Week 5: Support library, Themes & Styles, ActionBar, Graphics

UW PCE Android Application Development Program Course 1 – Android Development Fundamentals

Agenda

- Week 4 review
  - ListView
  - ExpandableListView
  - GridView
  - ActionBar
  - Action Items
- Support Library
  - Themes and styles
- Graphics Intro
  - Bitmaps
  - Drawables
  - Animation
- Android 5.0 (Lollipop) Intro – if time permits
  - Material design

Android Support Library

- Android is a “living” OS platform. With each new release, comes new APIs that are only available from that release onwards.
- What about “old” devices, or devices running older Android?
- Android Support Library
    - “Bridge” layer that backports certain new APIs to older Android.
    - Not all new APIs are available, but the most important features are usually available
    - Supported from android 1.6 (API level 4) and up.
    - Each support library is backward compatible to a specific Android version.
- You must first determine what features they want, before incorporating one or more support libraries.

Android Support Libraries - features

- **V4 Support Library**
  - for >= Android 1.6 (API 4)
  - App Components, UI, Accesibility & Content
  - compile ‘com.android.support:support-v4:21.0.0’

- **Multidex Support Library**
  - Enables work around Dalvik’s 65K method limitation
  - No longer needed >= 5.0/lollipop because of new ART
  - compile ‘com.android.support:multidex:1.0.+’

- **V7 Support Libraries:**
  - >= Android 2.1/API level 7
  - V7 appcompat, v7 cardview, v7 gridlayout, v7 mediarouter, v7 palette & v7 recyclerview libraries

Android Support Libraries - features

- **v8 Support Library**
  - >= Android API level 8
  - Support for RenderScript computation framework

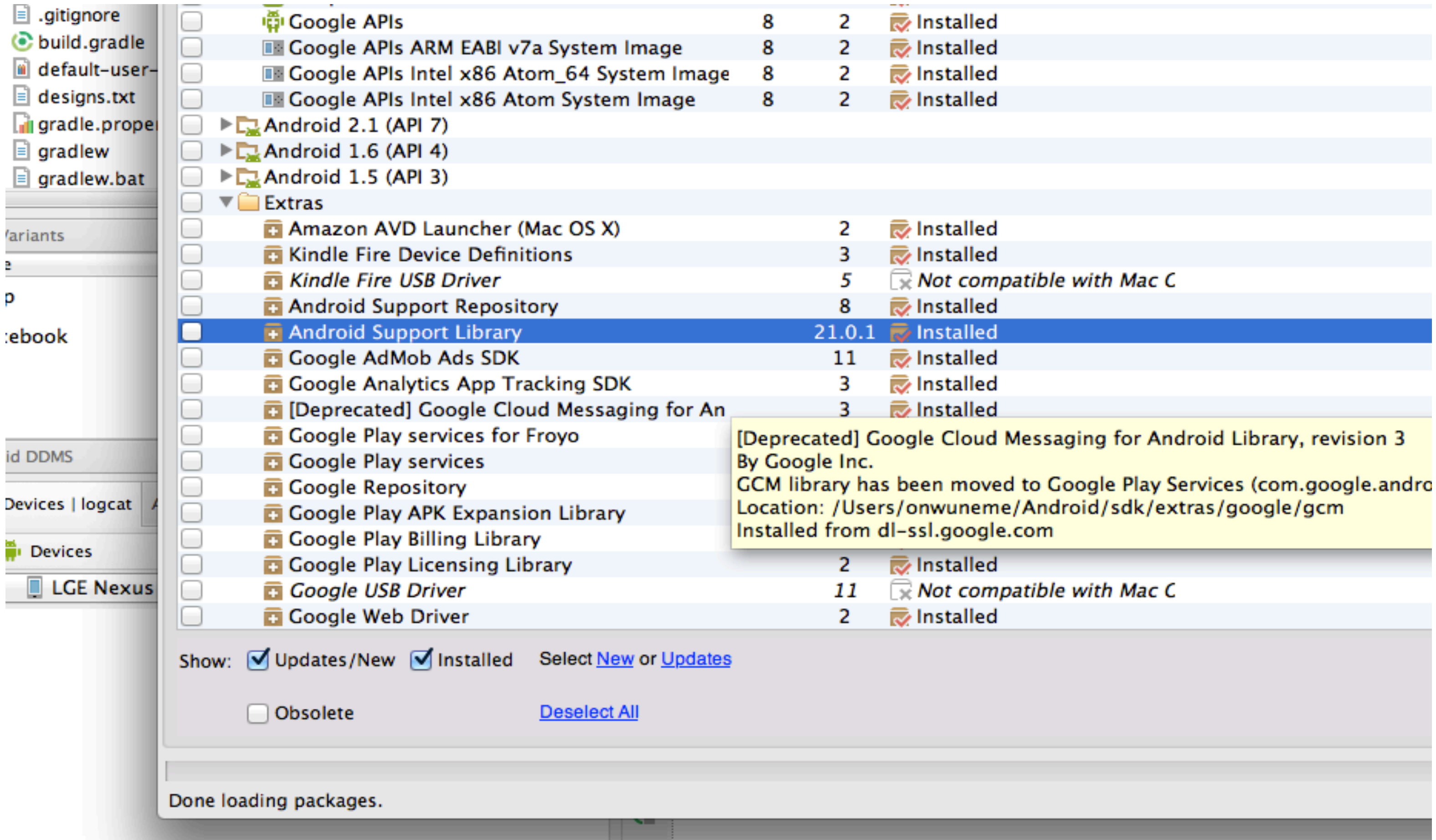
- **v13 Support Library**
  - >= Android 3.2 (API level 13)
  - Additional Fragment support classes
  - compile ‘com.android.suport:support-v13.18.0.+’

