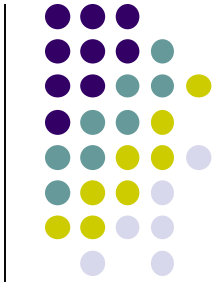
Mobile 3D Graphics

Introduction to Android

OpenGL ES

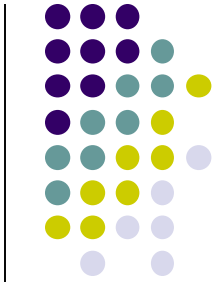
Build an OpenGL ES environment

OpenGL ES



- **Open Graphics Library** OpenGL : is a **cross-platform** graphics API that specifies a standard software interface for **3D graphics** processing hardware.
- **OpenGL ES** is a flavor of the OpenGL specification intended for embedded devices.
- **Android supports several versions of the OpenGL ES API:**
  - **OpenGL ES 1.0 and 1.1** - supported by Android **1.0** and higher.
  - **OpenGL ES 2.0** - supported by Android 2.2 (**API level 8**) and higher.
  - **OpenGL ES 3.0** - supported by Android 4.3 (**API level 18**) and higher.
  - **OpenGL ES 3.1** - supported by Android 5.0 (**API level 21**) and higher

OpenGL ES

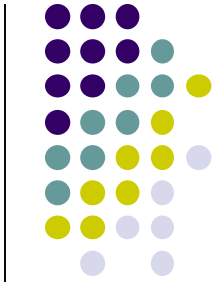


There are **TWO** **foundational classes** in the Android framework that let you create and manipulate graphics with the **OpenGL ES API:**

- **GLSurfaceView** : use this **class** by creating an **instance of GLSurfaceView** and adding your **Renderer** to it.

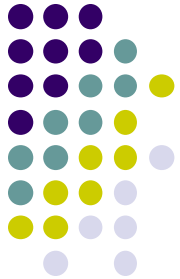
- **GLSurfaceView.Renderer:** This **interface** defines the methods required for **drawing graphics in a GLSurfaceView**. You must provide an implementation of this interface as a **separate class** and attach it to your **GLSurfaceView** instance using **GLSurfaceView.setRenderer()**.

Create an activity for OpenGL ES graphics



```
public class OpenGLES20Activity 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 MyGLSurfaceView(this);
        setContentView(mGLView);
    }
}
```

Declaring OpenGL requirements



- **OpenGL ES version requirements** - If your application requires a **specific version of OpenGL ES**, you must declare that requirement by adding the following settings to your **manifest** as shown below.
- **For OpenGL ES 2.0:**

```
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
```

- **For OpenGL ES 3.0:**

```
<uses-feature android:glEsVersion="0x00030000" android:required="true" />
```

- **For OpenGL ES 3.1:**

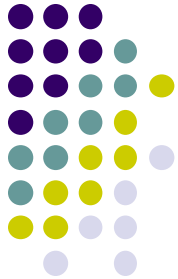
```
<uses-feature android:glEsVersion="0x00030001" android:required="true" /
```

Declaring OpenGL requirements

- If your application uses **texture compression**, you must also declare which compression formats your app supports, so that it is only installed on compatible devices.

```
<supports-gl-texture
android:name="GL_OES_compressed_ETC1_RGB8_texture" />

<supports-gl-texture
android:name="GL_OES_compressed_paletted_texture" />
```

Build a GLSurfaceView object



- A **GLSurfaceView** is a specialized view where you can draw OpenGL ES graphics.
- It does not do much by itself. The **actual drawing** of objects is controlled in the **GLSurfaceView.Renderer** that you set on this view.

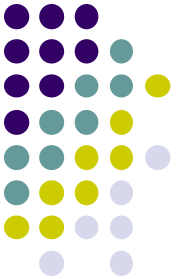
```
class MyGLSurfaceView extends GLSurfaceView {

    private final MyGLRenderer mRenderer;

    public MyGLSurfaceView(Context context){
        super(context);

        // Create an OpenGL ES 2.0 context
        setEGLContextClientVersion(2);

        mRenderer = new MyGLRenderer();

        // Set the Renderer for drawing on the GLSurfaceView
        setRenderer(mRenderer);
    }
}
```

inner class in the activity that uses it

Build a renderer class

- There are **three methods** in a **renderer** that are called by the Android system in order to figure out **what** and **how** to draw on a **GLSurfaceView**:
  1. **onSurfaceCreated()** - Called once to **set up** the view's OpenGL ES environment.
  2. **onDrawFrame()** - Called for each **redraw** of the view.
  3. **onSurfaceChanged()** - Called if the **geometry** of the view changes, **for example** when the device's screen orientation changes.

Build a renderer class



```
public class MyGLRenderer implements GLSurfaceView.Renderer {

    public void onSurfaceCreated(GL10 unused, EGLConfig config) {
        // Set the background frame color
        GLES20.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    }

    public void onDrawFrame(GL10 unused) {
        // Redraw background color
        GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT);
    }

    public void onSurfaceChanged(GL10 unused, int width, int height) {
        GLES20.glViewport(0, 0, width, height);
    }
}
```

separate class

Build an OpenGL ES environment

- **That's all there is to it!**
- The code examples create a simple Android application that displays **a black screen using OpenGL**. While this code does not do anything very interesting, by creating these classes, you have **laid** the foundation you need to start drawing graphic elements with OpenGL.

References

- **Build an OpenGL ES environment**

https://developer.android.com/training/graphics/opengl/environment