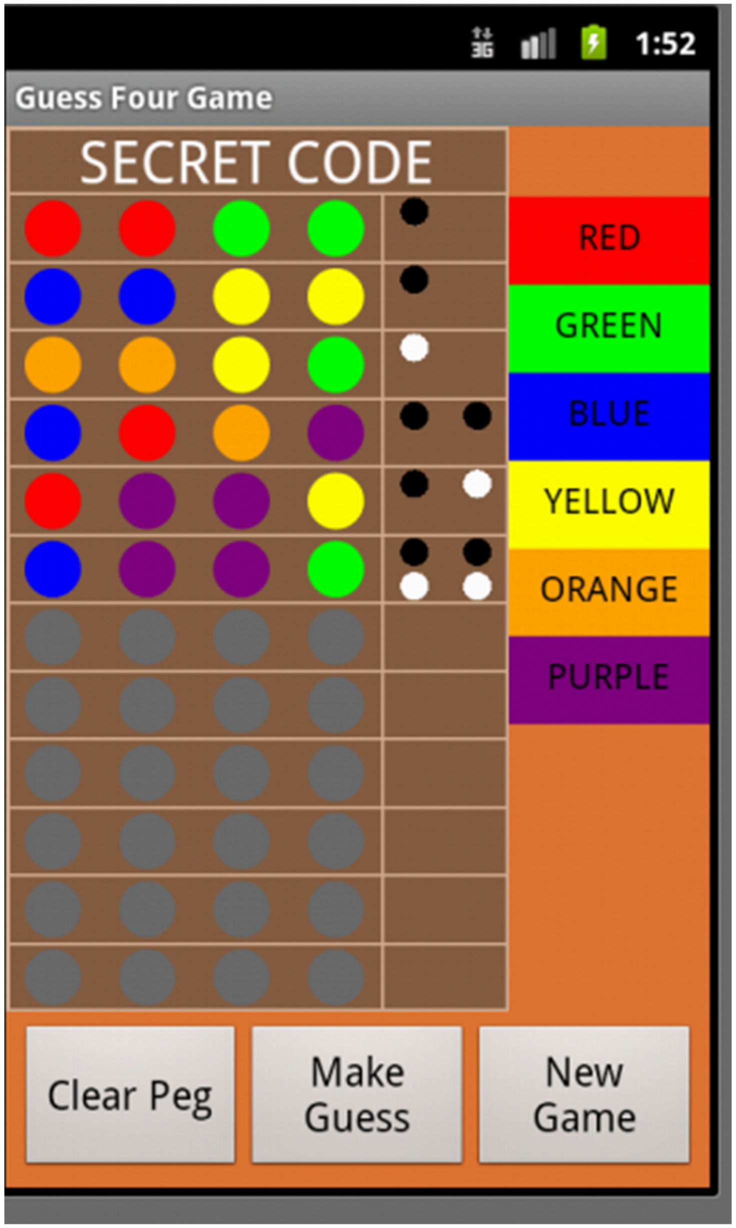
Gradients

- Gradient Paints can add depth to 2d primitives
- Notice the gradient paint on the pegs and shading on numbers

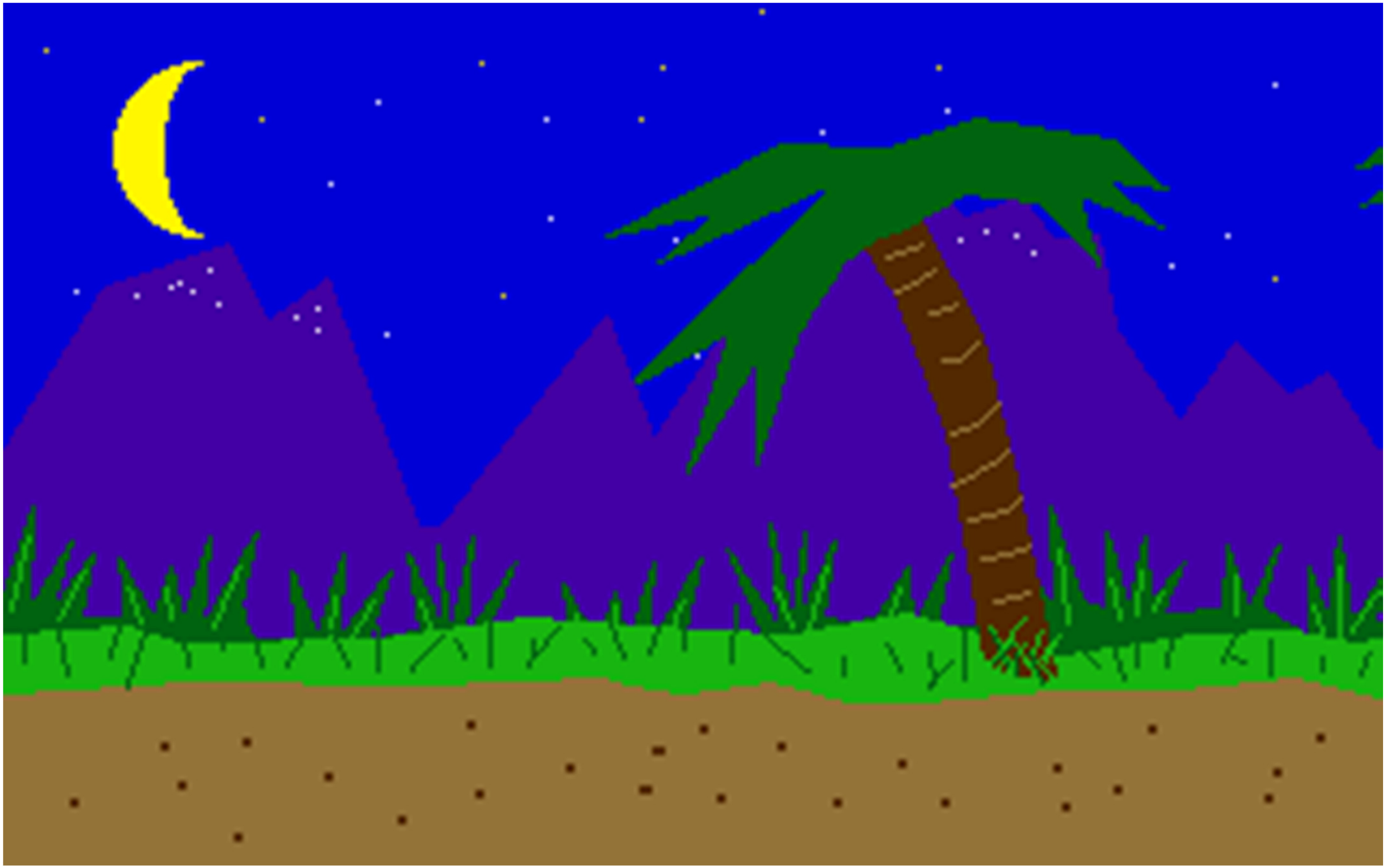


2D Graphics


BACK
GuesstheCode
NEW
12
11
10
9
OK
8
7
6
5
4
3

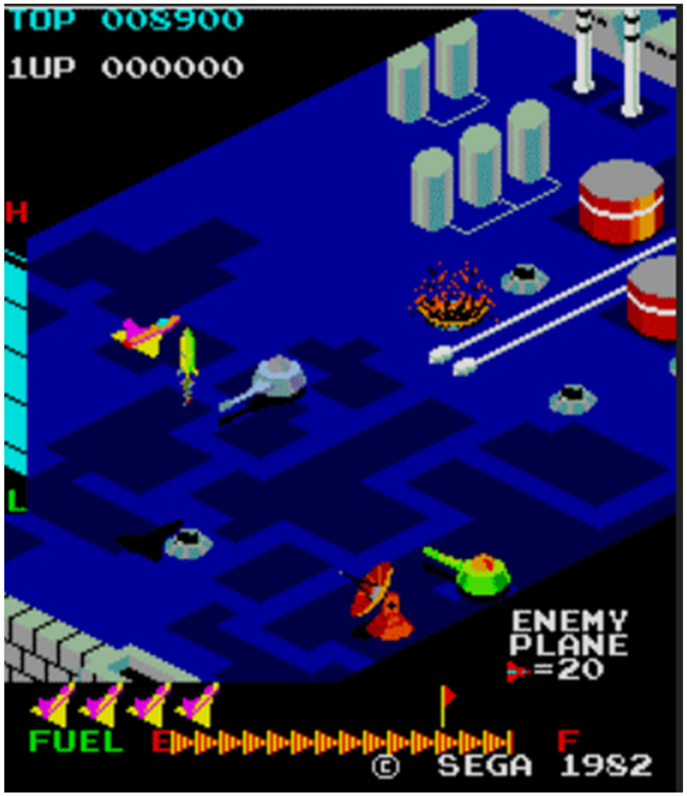

1:52
Guess Four Game
SECRET CODE
RED
GREEN
BLUE
YELLOW
ORANGE
PURPLE
Clear Peg
Make Guess
New Game