- **V17 Leanback Library**
  - APIs to support UIs for TV devices
  - compile ‘com.android.suport:leanback-v17.21.0.+’

- http://developer.android.com/tools/support-library/features.html

Android Support Libraries – SDK installation

- From the android SDK Manager



.gitignore
build.gradle
default-user-
designs.txt
gradle.prope
gradlew
gradlew.bat

'ariants
ebook
id DDMS
Devices | logcat
Devices
LGE Nexus

Google APIs	8	2	Installed
Google APIs ARM EABI v7a System Image	8	2	Installed
Google APIs Intel x86 Atom_64 System Image	8	2	Installed
Google APIs Intel x86 Atom System Image	8	2	Installed
▶ Android 2.1 (API 7)			
▶ Android 1.6 (API 4)			
▶ Android 1.5 (API 3)			
▼ Extras			
Amazon AVD Launcher (Mac OS X)		2	Installed
Kindle Fire Device Definitions		3	Installed
Kindle Fire USB Driver		*5*	*Not compatible with Mac C*
Android Support Repository		8	Installed
Android Support Library		21.0.1	Installed
Google AdMob Ads SDK		11	Installed
Google Analytics App Tracking SDK		3	Installed
[Deprecated] Google Cloud Messaging for An		3	Installed
Google Play services for Froyo			
Google Play services			
Google Repository			
Google Play APK Expansion Library			
Google Play Billing Library			
Google Play Licensing Library		2	Installed
Google USB Driver		*11*	*Not compatible with Mac C*
Google Web Driver		2	Installed

[Deprecated] Google Cloud Messaging for Android Library, revision 3
By Google Inc.
GCM library has been moved to Google Play Services (com.google.andro
Location: /Users/onwuneme/Android/sdk/extras/google/gcm
Installed from dl-ssl.google.com

Show: ✓ Updates/New ✓ Installed Select New or Updates
Obsolete Deselect All

Done loading packages.

Introduction to Styles and Themes

- A style defines the look and feel for a View or window
  - Can specify properties such as height, padding, font color, etc
  - Defined in an XML resource, i.e. styles.xml

- Example

```
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:textColor="#00FF00"
   android:typeface="monospace"
   android:text="@string/hello" />
```

- The above TextView can be written using styles, as

```
<TextView
   style="@style/CodeFont"
   android:text="@string/hello" />
```

