# COMP30510 Mobile Application Development

# Graphics on Android

Dr. Abraham Campbell
University College Dublin
Abey.campbell@ucd.ie

# Outline

- Graphics Options beyond using View/ Widget Objects
  - Drawable
  - Canvas
    - SurfaceView
- 3D Graphics option
  - GLSurface View
    - OpenGL ES
- Video on Android

# Remember everything is just a bitmap

- In Android , your screen is just a large buffer that the device draws onto
- Every view is just an underlying bitmap, and its canvas is attached to it.
- This change of the graphic buffer needs to happen sufficiently for the user to believe the screen is animating.

# Drawable approach

- Android custom 2D graphics library
- Primary library : Android.graphics.drawable
- A drawable is an abstraction of anything that can be drawn
- As these are so important and can be unique to each phone/region, they use the  res/drawable/ folder, using typical Android annotations
- There are three approach's the use an image or XML description or class constructors.
- For this course I will recommend just using the standard image, if you want to use animations, I will recommend the Canvas option for now.

# Creating from resource images

- Adding image files (normally PNG files are recommend)
- Remember PNG are Lossless compressed.
- JPG do work but they lossy compressed

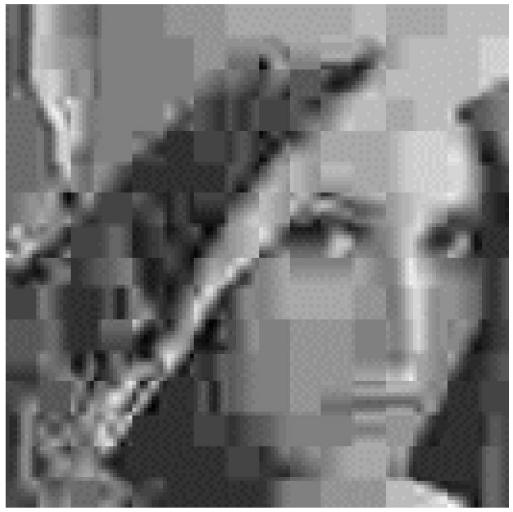
- Lossless > lossy , as Lossless do not change the image but are reduced size due to the compression  from a bitmap
- Lossy compression works by removing and changing information from an original bitmap, this compression is very smart as the human eye can be tricked easily.

# Classic Lena example

*Example of Lossy Compression*

Original Lena Image (12KB size)

Lena Image, Compressed (85% less information, 1.8KB)



Lena Image, Highly Compressed (96% less information, 0.56KB)

# Image Formats Supported

| | | | | | |
|---|---|---|---|---|---|
| Image | JPEG | • | • | Base+progressive | JPEG (.jpg) |
| | GIF | | • | | GIF (.gif) |
| | PNG | • | • | | PNG (.png) |
| | BMP | | • | | BMP (.bmp) |
| | WEBP | • (Android 4.0+) | • (Android 4.0+) | | WebP (.webp) |

# Code example from Android Deveopler Doc's

```
LinearLayout mLinearLayout;

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

  // Create a LinearLayout in which to add the ImageView
  mLinearLayout = new LinearLayout(this);

  // Instantiate an ImageView and define its properties
  ImageView i = new ImageView(this);
  i.setImageResource(R.drawable.my_image);
  i.setAdjustViewBounds(true); // set the ImageView bounds to match the Drawable's dimensions
  i.setLayoutParams(new Gallery.LayoutParams(LayoutParams.WRAP_CONTENT,
      LayoutParams.WRAP_CONTENT));

  // Add the ImageView to the layout and set the layout as the content view
  mLinearLayout.addView(i);
  setContentView(mLinearLayout);
}
```

# Canvas approach

- You can direct add a canvas to your activity by extending a view
- Example in android-sdk\samples\android-15\Snake
- A better way would be to extend a Surfaceview  object.
- Technically it is just extending a View but the SurfaceView is dedicated to drawing on a Canvas.
- Remember each view technical has its own canvas which is just a bitmap.
- Best example I cold find was actually not in Android SDK,
- https://github.com/MrBlackk/KillThemAll-Training
  - I have re-written this and adding in touch examples , this version will be put up on the moodle later so you do not have to take pictures ☺
- Remember GITHUB is a great source for tutorial, but learn from it , do not copy .
- As this is such a simple example though you could build on top of it and I would not see that as grounds for plagiarism.  Normally I would only say that for direct samples from the Android SDK samples.

