Week 1: UI Design and Layouts

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

Agenda

- Week 1 review
- User Interface Introduction
  - Multiscreen considerations
  - Android Application – app structure
  - Activities, Services, Receivers and Providers
  - User Interface Layout
  - AndroidManifest
  - Resources
  - Layout Resource files
  - Layout Manager and ViewGrops

Week 1 review

- Introduction to Android
- SDK & environment setup
- First app
- Logging & debugging
  - Android Log()
  - adb logcat

UI – multi screen considerations

- Android is multi-screen capable
  - Devices with various sizes and densities
  - Design goals -> compatible with all screen sizes
  - Optimize for various screen configurations

- Android provides a consistent dev. env.
  - Automatically handles most of the adaptive work
  - System APIs allow control of UI for specific sizes
  - User experience: optimize for their SPECIFIC device

- New APIs for layout control
  - Tablets were introduced in Android 3.2.
  - New APIs introduced in 3.2 supporting tablet sizes

UI – Screens Support Overview

- Screen size
  - Physical size, measured as the screen's diagonal length in pixels
  - Four categories: small, normal, large, extra-large

- Screen density
  - Quantity of pixels within a physical area of the screen
  - Usually referred to as dpi (dots per inch)
  - Six categories: low, medium, high, extra-high, extra-extra-high, extra-extra-extra-high

- Orientation
  - Orientation of device from user's perspective
  - Portrait and landscape

- Resolution
  - Total number of physical pixels on a screen.
  - Applications are only concerned with screen size and density

- Density-independent pixel (dp)
  - Virtual pixel unit you **should** use when defining layouts. **Never use pixels!!!!!**
  - 1dp = 1 pixel on 160dpi screen, which is a medium density and baseline for Android