Introduction to Styles and Themes

- Define codeFont in an XML file under res/values/

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textColor">#00FF00</item>
    <item name="android:typeface">monospace</item>
  </style>
</resources>
```

- Styles can be inherited from other Styles. You can inherit from styles you create yourself, or platform styles

- Platform Styles can be found here –
  - http://developer.android.com/guide/topics/ui/themes.html#PlatformStyles

Introduction to Styles and Themes

- A Theme is a style applies to an entire Activity or application, rather than an individual View.

- When a style is applied as theme, every View in the Activity or application will apply each property that is supports.

- To apply a theme to an entire Activity or application, add the android:theme attribute to the <activity> or <application> element.

- Newer Android versions have additional themes for use, e.g.

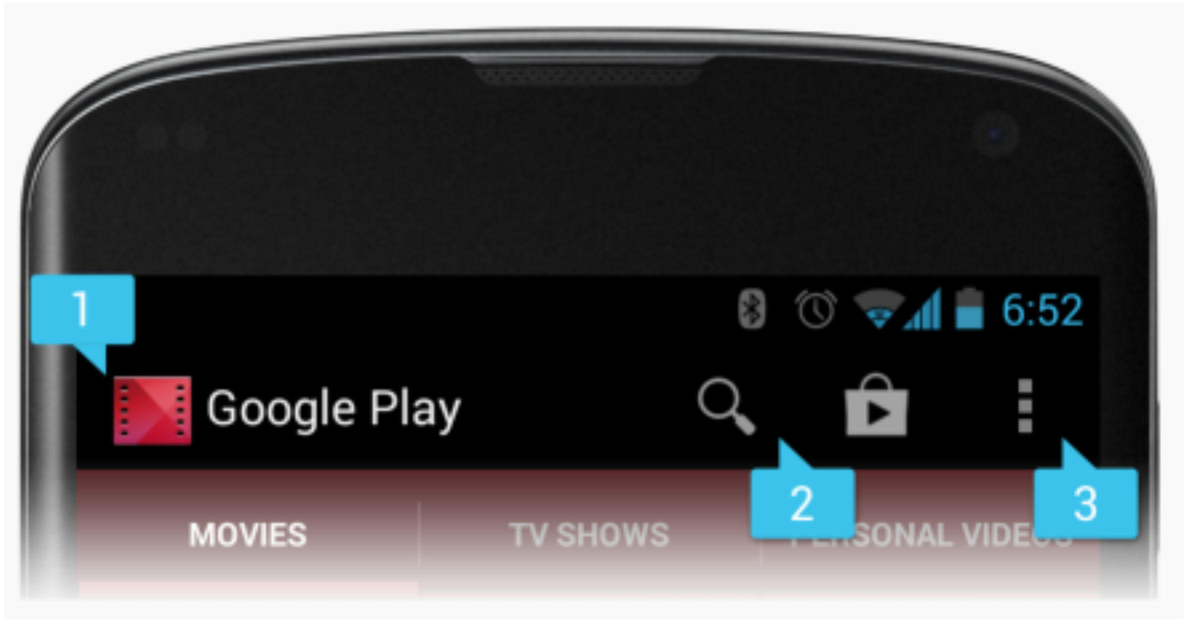
```
<style name="LightThemeSelector" parent="android:Theme.Holo.Light">
   ...
</style>
```

ActionBar

- Introduced in Android 3.0 (API 11)
- Available via Support Library on >= Android 2.1 (API 7)
- Provides familiar and seamless interface across applications
- Provides three key functions:
  - App identify and location within app
  - Important actions (i.e. search) prominent in a predictable way
  - Consistent navigation and view switching

- Use the correct ActionBar APIs in your apps
  - If supporting APIs lower than 11, use
    - import android.support.v7.app.ActionBar
  - If supporting only APIs greater than 11, use
    - import android.app.ActionBar

ActionBar



1 – App icon

2 - Action items

3 – Action overflow

Working with ActionBar

- To include ActionBar in your app
  - Create your activity by extending ActionBarActivity
  - Use (or extend) one of the Theme.AppCompat themes e.g.
    - <activity android:theme="@style/Theme.AppCompat.Light" ... >

- Get reference to ActionBar object
  - ActionBar myActionBar = getSupportActionBar();
  - myActionBar.hide()   -> to hide
  - myActionBar.show() -> to show
- If working with API level 11 and above, get reference using
  - ActionBar myActionBar = getActionBar();
- ActionBar uses app icon by default. To change, add icon attribute in AndroidManifest.xml

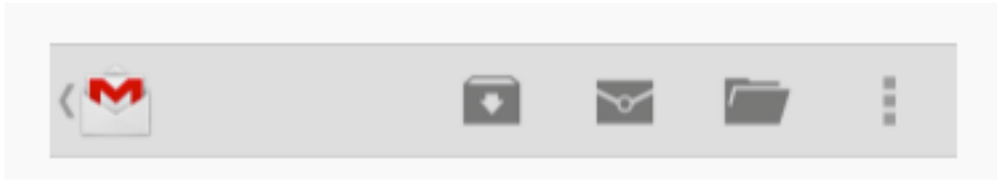
Working with ActionBar – Action Items

- ActionBar provides users access to the most important actions in the current context.
  - Items (icons and/or text) on ActionBar are called action buttons.
  - Items that can't fit on the ActionBar are hidden in the action overflow.
  - Overflow button reveals the overflow items (on the right side)

- The android system populates the ActionBar when Activity starts by calling onCreateOptionsMenu() method.
  - Inflate a menu resource inside this method.

- Each Activity (or Fragment) with action items should have its own menu resource (i.e. menu.xml) located at:
  - res/menu/menu.xml

Working with ActionBar – Action Items



