Android 3D Graphics

Sang Shin
Michèle Garoche
www.javapassion.com

"Learn with Passion!"

Disclaimer

- Portions of this presentation are modifications based on work created and shared by the Android Open Source Project
  - http://code.google.com/policies.html
- They are used according to terms described in the Creative Commons 2.5 Attribution License
  - http://creativecommons.org/licenses/by/2.5/

Topics

- 3D support in Android
- Android APIs
- OpenGL ES APIs
- Building a polygon

3D Support in Android

3D with OpenGL

- Android includes support for high performance 3D graphics via the OpenGL API — specifically, the OpenGL ES (Embedded System) API.
- OpenGL ES is a flavor of the OpenGL specification intended for embedded devices.
- Versions of OpenGL ES are loosely peered to versions of the primary OpenGL standard
- Beginning with Android 2.2, Android supports OpenGL ES 2.0 (with backward compatibility support for OpenGL ES 1.1)

Android APIs

Classes

- GLSurfaceView
- GLSurfaceView.Renderer

GLSurfaceView Class

- A *GLSurfaceView class* is an extension of *SurfaceView* class and uses the dedicated surface for displaying OpenGL rendering
  - SurfaceView is subclass of View class
- A *GLSurfaceView* provides the following features:
  - Manages a surface, which is a special piece of memory that can be composited into the Android view system.
  - Accepts a user-provided Renderer object that does the actual rendering.
  - Renders on a dedicated thread to decouple rendering performance from the UI thread.
  - Supports both on-demand and continuous rendering.

Example: GLSurfaceView Class

```java
public class MainActivity extends Activity {

    private GLSurfaceView mGLSurfaceView;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Create GLSurfaceView object and set custom renderer
        mGLSurfaceView = new GLSurfaceView(this);
        mGLSurfaceView.setRenderer(new PyramidRenderer(false));
        setContentView(mGLSurfaceView);
    }

    protected void onResume() {
        super.onResume();
        mGLSurfaceView.onResume();
    }

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

}
```

GLSurfaceView.Renderer Interface

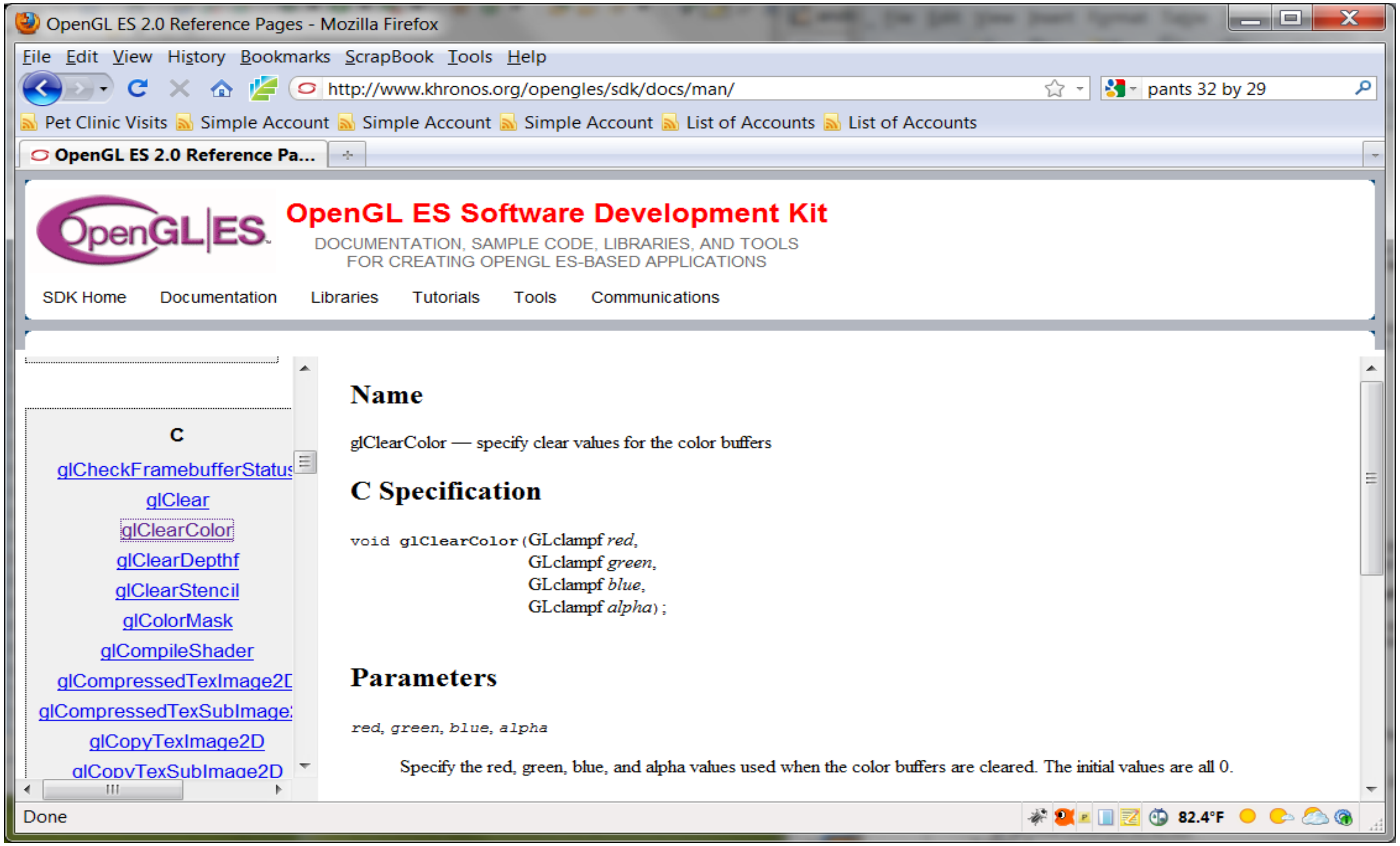
- A generic render interface
- The renderer is responsible for making OpenGL calls to render a frame
- Methods to implement

```
// Called when the surface is created or recreated.
public void onSurfaceCreated(GL10 gl, EGLConfig config)

// Called to draw the current frame.
public void onDrawFrame(GL10 gl)

// Called when the surface changed size.
public void onSurfaceChanged(GL10 gl, int width, int height)
```