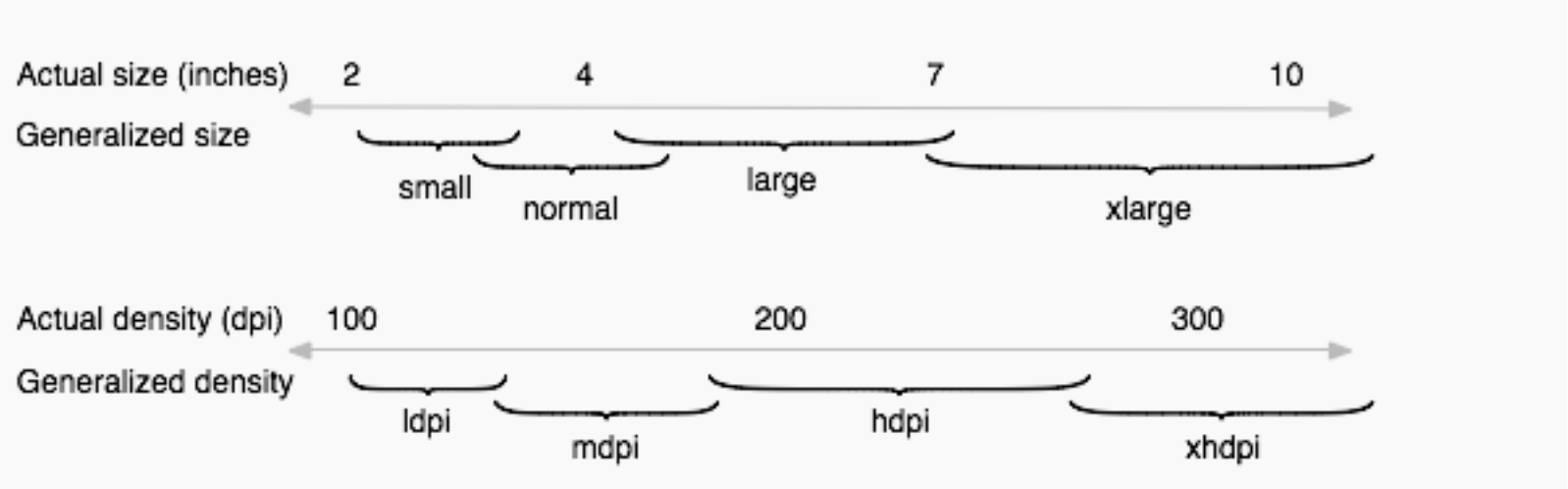
UI – Range of Screens Supported

- Screen sizes
  - small, normal, large, xlarge

- Generalized screen density categories
  - ldpi (low) ~120dpi
  - mdpi (medium) ~160dpi
  - hdpi (high) ~240dpi
  - xhdpi (extra-high) ~320dpi
  - xxhdpi (extra-extra-high) ~480dpi
  - xxxhdpi (extra-extra-extra-high) ~640dpi

- Arranged around baseline configuration that is:
  - Normal size
  - Medium density

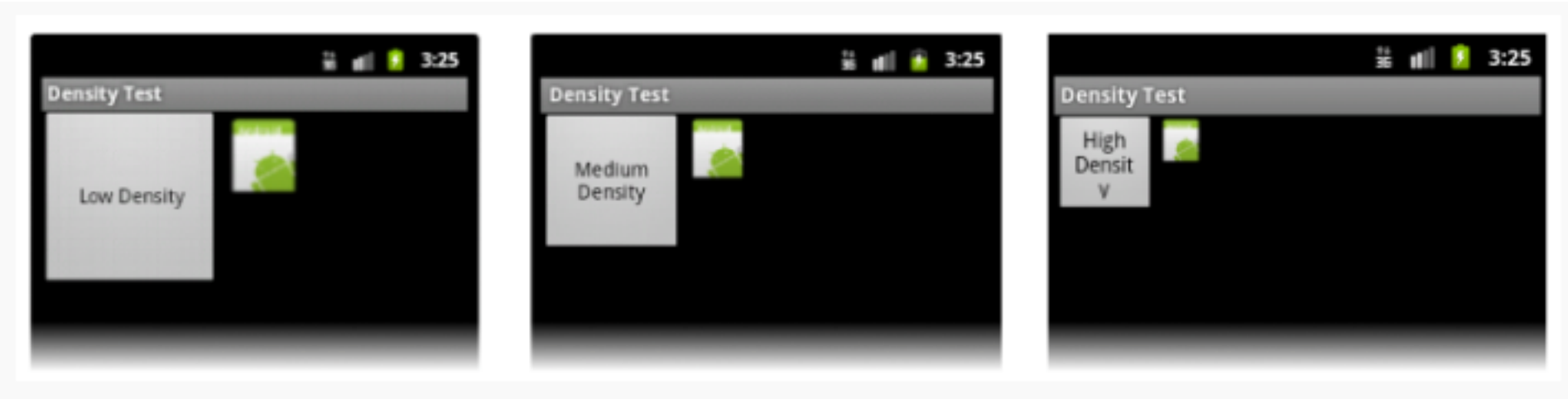
Generalized size and density categories



- xlarge: at least 960dp x 720dp
- large: at least 640dp x 480dp
- Normal: at least 470dp x 320dp
- Small: at least 426dp x 320dp

Density Independence

- Preservation of physical size (from a user's POV) of UI elements when displayed on screens with different densities.