Parallax Scrolling Example

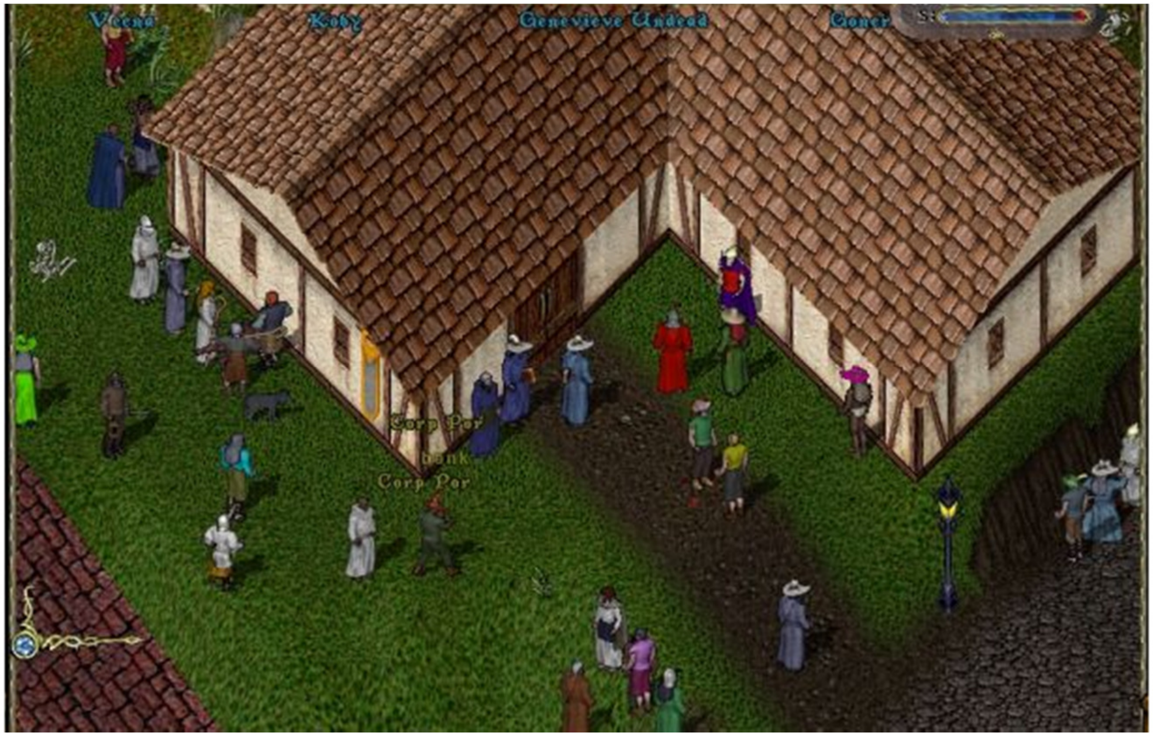


2.5D

- Isometric Graphics
- "rotate" object to reveal details on the side



Zaxxon



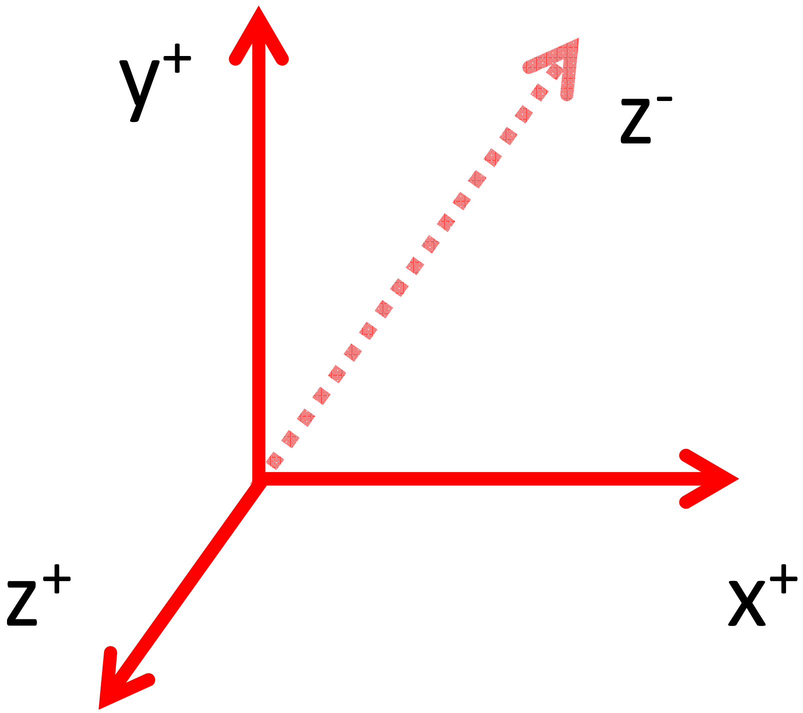
Ultima Online

3D Graphics

- Create 3D model
  - a small scene or a large world
- Model rendered into a 2D projection
- model includes
  - objects (boxes, cones, cylinders, sphere, user defined models)
  - lighting
  - cameras
  - textures
  - dynamic behaviors

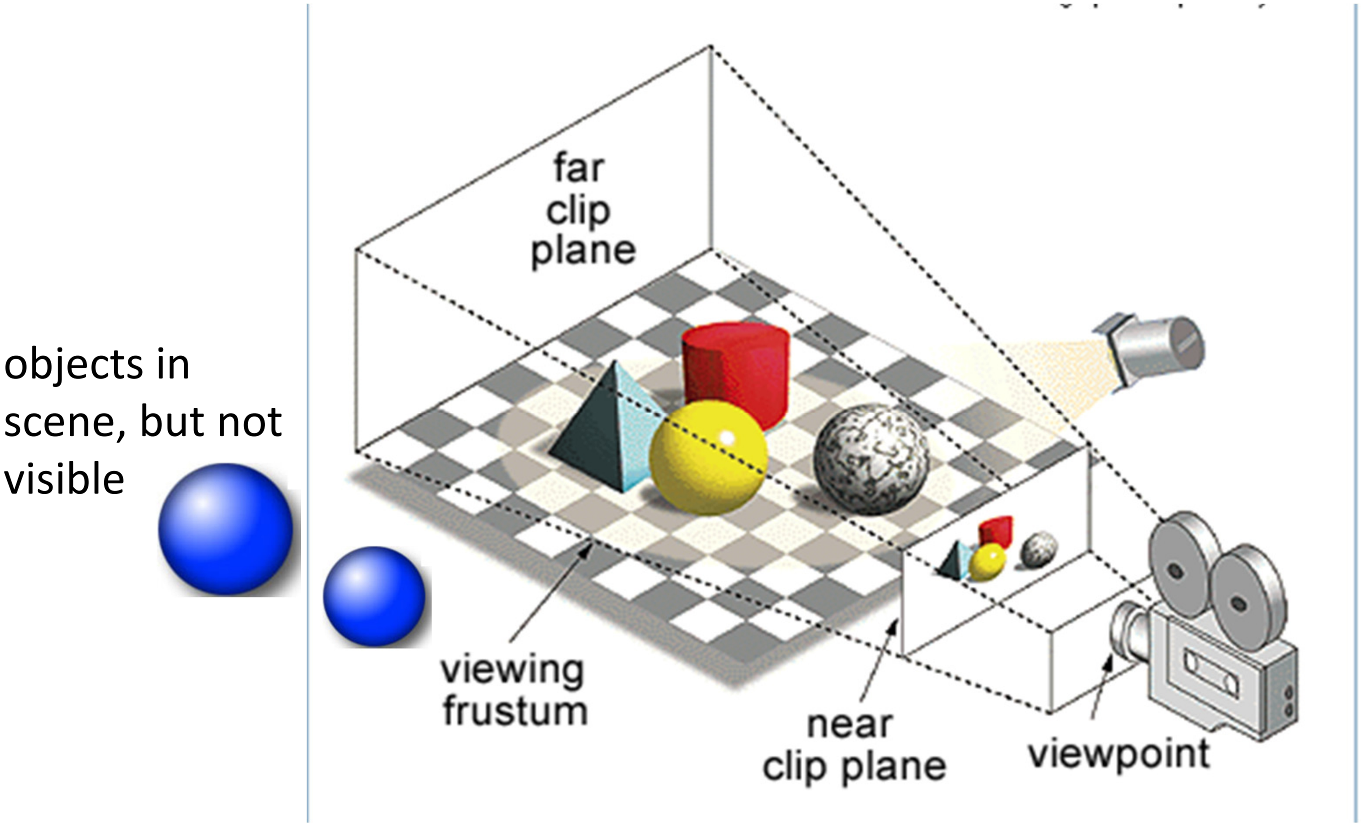
3D Coordinate System

- x and y as expected (positive y is up, not down as in 2d graphics
- z axis - positive z is out of screen, negative z is into screen



Visual Portion

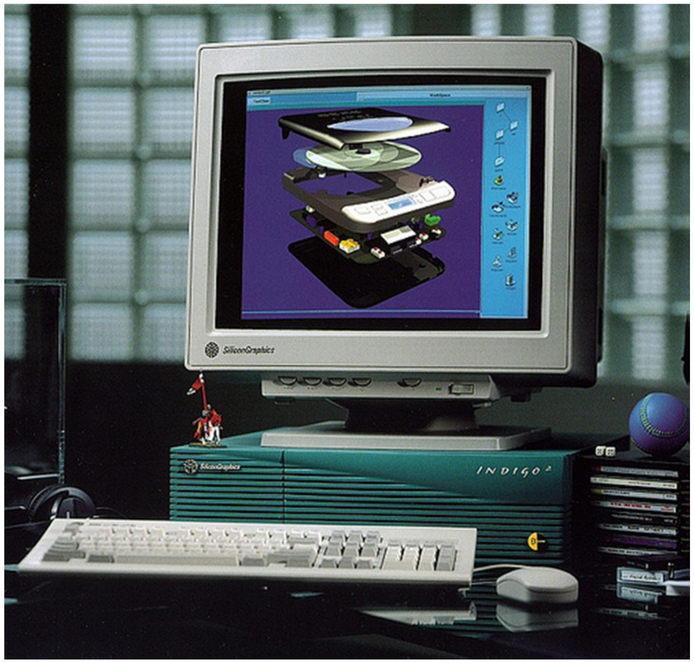
- Portion of 3D Scene that is rendered is contained in a *frustum* *(pro: frəstəm)*
  - a pyramid or cone with its top cut off



objects in scene, but not visible

OpenGL

- Developed by Silicon Graphics Inc.
  - developer of high end graphics systems and machines in 80s and 90s
- Integrated Raster Imaging System Graphics Library
  - 1992 OpenGL
  - maintained by non profit Khronos Group



OpenGL

- low level, procedural API
  - programmer responsible for defining steps to create and render (show) a scene
- alternatives use a scene graph where programmer describes scene and actions (behaviors) and library manages the details of rendering it
  - Example of Graphics libraries that use Scene Graphs: Java3D, Acrobat 3D, AutoCAD, CorelDRAW, RenderMan (Pixar)

OpenGL ES

- ES = Embedded Systems
- Used in a wide variety of devices, not just Android
  - iPad, iPhone, Blackberry, symbian, Nintendo3DS, Playstation 3, Web GL
- OpenGL version ES 2.0 API supported in Android 2.2 and higher (API levels 8 and higher)
  - prior versions of Android support ES 1.1
- emulator DOES NOT support ES 2.0

Android and OpenGL ES

- two ways of working with GL:
  - through the framework APIandroid.opengl package
  - via the Android Native Development Kit (NDK)
    - companion tool to Android SDK to build portions of apps in native code in C or C++
- Required Android classes for first approach:
  - GLSurfaceView and GLSurfaceView.Renderer

GLSurfaceView

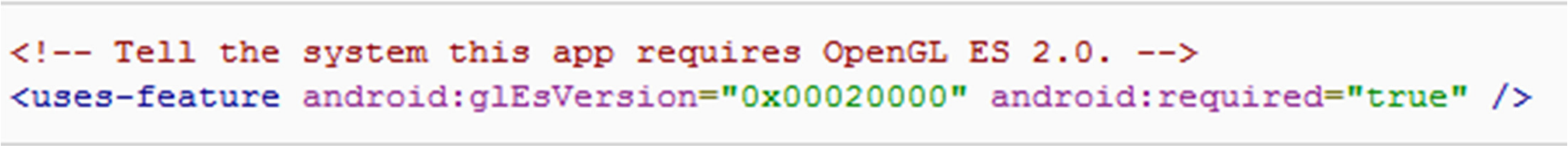
- Similar to SurfaceView
- draw and manipulate objects using Open GL API calls
- to respond to touch screen events subclass GLSurfaceView and implement touch listeners