```
<menu xmlns:android=http://schemas.android.com/apk/res/android
     xmlns:yourapp="http://schemas.android.com/apk/res-auto" >

  <item android:id="@+id/action_search"
      android:icon="@drawable/ic_action_search"
      android:title="@string/action_search"/>

  <item android:id="@+id/action_compose"
      android:icon="@drawable/ic_action_compose"
      android:title="@string/action_compose”
      yourapp:showAsAction="ifRoom" />
</menu>
```

Working with ActionBar – Action Items

- If you want menu items to supply both icon and text, use:
  - <item yourapp:showAsAction="ifRoom|withText" ... />
- You can use "always", but this can create problems with narrow screen devices. It's advisable to use "ifRoom" to request that items appear on the ActionBar.

Working with ActionBar – Handling clicks

- When the user clicks on an action button, the system calls the Activity's onOptionsItemSelected() method. Override this method to provide functionality, thus:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   // Handle presses on the action bar items
  switch (item.getItemId()) {
      case R.id.action_search:
         openSearch();
         return true;
      case R.id.action_compose:
         composeMessage();
         return true;
      default:
         return super.onOptionsItemSelected(item);
   }
}
```

- We will revisit ActionBar's advanced functions in Activity lesson.

Homework 4

- Interact with Views on your GridView via action buttons.
- Use your GridView homework 3
- Add an action button that creates an animation of any view object within a random cell.
- ActionBar and actions buttons
  - Add an action button with an icon to your ActionBar
  - Your action button should animate a property (or properties) of a random cell in your GridView, when clicked
    - E.g. animate the background color of a random cell, animate the color of the squares within the cells, animate the position of the squares, etc.

- Bonus points: - make an animation that loops through the entire grid.

Graphics and Animations

- Android provides powerful APIs for drawing custom 2D and 3D graphics and animations.
- Animations – android provides two animation systems
  - Property animation (introduced in Android 3.0 – API level 11)
  - View animation
- Property animation
  - Lets you animate properties of any object, including custom objects.
  - Also allows you to animate properties of objects that are not rendered to the screen
- View animation
  - View animation is the older system, only used for Views.

Drawables

- Drawable
  - A general abstraction for something that can be drawn on the screen.
  - Usually a resource retrieved for drawing things
  - Does not have facilities to receive events or interact with the user

Drawables

- Drawables can take these forms:
  - Bitmap: the simplest form, usually PNG of JPEG image
  - Nine Patch: a PNG format extension, which specifies information about how to stretch it and place things inside of it
  - Shape: contains simple drawing commands instead of a raw bitmap, allowing it to resize better in some cases
  - Layers: compound drawable, which draws multiple underlying drawables on top of each other
  - States: a compound drawable that selects one of a set of drawable based on its state
  - Levels: a compund drawable that selects one of a set of drawables based on its level
  - Scale: a compound drawable with a single child drawable, whose overall size is modified base don the current level.

Drawable Animations

- In addition to the two animation systems, Android provides a Drawable Animation, which involves displaying Drawable resources in succession.

- Hardware Acceleration
  - Introduced in Android 3.0.
  - Provides GPU based rendering, which increases performance, but consumes more RAM.
  - More info:
    - http://developer.android.com/guide/topics/graphics/hardware-accel.html
  - Enabled by default, if your app is >= API level 14.
  - For other platforms, you can control at the following levels:
    - Application
    - Activity
    - Window
    - View

Animations

- In addition to the two animation systems, Android provides a Drawable Animation, which involves displaying Drawable resources in succession.