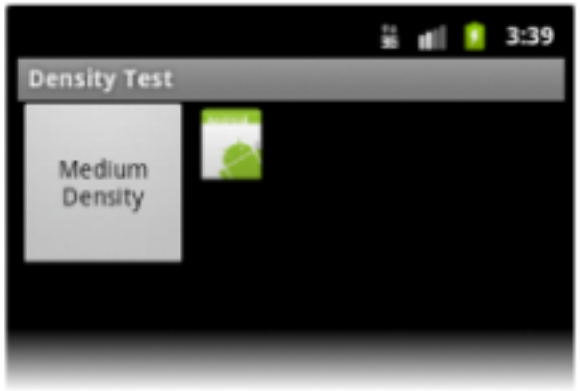
Density Independence

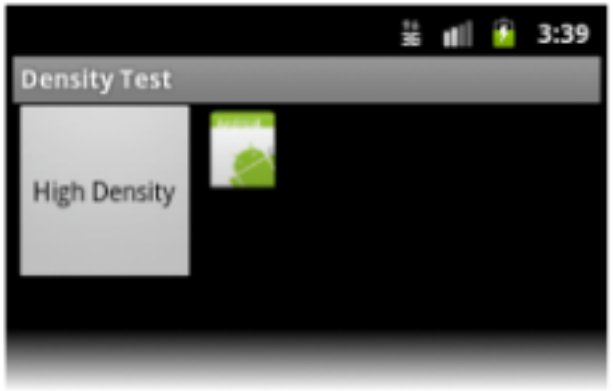
- Android system achieves DI in two ways
  - “dp” units are scaled for current screen density
  - drawable resources are scaled to appropriate size, based on current screen density
- Specifying dims in pixels (px) will result in varying views (larger on low density screens, and smaller on on high density screens)

Density Independence

- Dims specified in density-independent units (dp)
  - Baseline is medium, hence looks same
  - Low density scales down
  - High density scales up

- "scaled" bitmaps could get blurry
  - Provide alternative bitmaps for different densities





Parts of an Android Application

- An Android application is a single installable unit, which an be started and used independent os other Android applications.

- Has one Application Class, which is instantiated as soon as the application starts, and is the last component which is stopped on shutdown

- Consists of android software components and Resource files

Android software Components

- Activity
- Services
- Broadcast Receivers (or simply Receivers)
- Content Providers (or simple Providers)

Context

- Context class provides connection to the Android system which executes the application.
- Also provides access to Android services
- Activities and services extend the Context class, hence they can be directly used to access the Context.
- class android.content.Context
  - http://developer.android.com/reference/android/content/Context.html

App components overview

- Activity
  - Visual representation of an Android application
  - An application can have several activities
  - Activities use Views and Fragments to create UIs and interactions
  - http://developer.android.com/reference/android/app/Activity.html