GLSurfaceView.Renderer

- An interface
- Must implement these methods:
  - onSurfaceCreated for actions that only happen once such as initializing GL graphics objects
  - onDrawFrame() work horse method to create movement and animation
  - onSurfacechanged() called when size of view changes or orientation

Manifest Requirements

- To use OpenGL ES 2.0 (Android 2.0 and later)



```
<!-- Tell the system this app requires OpenGL ES 2.0. -->
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
```

- if app uses texture compression formats must declare which formats application supports
  - <support-gl-texture>

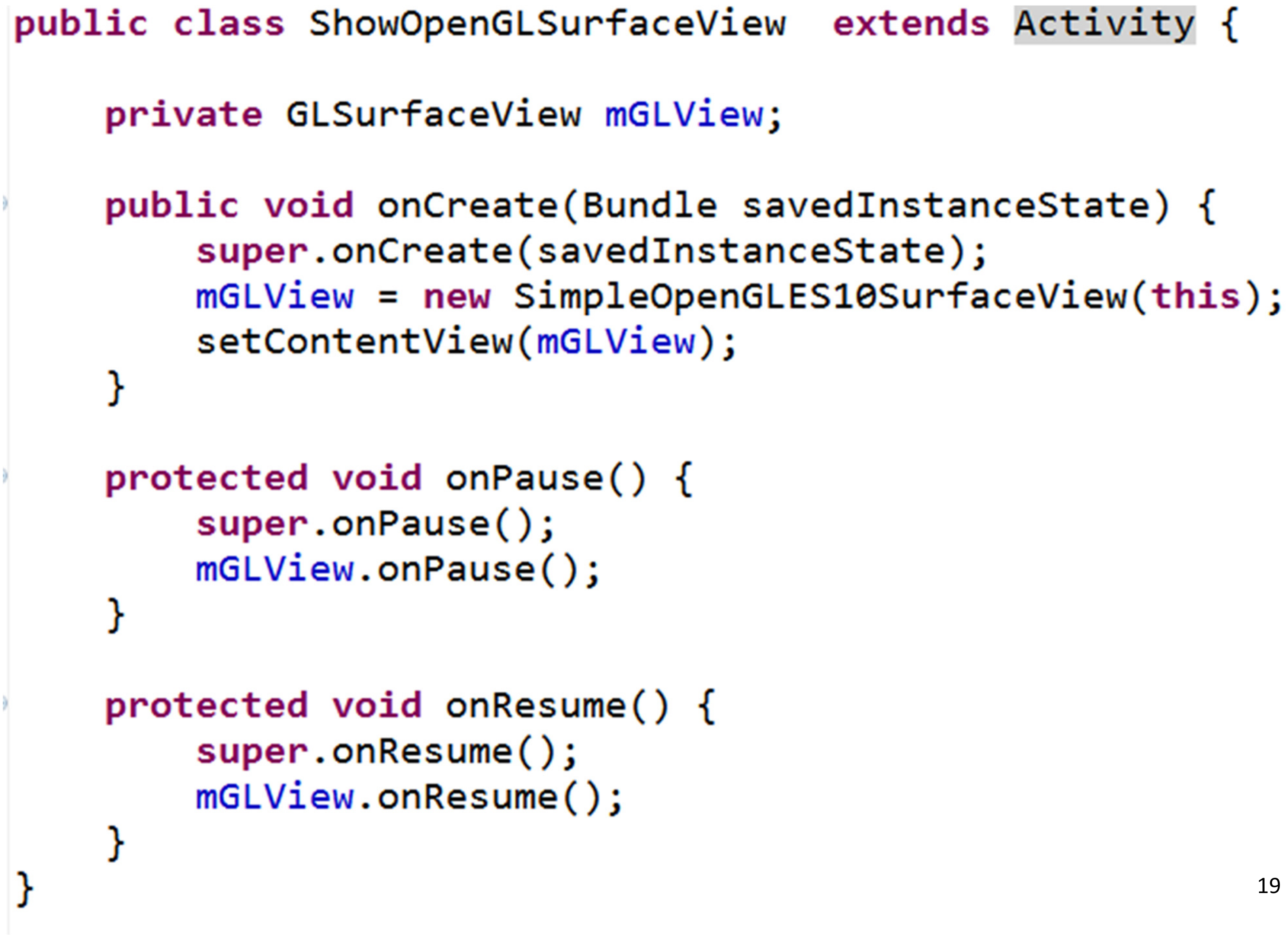
Steps to Use OpenGL

- Create activity using GLSurfaceView and GLSurfaceView.Renderer
- Create and draw graphics objects
- define projection for screen geometry to correct for non square pixels
- define a camera view
- perform actions to animate objects
- make view touch interactive if desired

Sample Program

- Demonstrate set up of required elements
- draw and rotate a 3d object (a cube)
- Create Simple Activity that has a GLSurfaceView as its content view
- To draw objects must implement GLSurfaceView.Renderer

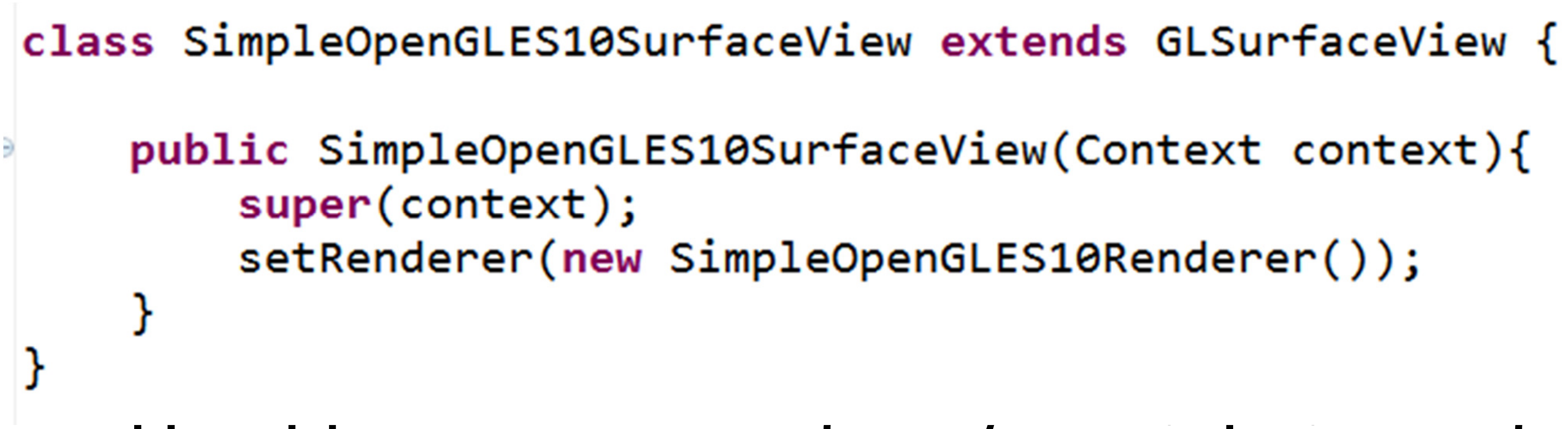
Activity



```
public class ShowOpenGLSurfaceView  extends Activity {

    private GLSurfaceView mGLView;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mGLView = new SimpleOpenGLES10SurfaceView(this);
        setContentView(mGLView);
    }

    protected void onPause() {
        super.onPause();
        mGLView.onPause();
    }

    protected void onResume() {
        super.onResume();
        mGLView.onResume();
    }
}
```

GLSurfaceView

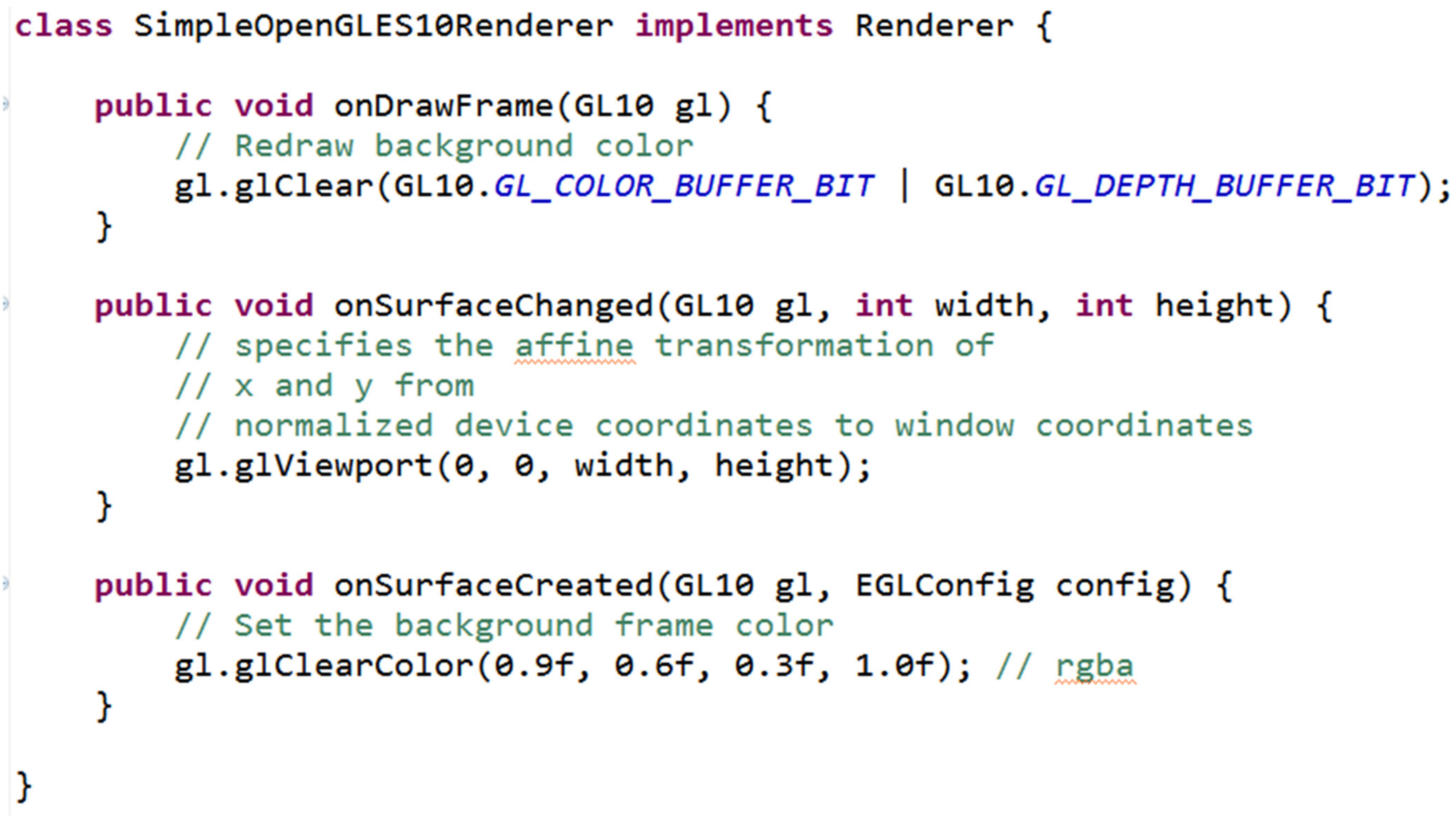
- Shell of class



```
class SimpleOpenGLES10SurfaceView extends GLSurfaceView {

    public SimpleOpenGLES10SurfaceView(Context context){
        super(context);
        setRenderer(new SimpleOpenGLES10Renderer());
    }
}
```