OpenGL ES APIs

OpenGL ES APIs

http://www.khronos.org/opengles/sdk/docs/man/



Name

glClearColor — specify clear values for the color buffers

C Specification

```
void glClearColor(GLclampf red,
                  GLclampf green,
                  GLclampf blue,
                  GLclampf alpha);
```

Parameters

red, green, blue, alpha

Specify the red, green, blue, and alpha values used when the color buffers are cleared. The initial values are all 0.

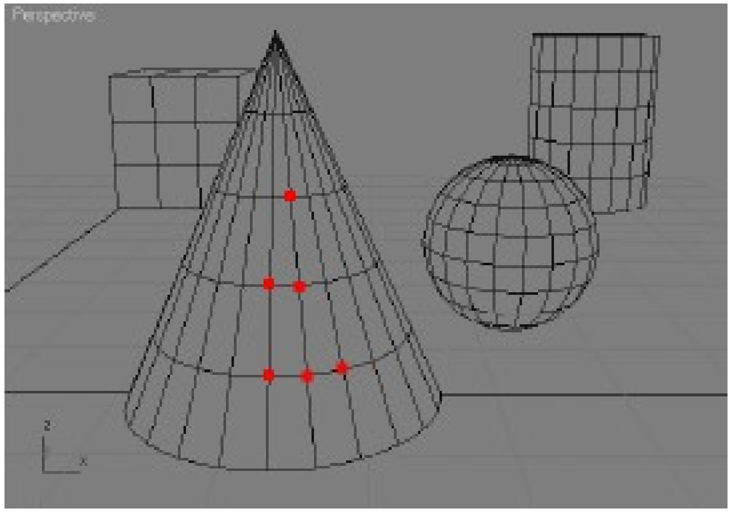
Building a Polygon

Components of 3D models

- 3D models are built up with smaller elements (vertices, edges, faces, and polygons) which can be manipulated individually.

Vertex

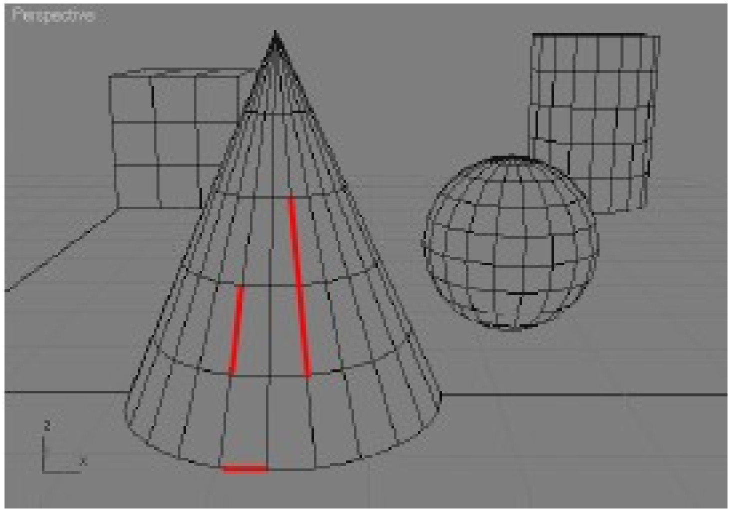
- Vertexes are the points that connect between edges in a 3D model
- To define the vertices on android we define them as a float array that we put into a byte buffer to gain better performance.
- A vertex can also be a representation for the position of a camera or a light source