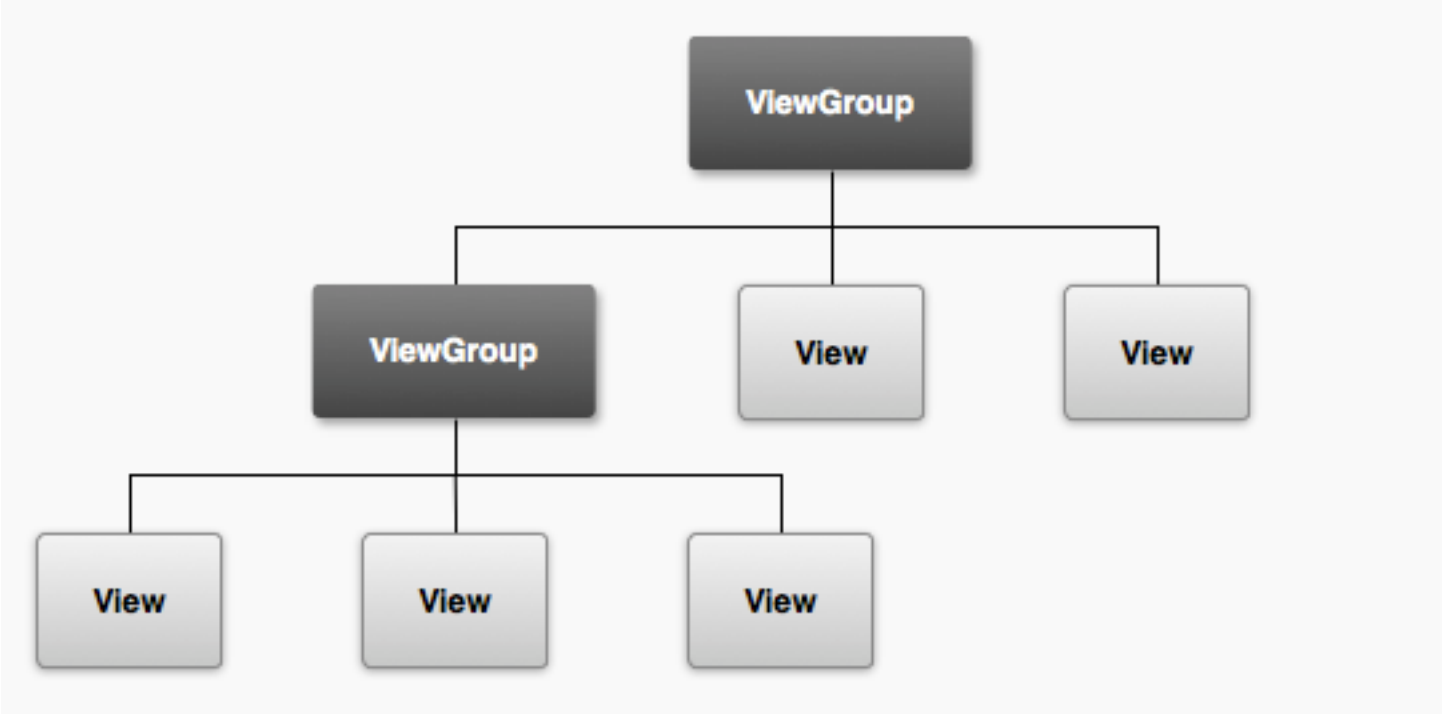
- BroadcastReceiver
  - Can be registered to listen to system messages and intents
  - E.g. bootup completed, incoming phone call, etc.
  - http://developer.android.com/reference/android/content/BroadcastReceiver.html

App components overview

- Service
  - Performs a task without providing a UI.
  - E.g. background service
  - http://developer.android.com/reference/android/app/Service.html
- ContentProvider
  - Defined a structured interface to application data
  - Analogous to an API
  - http://developer.android.com/reference/android/content/ContentProvider.html