- Used to manage surface (special piece of memory), manage EGL display (embedded graphics library, renders on thread decoupled from I thread, and more

Skeleton Renderer



```
class SimpleOpenGLES10Renderer implements Renderer {

    public void onDrawFrame(GL10 gl) {
        // Redraw background color
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
    }

    public void onSurfaceChanged(GL10 gl, int width, int height) {
        // specifies the affine transformation of
        // x and y from
        // normalized device coordinates to window coordinates
        gl.glViewport(0, 0, width, height);
    }

    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        // Set the background frame color
        gl.glClearColor(0.9f, 0.6f, 0.3f, 1.0f); // rgba
    }

}
```

OpenGL Documentation

- Android Documentation for GL10 list constants and methods but have no other useful information
- Check the OpenGL ES documentation
- http://www.khronos.org/opengles/sdk/1.1/docs/man/

Low Level Graphics Libraries

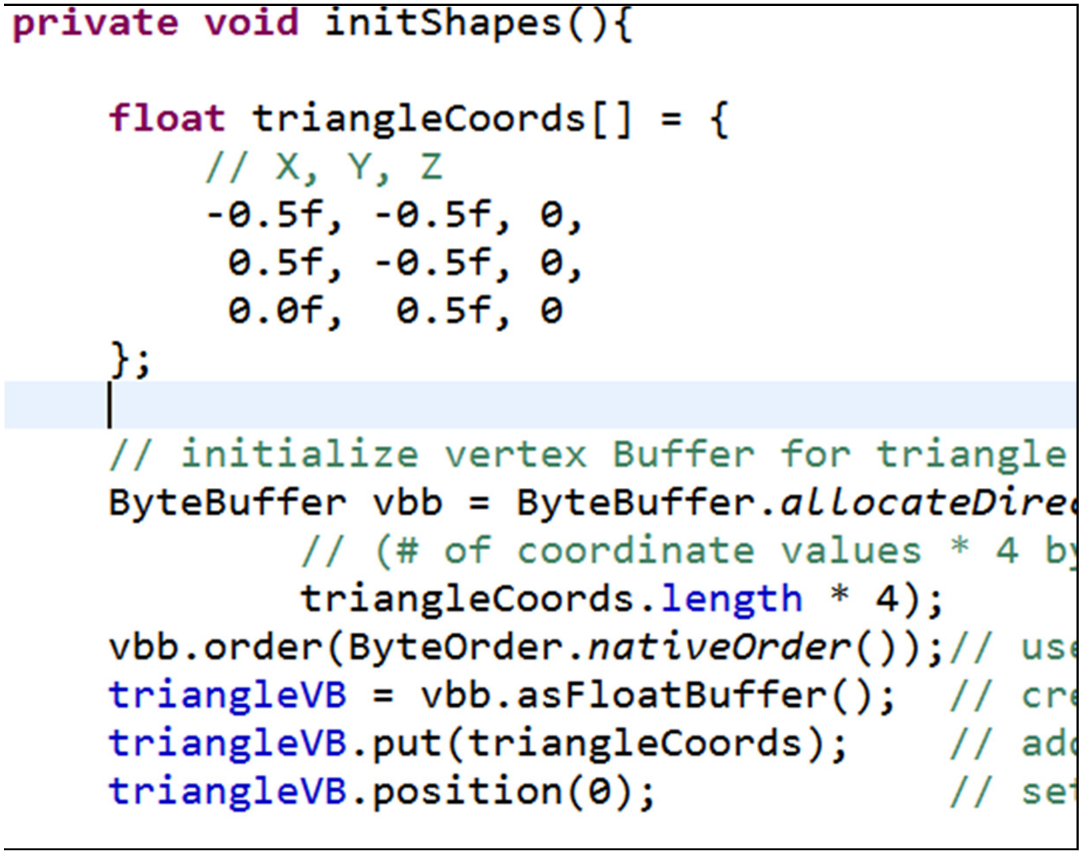
- "What makes the situation worse is that the highest level CS course I've ever taken is cs4, and quotes from the graphics group startup readme like '*these paths are abstracted as being the result of a topological sort on the graph of ordering dependencies for the entries*' make me lose consciousness in my chair and bleed from the nose."

  -mgrimes, Graphics problem report 134

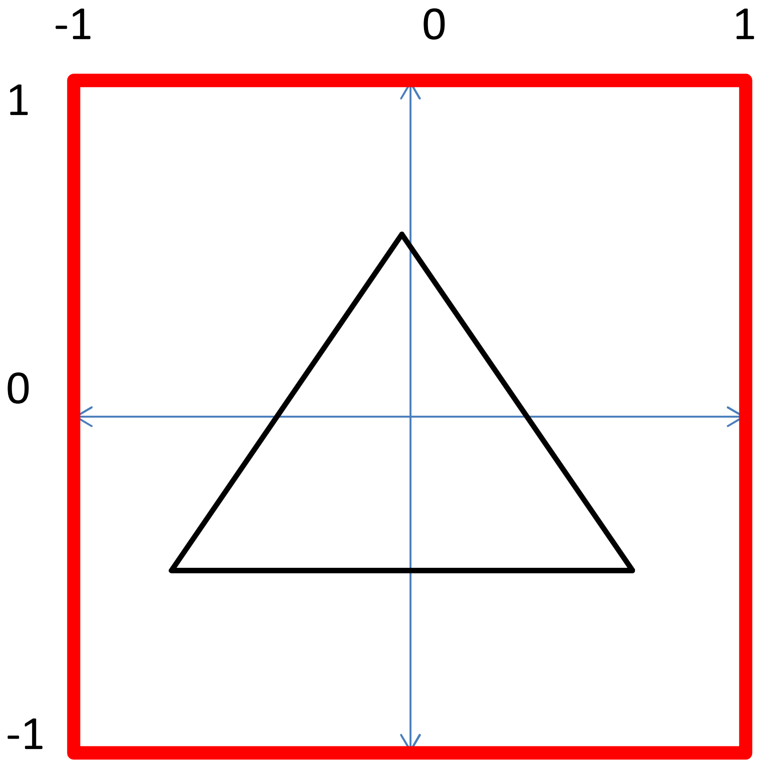
Draw a Shape

- Draw a simple, flat Triangle using OpenGL
- (X,Y,Z) coordinate system
- (0, 0, 0) center of frame
- (1, 1, 0) is top right corner of frame
- (-1, -1, 0) is bottom left corner of frame
- must define vertices of our triangle

Define Triangle

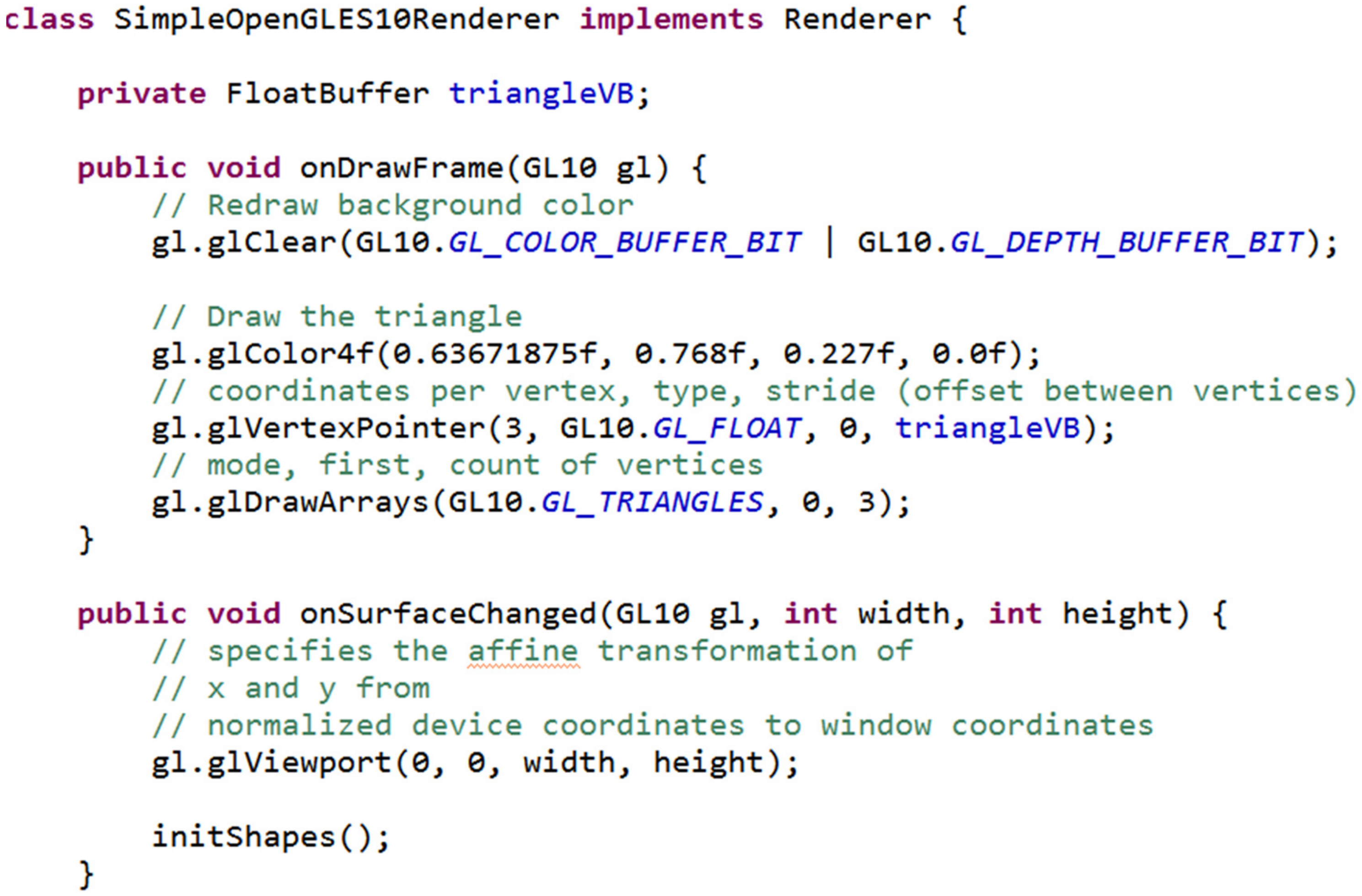


```
private void initShapes(){

    float triangleCoords[] = {
        // X, Y, Z
        -0.5f, -0.5f, 0,
         0.5f, -0.5f, 0,
         0.0f,  0.5f, 0
    };

    // initialize vertex Buffer for triangle
    ByteBuffer vbb = ByteBuffer.allocateDire
            // (# of coordinate values * 4 by
            triangleCoords.length * 4);
    vbb.order(ByteOrder.nativeOrder());// use
    triangleVB = vbb.asFloatBuffer();  // cre
    triangleVB.put(triangleCoords);    // add
    triangleVB.position(0);            // set
```