Thanks to Sherin Farid

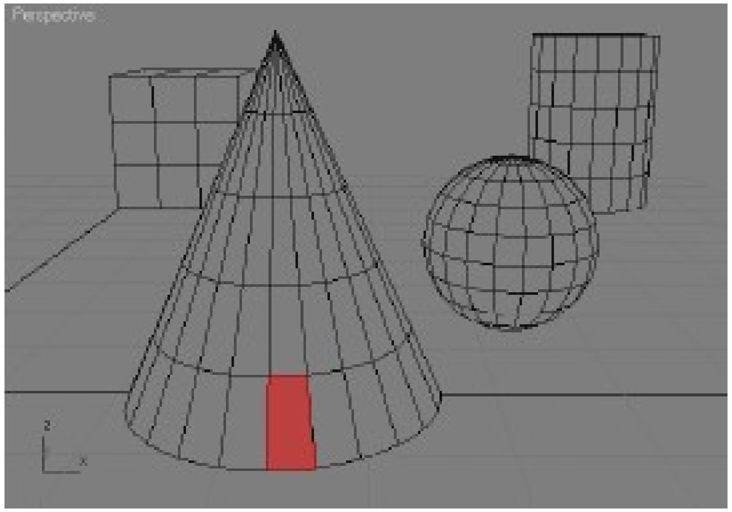
Edges

- Edges are lines/separators between each face and another, the edge of the face



Faces

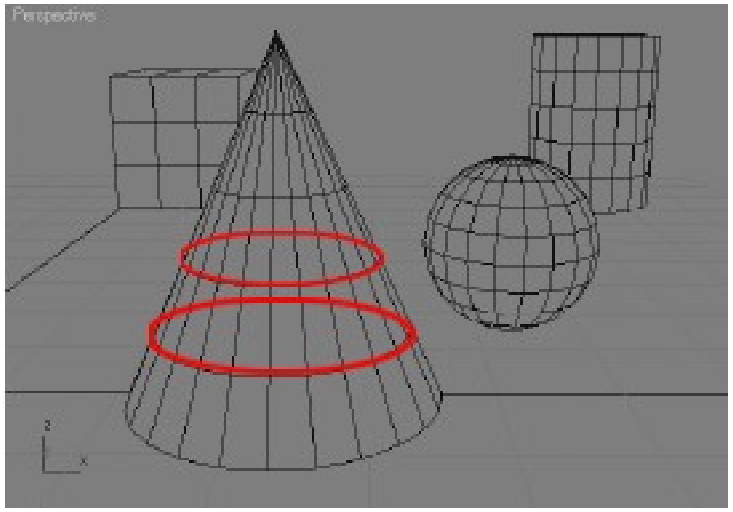
- A face is a 2D shape, many of these are connected together through vertexes and edges to form up a 3D model



Thanks to Sherin Farid

Segments

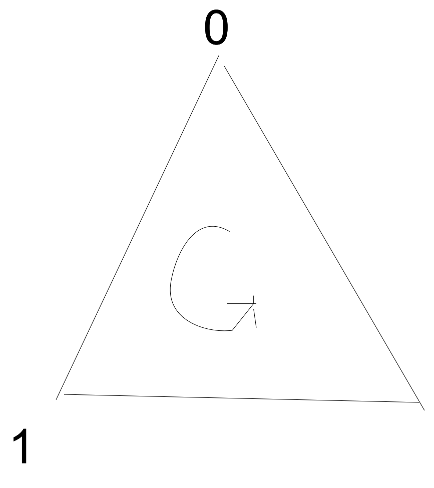
- Segments are like edges only they go all around the model, the more segments a model contains the more faces, edges and vertexes it contains and the smoother the model will appear.