# Canvas with Surface view

- First we need to setup our activity as usually but we add our custom Surface view called GameView

```
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(new GameView(this));
    }
}
```

# Thread to keep updating the SurfaceView

```
public class GameLoop extends Thread {
private GameView view;
private boolean running = false;

public GameLoop(GameView view) {
this.view = view;
} public void setRunning(boolean run) {
running = run;
}
@Override
public void run() {
while (running) {
Canvas c = null;
try {
c = view.getHolder().lockCanvas();
synchronized (view.getHolder()) {

view.draw(c);
}
} finally {
if (c != null) {
view.getHolder().unlockCanvasAndPost(c);
}}}}}
```

# GameView our extended SurfaceView

```
…
public class GameView extends SurfaceView {
…
public GameView(Context context) {
super(context);
gameLoop = new GameLoop(this);
holder = getHolder();
holder.addCallback(new SurfaceHolder.Callback() {
```

# GameView our extended SurfaceView

```
@Override
public void surfaceDestroyed(SurfaceHolder holder) {
        boolean retry = true;
        gameLoop.setRunning(false);
                while (retry) {
                        try {
                                gameLoop.join();
                                retry = false;
                        } catch (InterruptedException e) {
                                }}}
@Override
public void surfaceCreated(SurfaceHolder holder) {
        gameLoop.setRunning(true);
        gameLoop.start();
}
```

# GameView our extended SurfaceView

```
@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
}
});
// need to load in our textures once and not on every redraw
Character = BitmapFactory.decodeResource(getResources(), R.drawable.mario);
Background = BitmapFactory.decodeResource(getResources(), R.drawable.background);
}
```

# GameView our extended SurfaceView

```
 @Override
public void draw(Canvas canvas) {
        if (canvas != null)
                {
                ……….
                canvas.drawBitmap(Background, 0, 0, null);
                canvas.drawBitmap(Character, x, y +CurrentJumpHeight, null);
                }
}
```

# 3D Graphics option /OpenGL ES

- Several version of OpenGL ES
- Version 1 is most similar to OpenGl 2
- Version 2 – 3 Is more like Current OpenGL 4.2
- Powerful but does require more battery life
- New Android phones have dedicated GPU chips

# OpenGL ES activity

```
public class MainActivity extends Activity {

    private BasicGLSurfaceView mView;

    @Override
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        mView = new
BasicGLSurfaceView(getApplication());
     setContentView(mView);
    }

    @Override
    protected void onPause() {
        super.onPause();
        mView.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
        mView.onResume();
    }
}
```

# OpenGL ES

- To use you will need to make a specialised SurfaceView

```
import android.opengl.GLSurfaceView;

class BasicGLSurfaceView extends GLSurfaceView {
        public BasicGLSurfaceView(Context context) {
        super(context);
        setEGLContextClientVersion(2);
        setRenderer(new MYTriangleRenderer(context));
    }
}
```

# MyTriangleRenderer

- As OpenGL ES 2 , requires the programmer to setup the environment with far greater detail than lower version OpenGL
- To see full code
  - android-sdk\samples\android-15\BasicGLSurfaceView

- I’ll go through a demo using eclipse in class.

- For this module, I recommend using the Canvas Object.

# Video in Android

- To start playing a Sound or video file in android
- Use the MediaPlayer class.

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

- Remember if you want to add video directly in your app
- Use the VideoView and set its path

```
VideoView videoView = (VideoView)findViewById(R.id.VideoView);
VideoView.setVideoPath("/sdcard/video.mp4");
videoView.start();
```

# Video formats supported

| | | | | | |
|---|---|---|---|---|---|
| Video | H.263 | • | • | | • 3GPP (.3gp)<br>• MPEG-4 (.mp4) |
| | H.264 AVC | •<br>(Android 3.0+) | • | Baseline Profile (BP) | • 3GPP (.3gp)<br>• MPEG-4 (.mp4)<br>• MPEG-TS (.ts, AAC audio only, not seekable, Android 3.0+) |
| | MPEG-4 SP | | • | | 3GPP (.3gp) |
| | VP8 | | •<br>(Android 2.3.3+) | Streamable only in Android 4.0 and above | • WebM (.webm)<br>• Matroska (.mkv, Android 4.0+) |