Draw Triangle

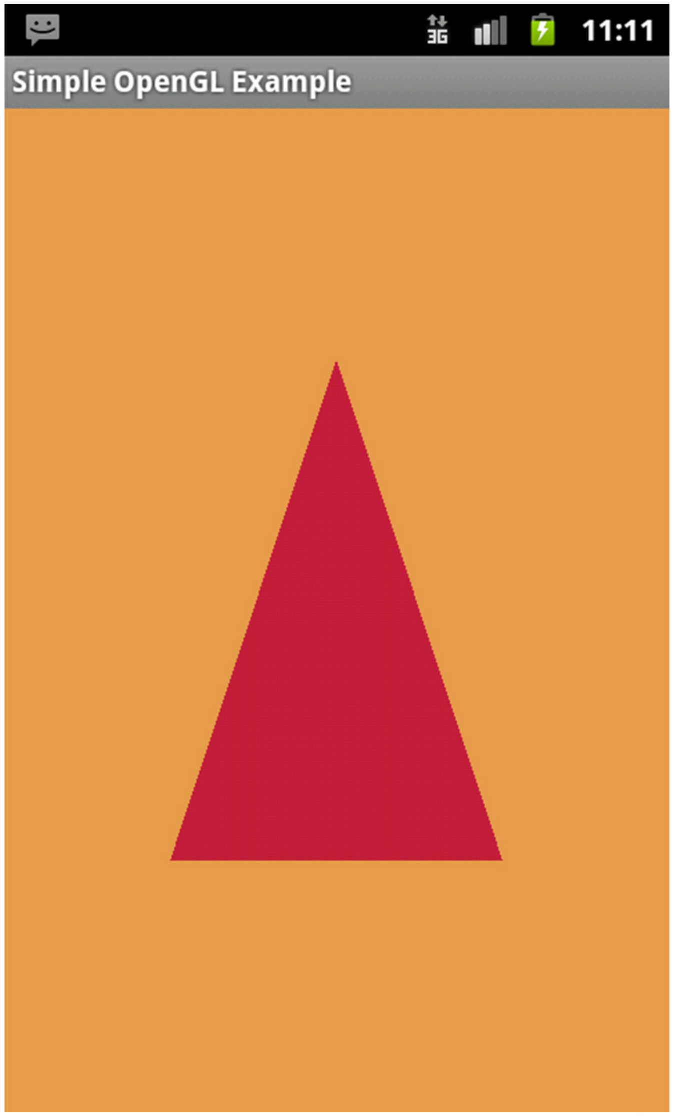
- init OpenGL to use vertex arrays
- call drawing API to draw triangle



```
class SimpleOpenGLES10Renderer implements Renderer {

    private FloatBuffer triangleVB;

    public void onDrawFrame(GL10 gl) {
        // Redraw background color
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);

        // Draw the triangle
        gl.glColor4f(0.63671875f, 0.768f, 0.227f, 0.0f);
        // coordinates per vertex, type, stride (offset between vertices)
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, triangleVB);
        // mode, first, count of vertices
        gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);
    }

    public void onSurfaceChanged(GL10 gl, int width, int height) {
        // specifies the affine transformation of
        // x and y from
        // normalized device coordinates to window coordinates
        gl.glViewport(0, 0, width, height);

        initShapes();
    }
```

Result

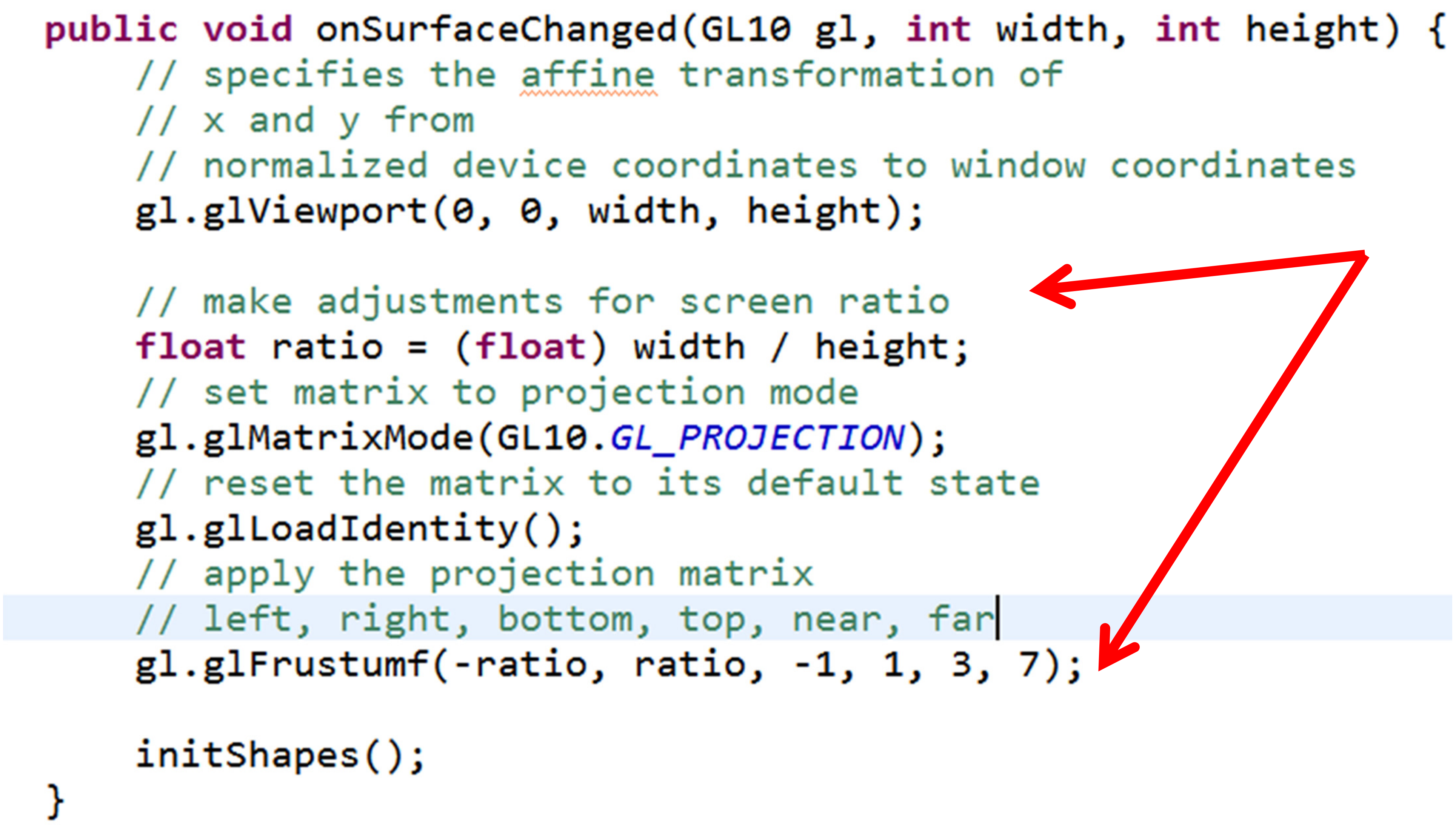
- oooo, ahhhh
- Graphics coordinate system assumes a square but mapped to a rectangular frame



Correcting Projection

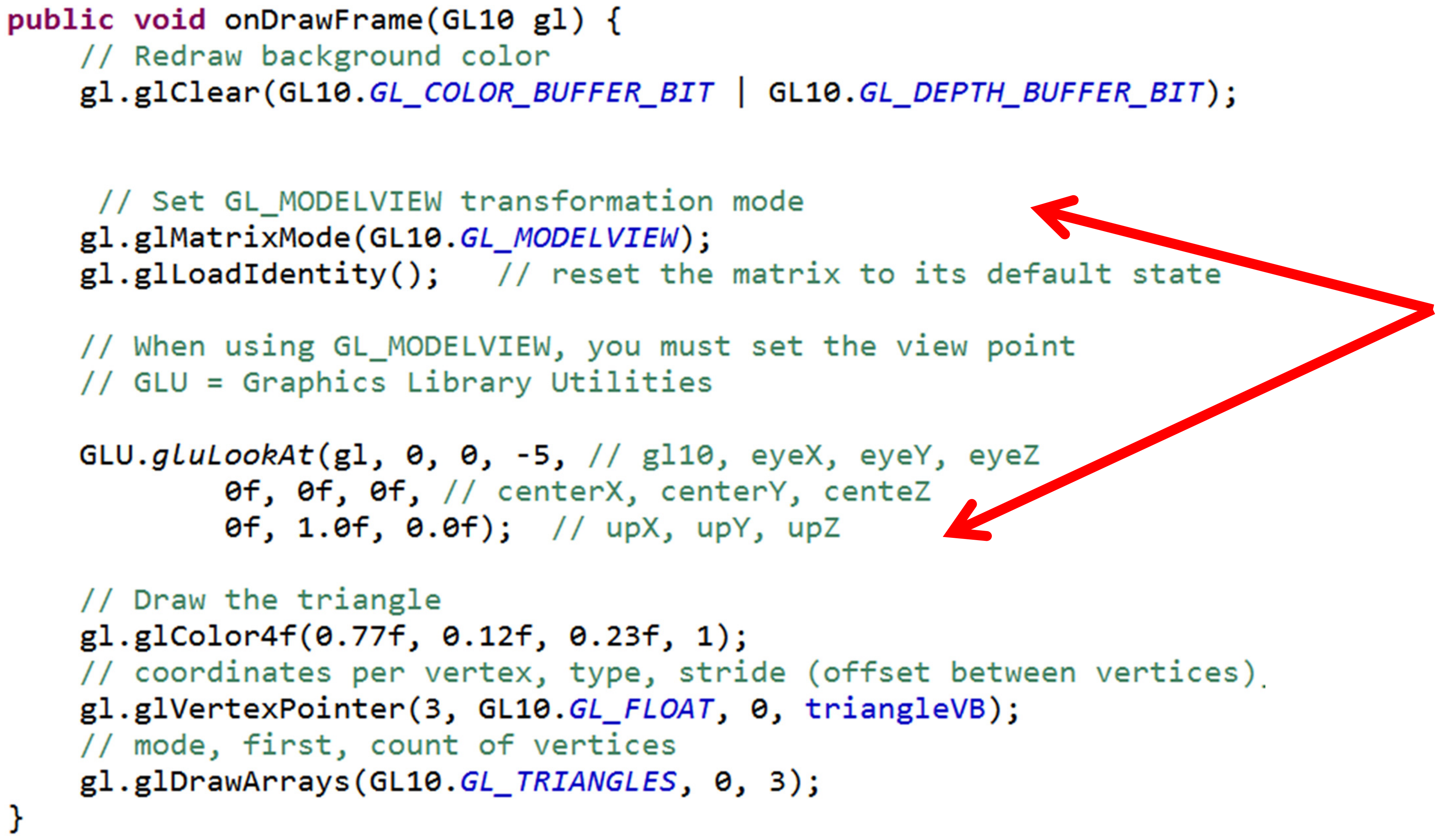
- Apply an OpenGL projection view and camera (eye point) to transform coordinates of the triangle
  - "correct" the position onSurfaceChanged and onDrawframe()