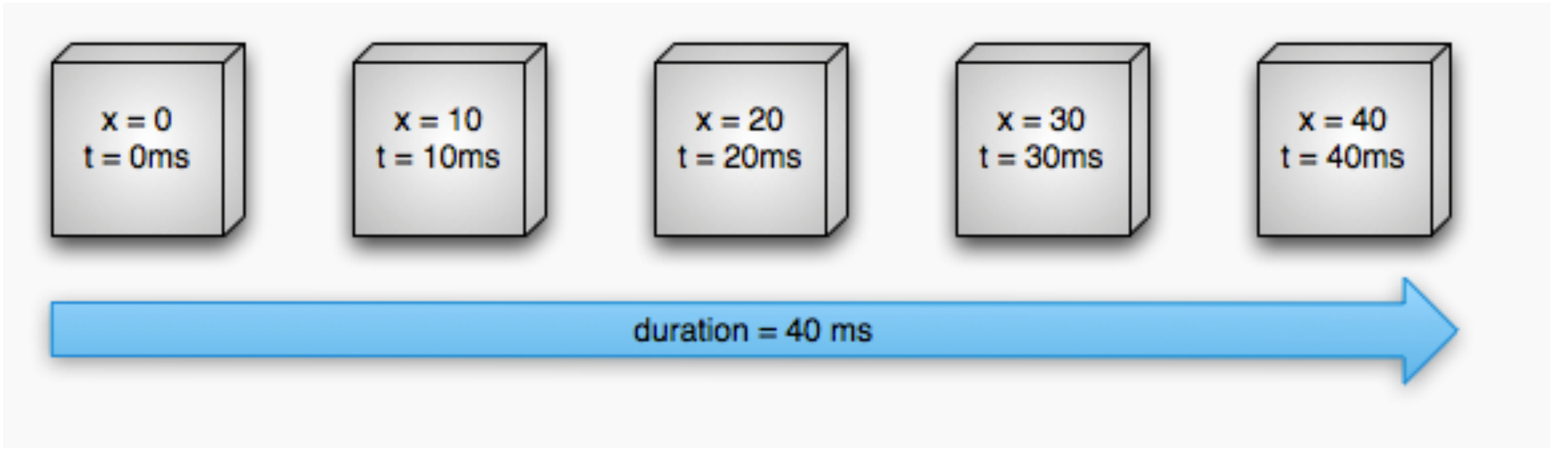
- Hardware Acceleration
  - Introduced in Android 3.0.
  - Provides GPU based rendering, which increases performance, but consumes more RAM.
  - More info:
    - http://developer.android.com/guide/topics/graphics/hardware-accel.html
  - Enabled by default, if your app is >= API level 14.
  - For other platforms, you can control at the following levels:
    - Application
    - Activity
    - Window
    - View

Animations

- Property animation - Change value of a property over time.
  - Duration: default length is 300ms
  - Time interpolation: specifies how the values for the property are calculated, as a function of the animation's current elapsed time
  - Repeat count: you can specify whether or not to have an animation repeat when it reaches the end of the duration, and how many times to repeat. Can also specify reverse.
  - Animator sets: grouping animations into logical sets that play together or sequentially, or after specified delays
  - Frame refresh delay: specifies how often to refresh frames

Linear Animation example

- Animation object from x position by 40 pixels in 40ms



- ValueAnimator object keeps track of the animation timing.
  - TimeInterpolator: defines time interpolator e.g. LinearInterpolator
  - TypeEvaluator: defines how to calculate the values e.g. IntEvaluator
- To start, give ValueAnimator start and end values for properties, along with duration of animation, then call start();

Object Animation example

```
private static final int RED = 0xffFF8080;
private static final int BLUE = 0xff8080FF;

ValueAnimator colorAnim = ObjectAnimator.ofInt(myTextView, "backgroundColor", RED, BLUE);
colorAnim.setDuration(3000);
colorAnim.setEvaluator(new ArgbEvaluator());
colorAnim.setRepeatCount(ValueAnimator.INFINITE);
colorAnim.setRepeatMode(ValueAnimator.REVERSE);
colorAnim.start();
```

OR in XML

Additional Reading

- http://developer.android.com/guide/topics/graphics/prop-animation.html

- Android 5.0 (Lollipop) Material Design

https://developer.android.com/design/material/index.html

- New Material Design paradigm from Google (interesting read)

http://www.google.com/design/