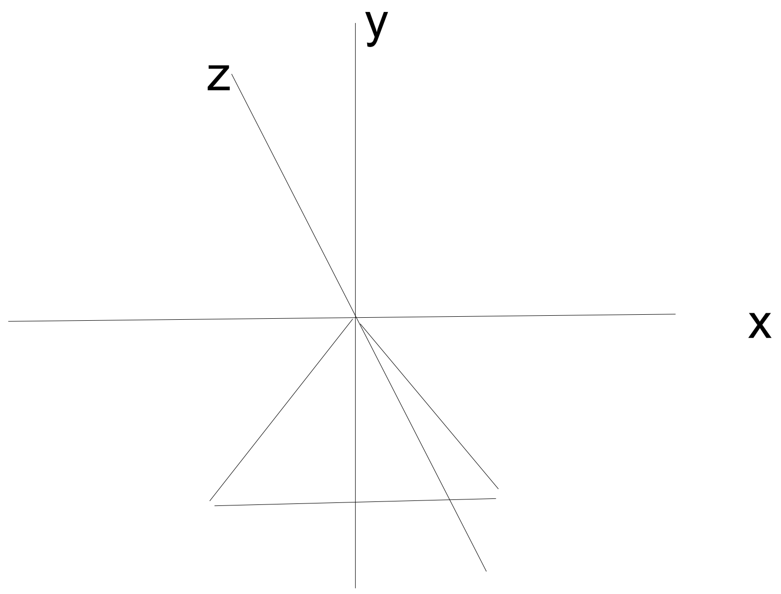


Thanks to Sherin Farid

Example: Vertices of Triangle

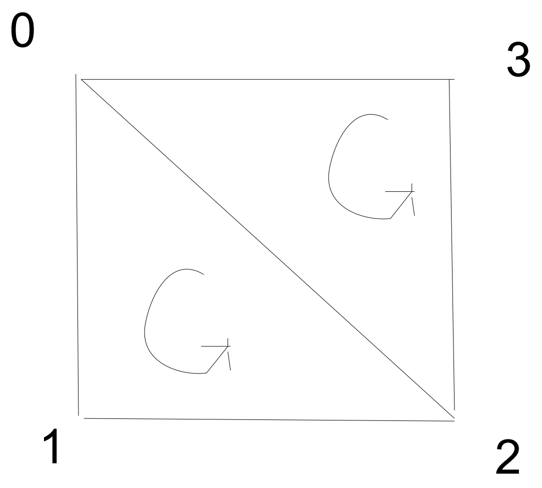
```
class Triangle {

   private float[] vertices = { // Vertices of the triangle
        0.0f, 1.0f, 0.0f,    // 0. top
        -1.0f, -1.0f, 0.0f, // 1. left-bottom
        1.0f, -1.0f, 0.0f    // 2. right-bottom
   };

   // Indices to above vertices (in CCW - Counter Clock Wise)
   private byte[] indices = { 0, 1, 2 };
```





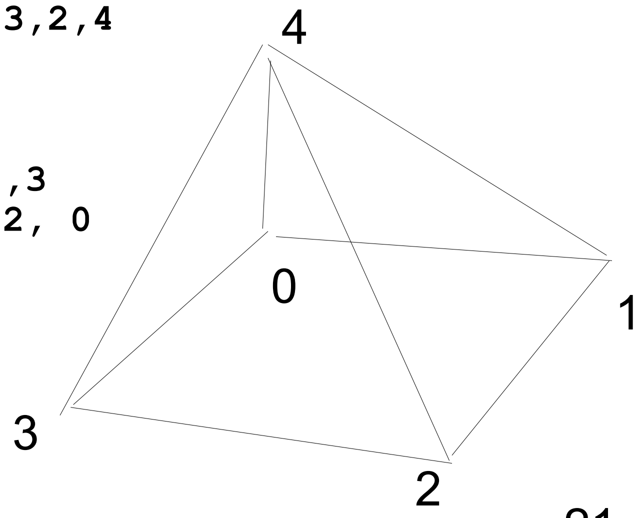
Example: Vertices of Square

```
class Square {

   // Square has 4 vertices
   private float[] vertices = {   // Vertices for the square
            -1.0f,  1.0f, -0.5f,  // 0, Top Left
            -1.0f, -1.0f, 0.5f,   // 1, Bottom Left
             1.0f, -1.0f, -0.5f,  // 2, Bottom Right
             1.0f,  1.0f, 0.5f,   // 3, Top Right
   };

   // Square is made of two triangles.
   private byte[] indices = { 0, 1, 2, // first triangle CCW
                              0, 2, 3  // second triangle CCW
                            };
```



Example: Vertices of Pyramid

```
class Pyramid {

   // Pyramid has 5 vertices
   private float[] vertices = { // 5 vertices of the pyramid in (x,y,z)
       -1.0f, -1.0f, -1.0f,     // 0. left-bottom-back
        1.0f, -1.0f, -1.0f,     // 1. right-bottom-back
        1.0f, -1.0f, 1.0f,      // 2. right-bottom-front
        -1.0f, -1.0f, 1.0f,     // 3. left-bottom-front
        0.0f, 1.0f, 0.0f        // 4. top
   };

   // 5 faces - made up with 6 triangles - bottom face, which is
   // a square is made of two triangles
   private byte[] indices = { // Vertex indices of the 4 Triangles
        2, 4, 3, // front face (CCW) - same as 4,3,2 or 3,2,4
        4, 2, 1, // right face  - same as 2,1,4 or 1,4,2
        0, 4, 1, // back face - same 4,1,0 or 1,0,4
        4, 0, 3, // left face - same 0,3,4 or 3,4,2
        2, 3, 0, // bottom half left - same 3,0,2 or 0,2,3
        2, 0, 1  // bottom half right - same 0,1,2 or 1,2, 0
   };
```



Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"