onSurfaceChanged



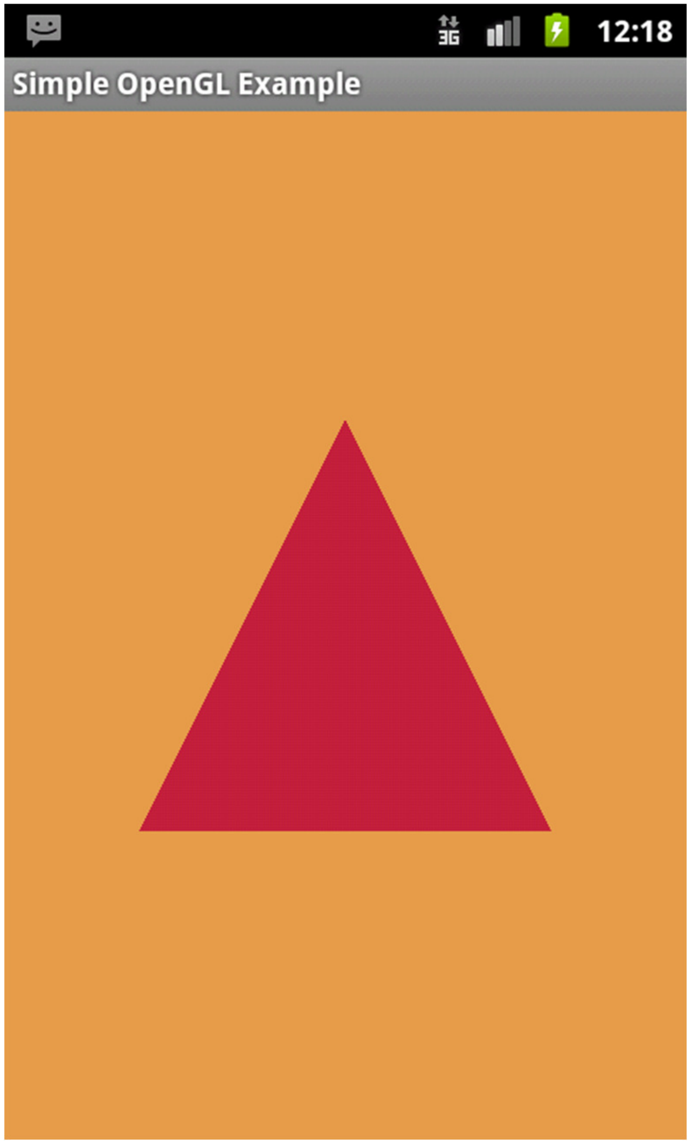
```
public void onSurfaceChanged(GL10 gl, int width, int height) {
    // specifies the affine transformation of
    // x and y from
    // normalized device coordinates to window coordinates
    gl.glViewport(0, 0, width, height);

    // make adjustments for screen ratio
    float ratio = (float) width / height;
    // set matrix to projection mode
    gl.glMatrixMode(GL10.GL_PROJECTION);
    // reset the matrix to its default state
    gl.glLoadIdentity();
    // apply the projection matrix
    // left, right, bottom, top, near, far
    gl.glFrustumf(-ratio, ratio, -1, 1, 3, 7);

    initShapes();
}
```

onDrawFrame

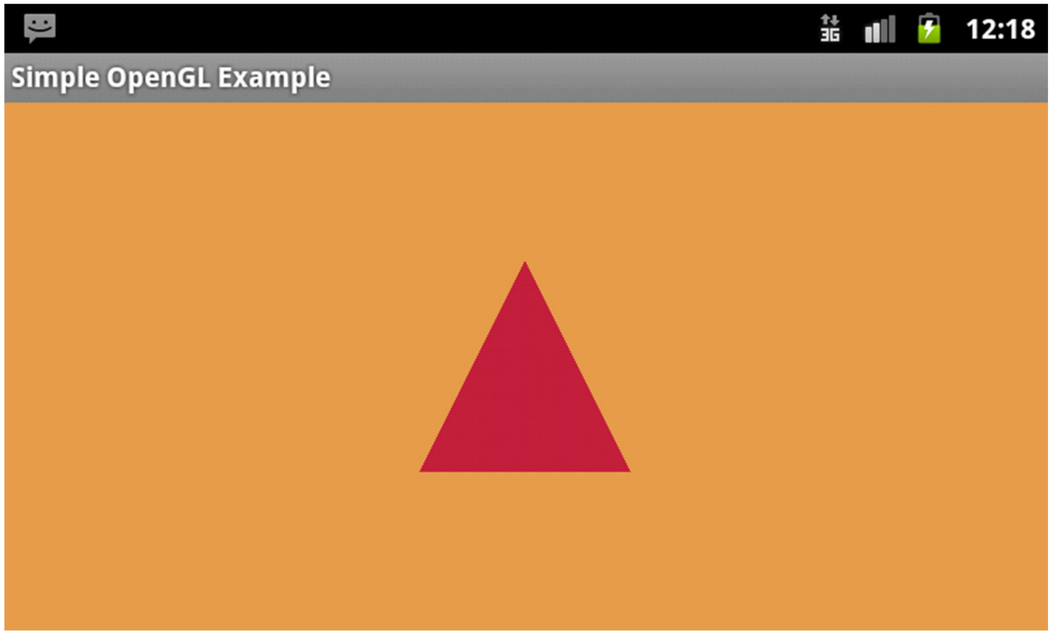


```
public void onDrawFrame(GL10 gl) {
    // Redraw background color
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);


     // Set GL_MODELVIEW transformation mode
    gl.glMatrixMode(GL10.GL_MODELVIEW);
    gl.glLoadIdentity();   // reset the matrix to its default state

    // When using GL_MODELVIEW, you must set the view point
    // GLU = Graphics Library Utilities

    GLU.gluLookAt(gl, 0, 0, -5, // gl10, eyeX, eyeY, eyeZ
            0f, 0f, 0f, // centerX, centerY, centeZ
            0f, 1.0f, 0.0f);  // upX, upY, upZ

    // Draw the triangle
    gl.glColor4f(0.77f, 0.12f, 0.23f, 1);
    // coordinates per vertex, type, stride (offset between vertices)
    gl.glVertexPointer(3, GL10.GL_FLOAT, 0, triangleVB);
    // mode, first, count of vertices
    gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);
}
```

Result of Correcting Projection



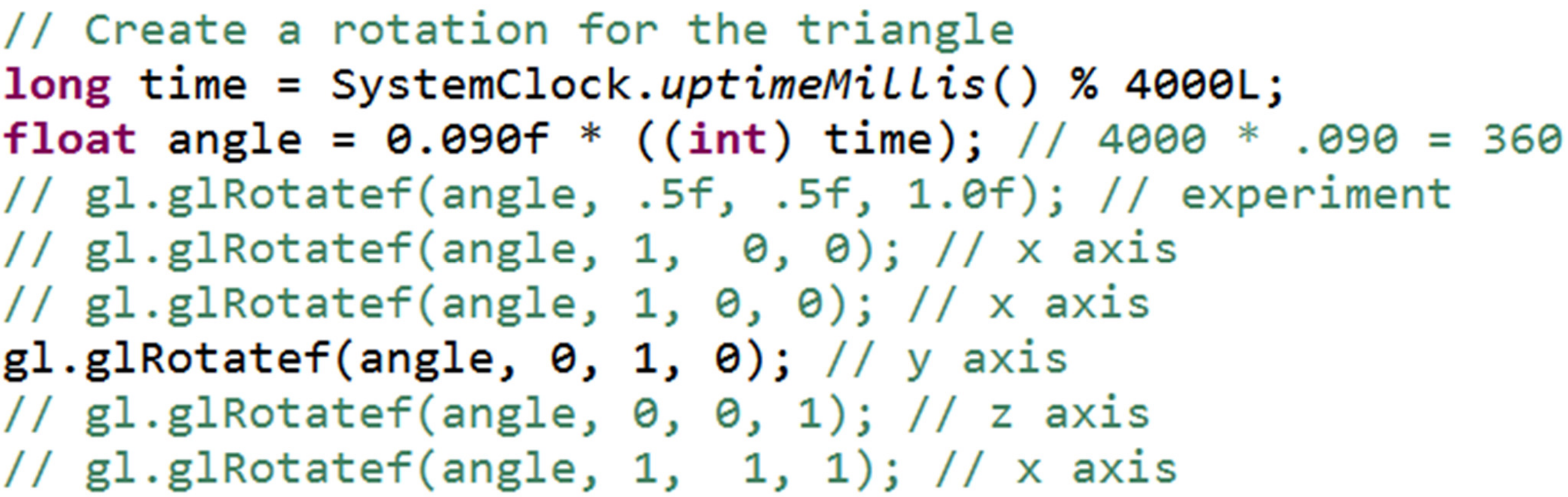
Simple OpenGL Example



Simple OpenGL Example

Adding Motion

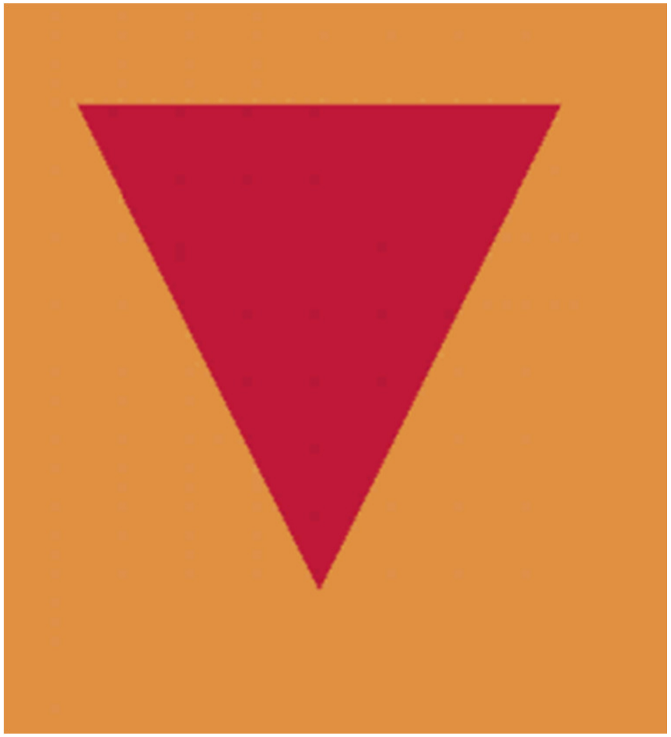
- in onDrawFrame
- define vector of rotation