User Interface Layout

- UI for each component is defined as a hierarchy of View and ViewGroup objects

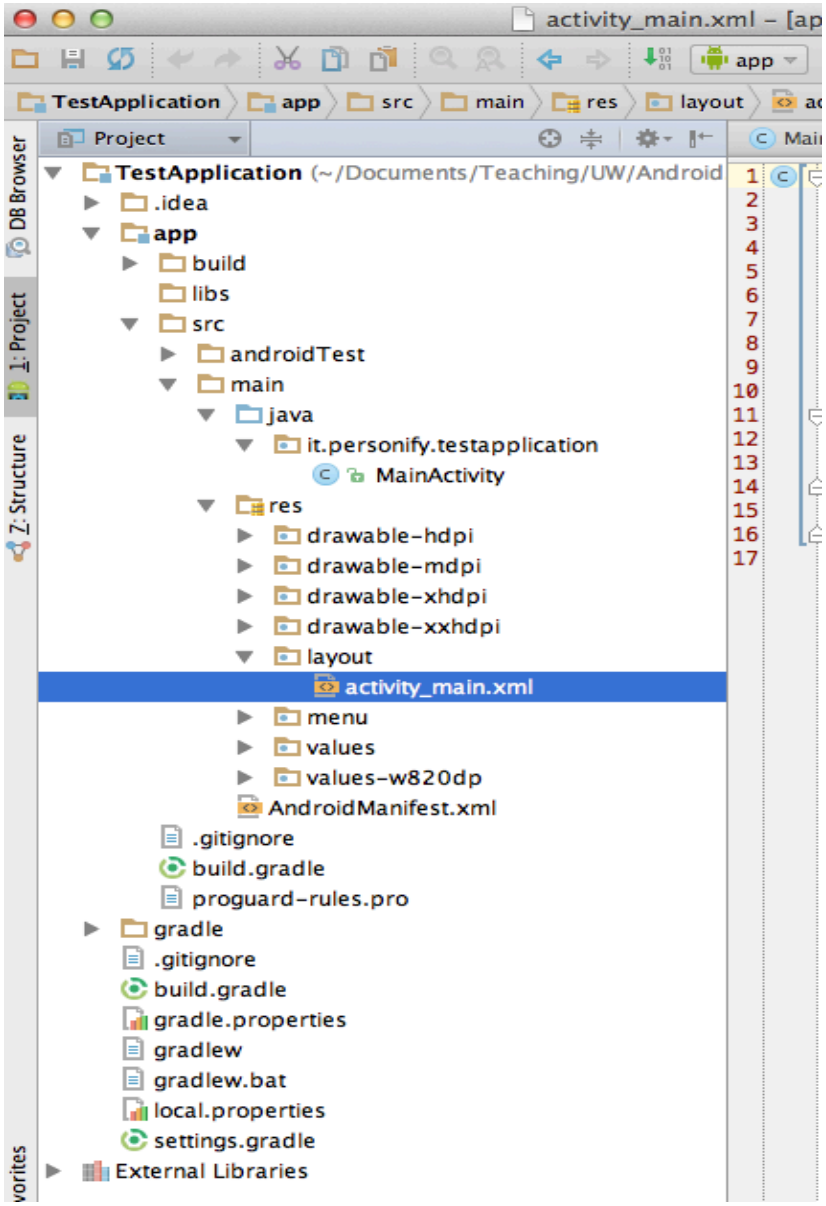


Views & ViewGroup

- Views
  - UI widgets e.g. buttons or text fields
  - Configurable attributes to configure appearance

- ViewGroup
  - Arranges other views
  - Also known as layout manager
  - Base class is android.view.ViewGroup
  - Extends android.view.View, which is a base for all views.
  - Can be nested to form complex layouts

App Structure (Android Studio)



App Structure: source files

- src/main/java
  - Contains all sources for the application
  - Activities and java sources are arranged under “packages”
  - Project can contain multiple packages

Resources

Resource	Folder
Drawables	*/res/drawables*
Simple Values	*/res/values*
Layouts	*/res/layout*
Styles and Themes	*/res/values*
Animations	*/res/animator*
Raw data	*/res/raw*
Menus	*/res/menu*

Resources

- Drawable Resources
  - Stores image files used by the app
  - Various folders prefixed with “drawable-” for each density
- Layout Resources (res/layout)
  - Activity screens layouts in XML

Resources

- Data Resources
  - Folders with “values” prefix
  - Contain data values you wish to use within the app
  - Eg. Text strings, constants, etc.
  - Can differentiate values per screen sizes and API levels
  - If same values can be used across devices, it can be saved in the plain “values” folder

Resource IDs and R.java

- Every resource gets an ID assigned by the Android build system.
- gen directory contains R.java which contains the generated values
  - Static integer values
- New resource files automatically adds references in R.java
- System provides methods to access corresponding resource files via their IDs

Layouts

- Defines visual structure for a UI
- Can be declared in two ways
  - Declared UI elements in XML
    - 1-1 mapping in android XML -> Views/ViewGroups
  - Instantiate layout elements at runtime
    - Apps can create Views and ViewGroups programmatically
- Using XML is easier
  - makes code more “modular”, separating presentation from behavior control
  - No need to modify source code or recompile

Layouts - XML

- Each layout must contain ONE root element
  - Must be a View or View Group

- Can add multiple layout objects or widgets as child objects, gradually building simple or complex view hierarchy for the layout.

Layout XML example

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
   <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
   <Button android:id="@+id/button"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Hello, I am a Button" />
</LinearLayout>
```