Android Graphics

- Custom 2D graphics library
- OpenGL ES 1.0 for high performance 3D graphcs

- The design of an application and the APIs used depend on the graphical demands:
  - static graphical application
  - dynamic interactive 2D and 3D rendering for games

AVD1.6

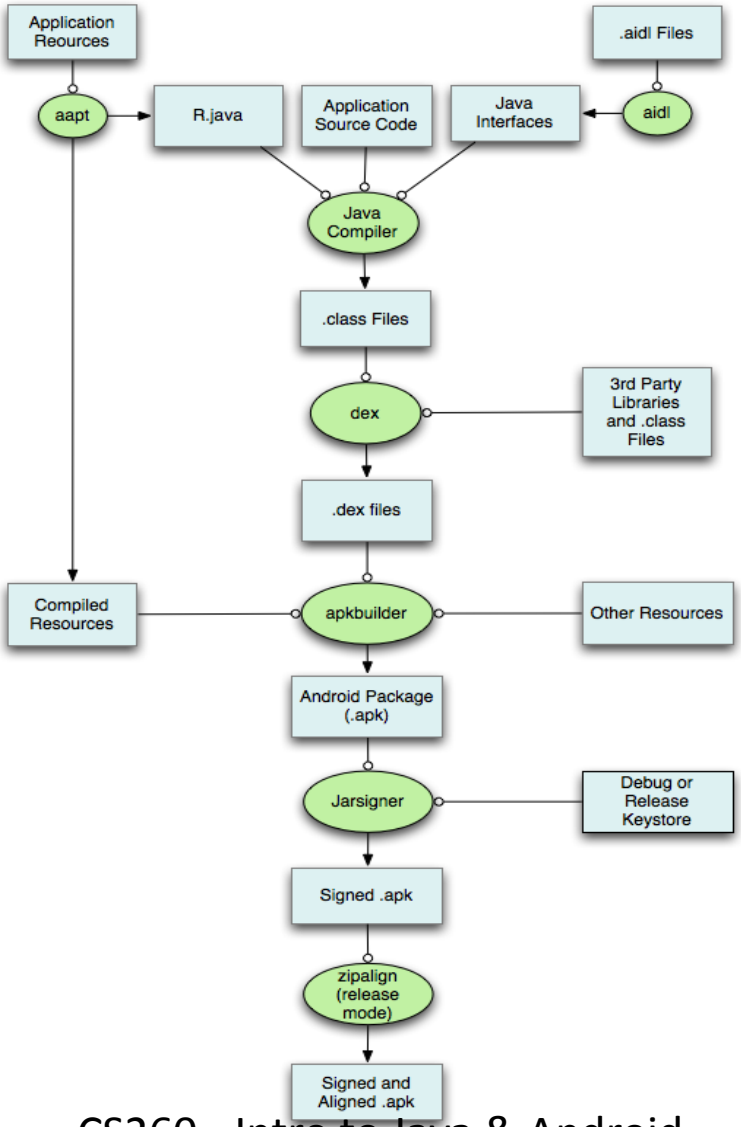
- Create an emulator using 1.6
- Find the config.ini file for the AVD and change hw.dPad=no to hw.dPad=yes
- My config.ini is in c:\Users\ryandj\android\avd\AVD1.6.avd

- We are dropping down to a simple, small device with a DPAD to do some basic graphics
- We will get into more complicated graphics and Threading by Friday!!!

Adding Graphics

- Referencing an image (PNG (preferred), JPG (acceptable), GIF (discouraged)) is the easiest way to add graphics
- IMPORTANT
  - Images placed in res/drawable may be optimized with lossless compression by the aapt (Android Asset Packaging Tool)
  - Images placed in the res/raw folder are not optimized

Notice aapt


Application Reources
.aidl Files
aapt
R.java
Application Source Code
Java Interfaces
aidl
Java Compiler
.class Files
dex
3rd Party Libraries and .class Files
.dex files
Compiled Resources
apkbuilder
Other Resources
Android Package (.apk)
Jarsigner
Debug or Release Keystore
Signed .apk
zipalign (release mode)
Signed and Aligned .apk

2D Graphics

- Drawing 2D graphics is done in one of two ways:
  1. Draw the graphics/animations into a View and let Android's View hierarchy take care of the drawing process
  2. Draw the graphics/animation directly to the Canvas by calling the appropriate class's draw() method passing a Canvas

Draw Graphics into a View

```
public class MainActivity extends Activity
{
  private Display mDisplay;
  private GraphicsView mGraphicsView;
  @Override
  protected void onCreate (Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);

    requestWindowFeature (Window.FEATURE_NO_TITLE);
    this.getWindow ().setFlags
         (WindowManager.LayoutParams.FLAG_FULLSCREEN,
          WindowManager.LayoutParams.FLAG_FULLSCREEN);

    WindowManager window = getWindowManager ();
    mDisplay = window.getDefaultDisplay ();

    mGraphicsView = new GraphicsView (this, mDisplay);
    mGraphicsView.setBackgroundColor (Color.BLACK);
    setContentView (mGraphicsView);
    //setContentView (new GraphicsSurfaceView (this, mDisplay));
  }
```

2D Graphics

•Draw the graphics/animation directly to the Canvas by calling the appropriate class's draw() method passing a Canvas

• We will discuss this Thursday or Friday

• Check out ImageView from the class repository and disconnect.

Animation Problem

1. Create MovingSprite inherited from FixedSprite that allows a Sprite to move
2. Get a ball to bounce off of the sides of the window.
3. Using the ArrayList or Vector class, get three different colored balls bouncing off the side of the window.
4. Add the paddle as a fourth object to the bottom of the window.
5. Move the paddle left and right based on the D-Pad arrow. Left is left and right is right.

Animation Problem

If time permits:

1. Capture an onTouch event
2. When the user touches the screen, add another ball on the screen at the position touched
3. Use polymorphism to move blue, green, and yellow balls in different ways