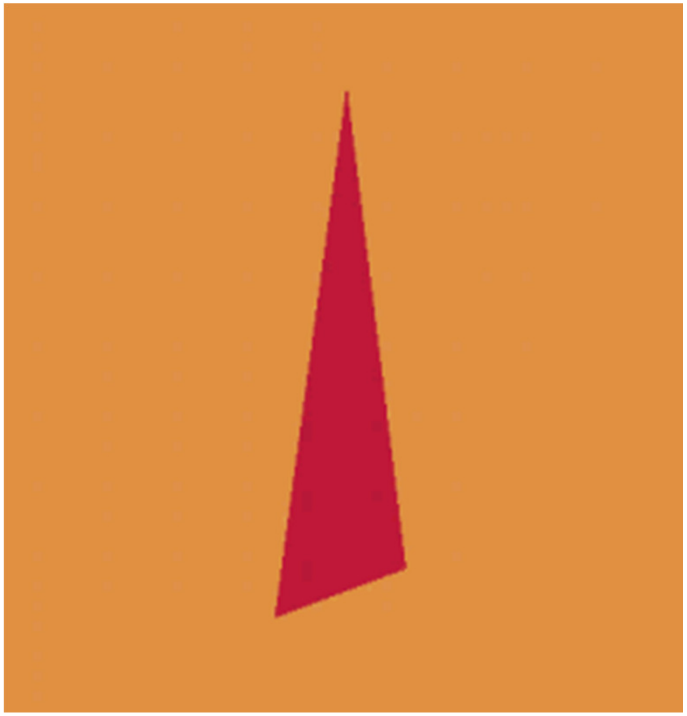
```
// Create a rotation for the triangle
long time = SystemClock.uptimeMillis() % 4000L;
float angle = 0.090f * ((int) time); // 4000 * .090 = 360
// gl.glRotatef(angle, .5f, .5f, 1.0f); // experiment
// gl.glRotatef(angle, 1,  0, 0); // x axis
// gl.glRotatef(angle, 1, 0, 0); // x axis
gl.glRotatef(angle, 0, 1, 0); // y axis
// gl.glRotatef(angle, 0, 0, 1); // z axis
// gl.glRotatef(angle, 1,  1, 1); // x axis
```

Results

X Axis (angle, 1, 0, 0)

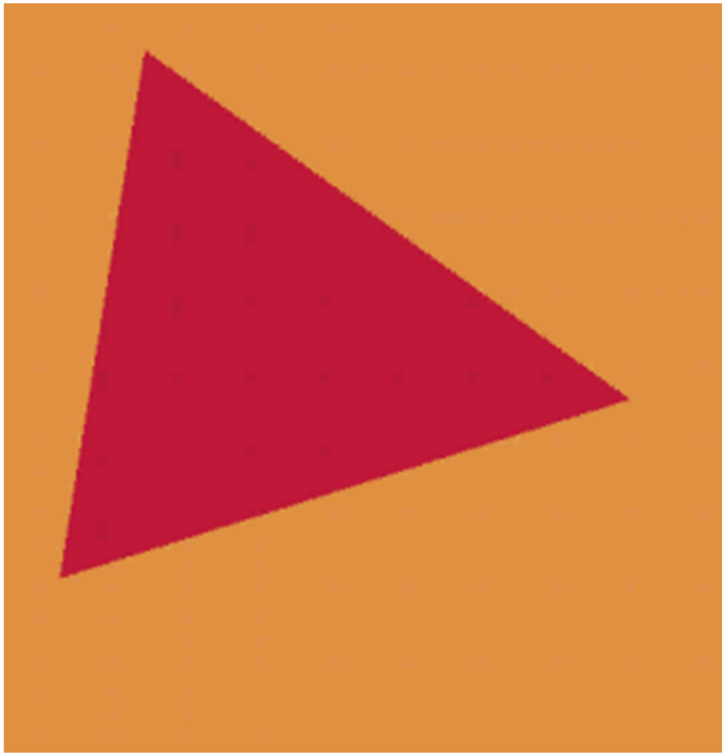


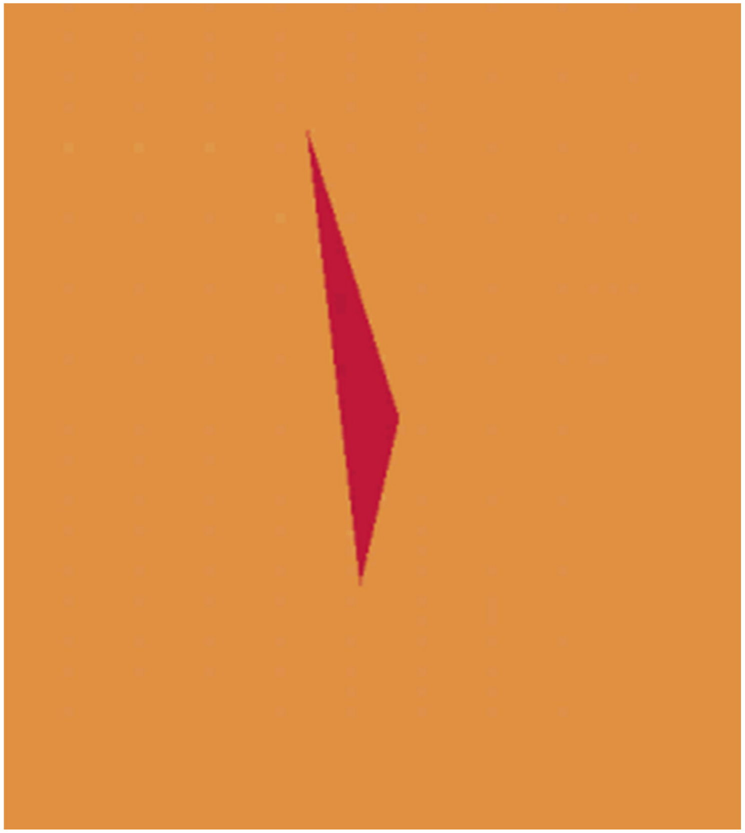
Y Axis (angle, 0, 1, 0)



Results

Z Axis (angle, 0, 0, 1)   Y Axis (angle, -1, 1, -1)



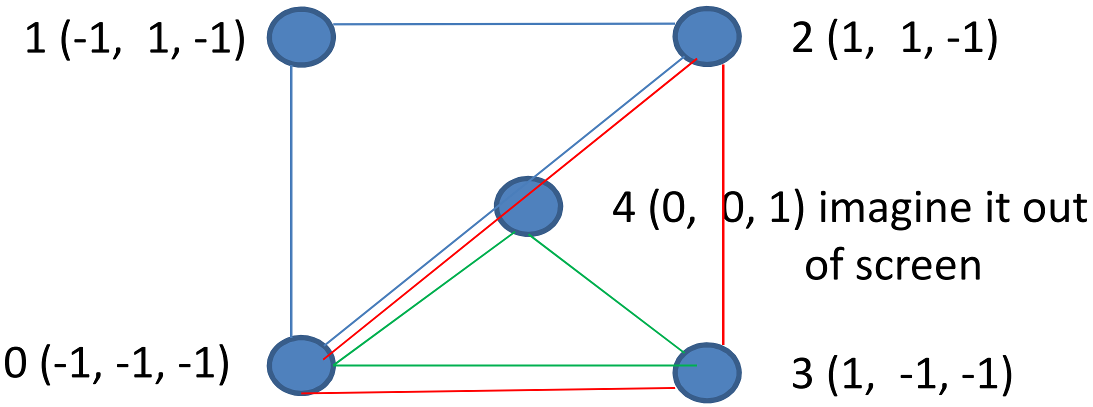


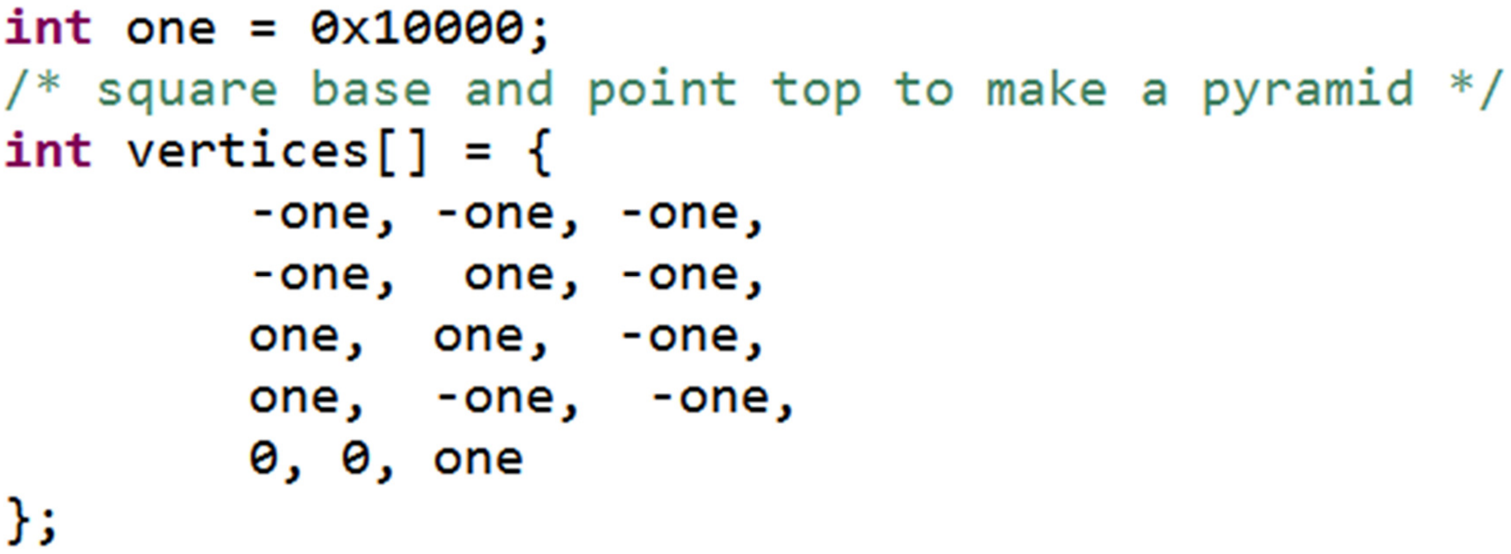
Another Example

- Draw a pyramid that bounces around the screen
- Same basic steps as previous apps
- Activity with GLSurfaceView
- Implementation of GLSurfaceView.Renderer
- Pyramid class that defines the geometry and appearance of 3d pyramid object

Constructing Pyramid

- specify vertices for 6 triangles
- 4 sides, 2 triangles for the base

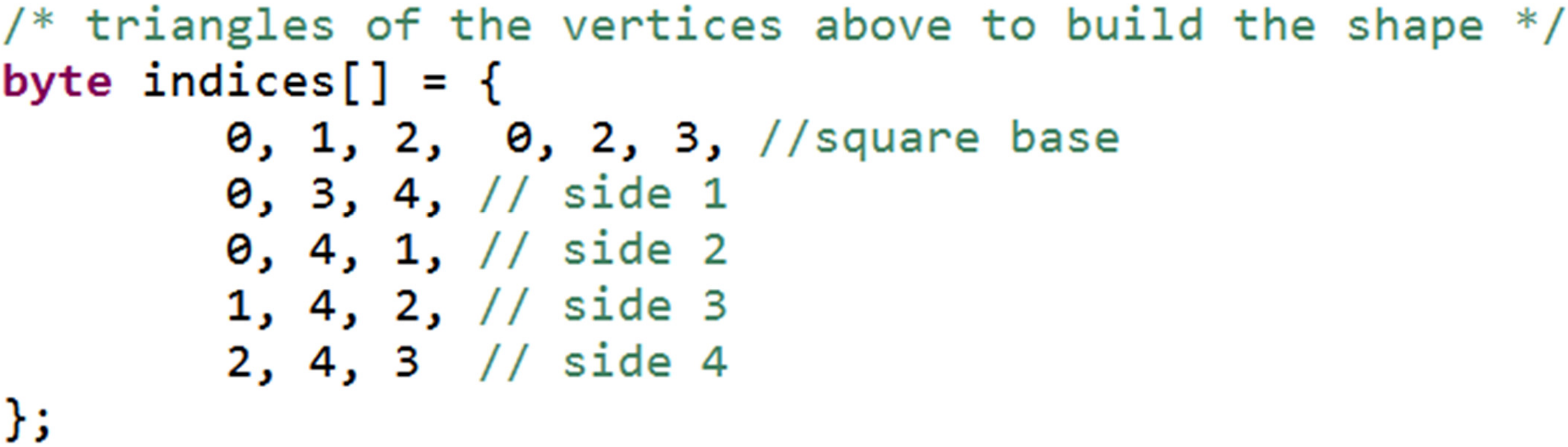




```
int one = 0x10000;
/* square base and point top to make a pyramid */
int vertices[] = {
        -one, -one, -one,
        -one,  one, -one,
        one,   one,  -one,
        one,   -one,  -one,
        0, 0, one
};
```

Constructing Pyramid

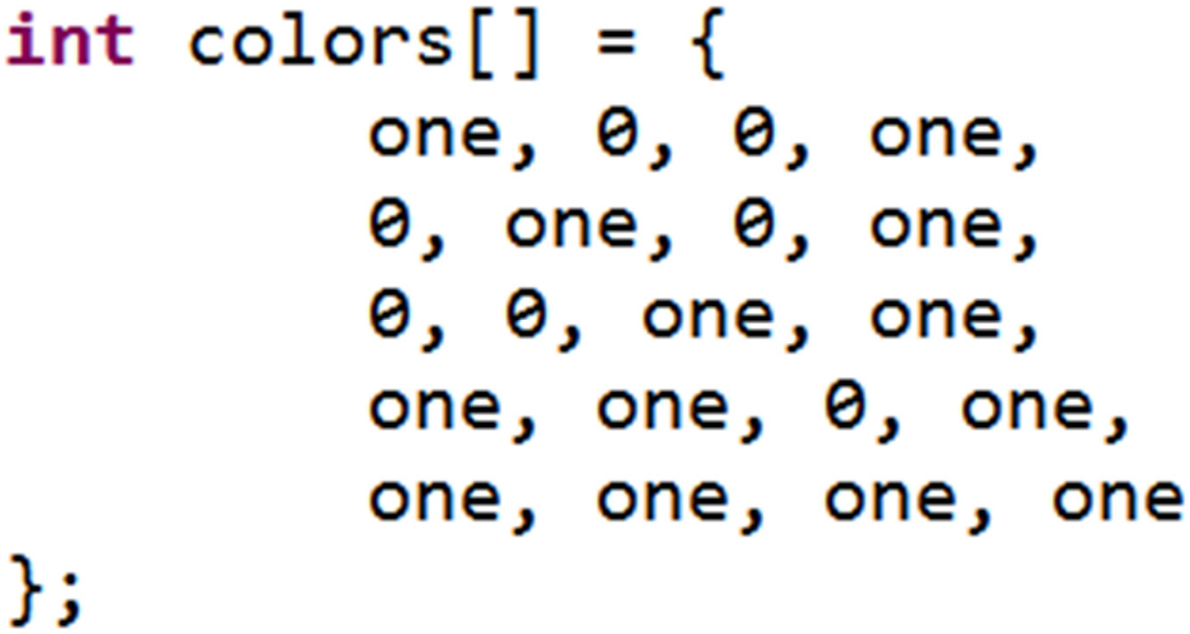
- Indices refers to set or coordinate (x, y, z)



```
/* triangles of the vertices above to build the shape */
byte indices[] = {
        0, 1, 2,  0, 2, 3, //square base
        0, 3, 4, // side 1
        0, 4, 1, // side 2
        1, 4, 2, // side 3
        2, 4, 3  // side 4
};
```

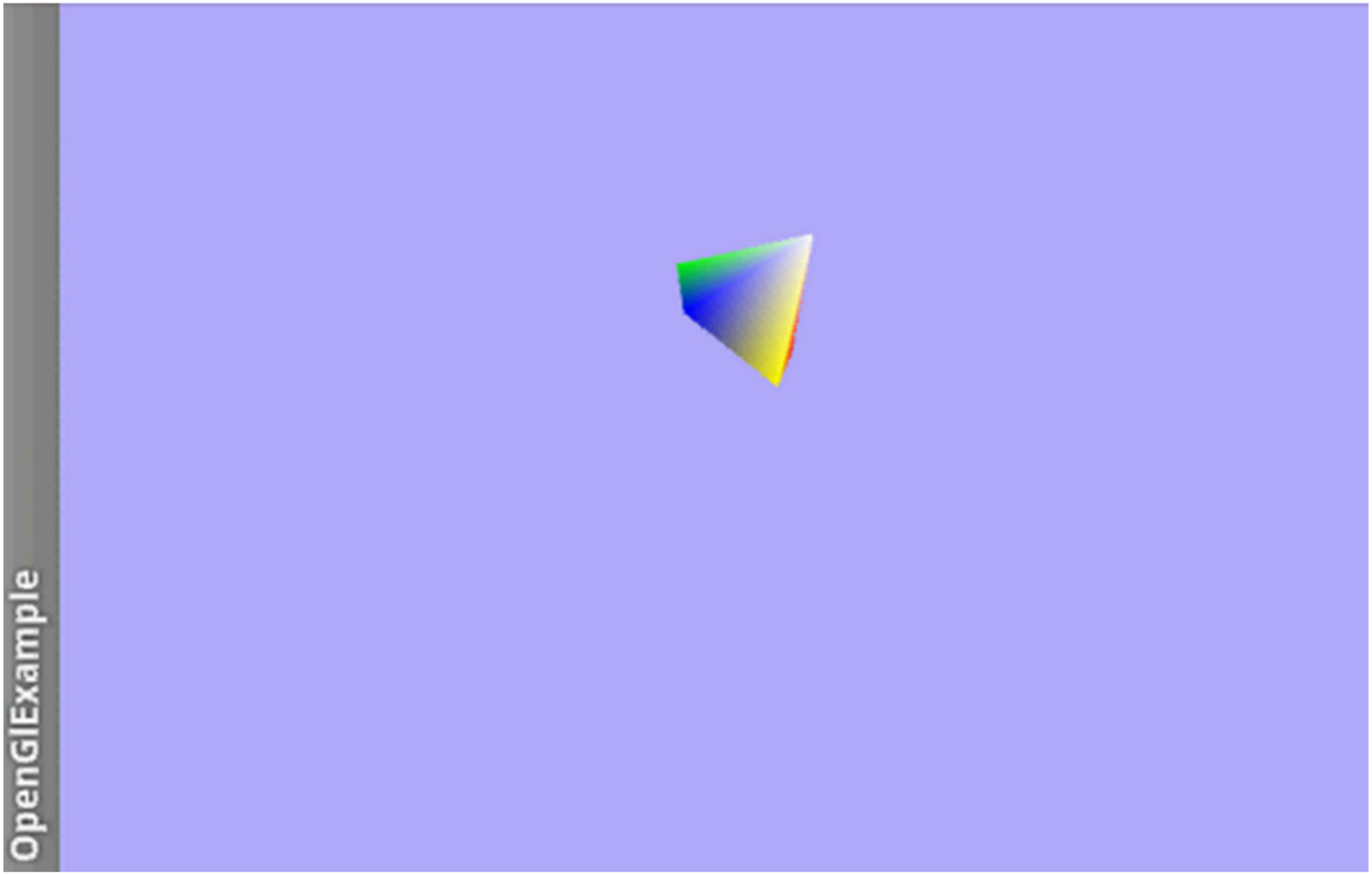
Coloring Pyramid

- Define colors for each of the 5 vertices
- Colors blend from one vertex to another
- recall, rgba



```
int colors[] = {
        one, 0, 0, one,
        0, one, 0, one,
        0, 0, one, one,
        one, one, 0, one,
        one, one, one, one
};
```

Result



OpenGL Options

- Renderscript
  - high performance, but low level
  - scripts written in C
- OpenGLUT, OpenGL Utility Toolkit
  - not officially part of Android, Android GLUT Wrapper
  - include more geometric primitives