Android for .NET Developers Series
Adopting the Android Mindset

Dynamically adapting to device differences

Jim Wilson
hedgehogjim.wordpress.com
@hedgehogjim
jimw@jwhh.com

pluralsight
hardcore developer training

Outline

- The need for app adaptability
- Android adaptability
- Density independence
- Android resource system adaptability
- Dealing with different size displays
- Resource aliasing

Adopting the android mindset

Be

Be adaptable

Be adaptable

adaptable

The need for adaptability

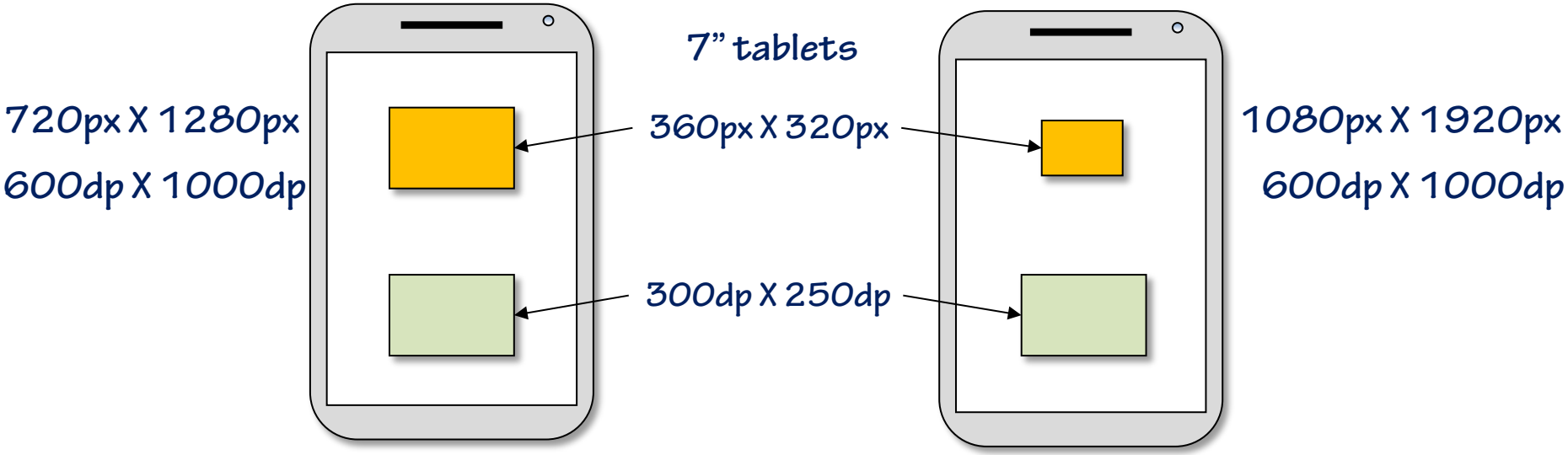
- **Widely varying device experiences**
  - New classes of devices emerging
    - Not just phones anymore
    - Tablet-based computing continues to grow
    - Devices will continue to evolve and change
  - Screen sizes
    - As platform evolves screen resolution and physical size rapidly changing
    - Larger versus smaller screens
    - Portrait versus landscape oriented screens

Adaptability is fundamental to Android

- **Android provides 4 key features to enable adaptability**
  - Layout management
  - Fragments
  - Density independence
  - Dynamic resource selection

Density independence

- **Android canonicalizes screen measurement**
  - Devices of a similar physical size often have very different resolutions
  - Expressing View dimensions in pixels creates inconsistent user experience
  - Density independent pixels create consistency
    - Creates a normalized unit equal to a 160 dpi display
    - Android handles device independent pixel and physical pixel translation
    - Abbreviated as dp



Dynamic resource selection

- **Android supports a highly adaptable resource system**
  - Can create most UI aspects as resources
    - Layouts, menu, strings, images, dimensions, and more
  - Android tracks a variety of device characteristics
    - Screen density
    - Orientation
    - OS Version
    - Screen size
    - Many others…
  - Can associate resource files with characteristics
    - Associated by adding suffix to resource folder
      - Can combine characteristics
    - Android will automatically select appropriate resource for characteristics

When size matters

- **Applications must adapt to available display size**
  - Android provides 2 ways of associating resources with display size
  - Screen size groups
    - General grouping of screen sizes
    - Available to all devices
    - Problematic
  - Screen size qualifiers
    - Specific screen size qualifiers
    - Available to API Level 13 (Android 3.2) and above
    - Provides close control

Summary

- **App adaptability is essential to providing a quality user experience**
- **Android provides features that simplify creating adaptable apps**
- **Density independent pixels**
  - Normalized unit equal to 160 dpi display
- **Android selects resources based on device characteristics**
  - Resources associated with characteristics based on directory name
- **Screen size groups associate resources based on general categories**
  - Supported by all Android versions
  - Can be problematic
- **Screen size qualifiers associate resources based on specific size info**
  - Added at API Level 13 (Android 3.2)
  - Provides close control
- **Resource aliasing helps reduce redundancy in resource files**

Fragments

- **Fragments enable dividing the UI into sections**
  - Provide a group of user interface elements and their behavior
    - Can hold the same UI elements as an Activity
    - Can contain logic
  - Create logical UI units
  - Allow reorganizing UI for device differences
    - Can use static layout files
    - Can be assembled dynamically in code
  - A key part of a many navigation behaviors
    - Page-based navigation
    - Tab navigation
    - List-based navigation

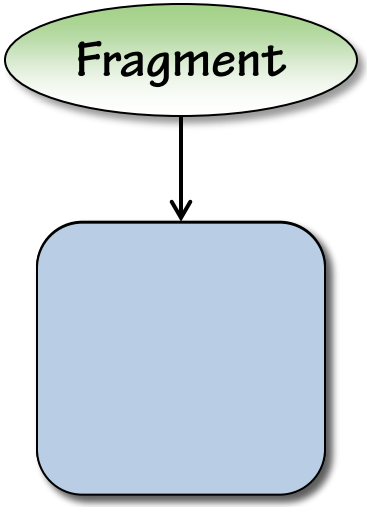
Fragment Availability

- **Fragments supported by 99.9% of active Android devices**
  - Native OS support for devices running Android 3.1 or newer
    - API level 12 and above
  - Compatibility library support for devices running Android 1.6 through 2.3
    - API levels 4 through 10
    - Available from …
      - http://bit.ly/AndroidV4SupportLib
    - Example use …
      - http://bit.ly/AndroidFragmentCompat

*Although Android 3.0 (API Level 11) technically exists – no devices in the marketplace run it

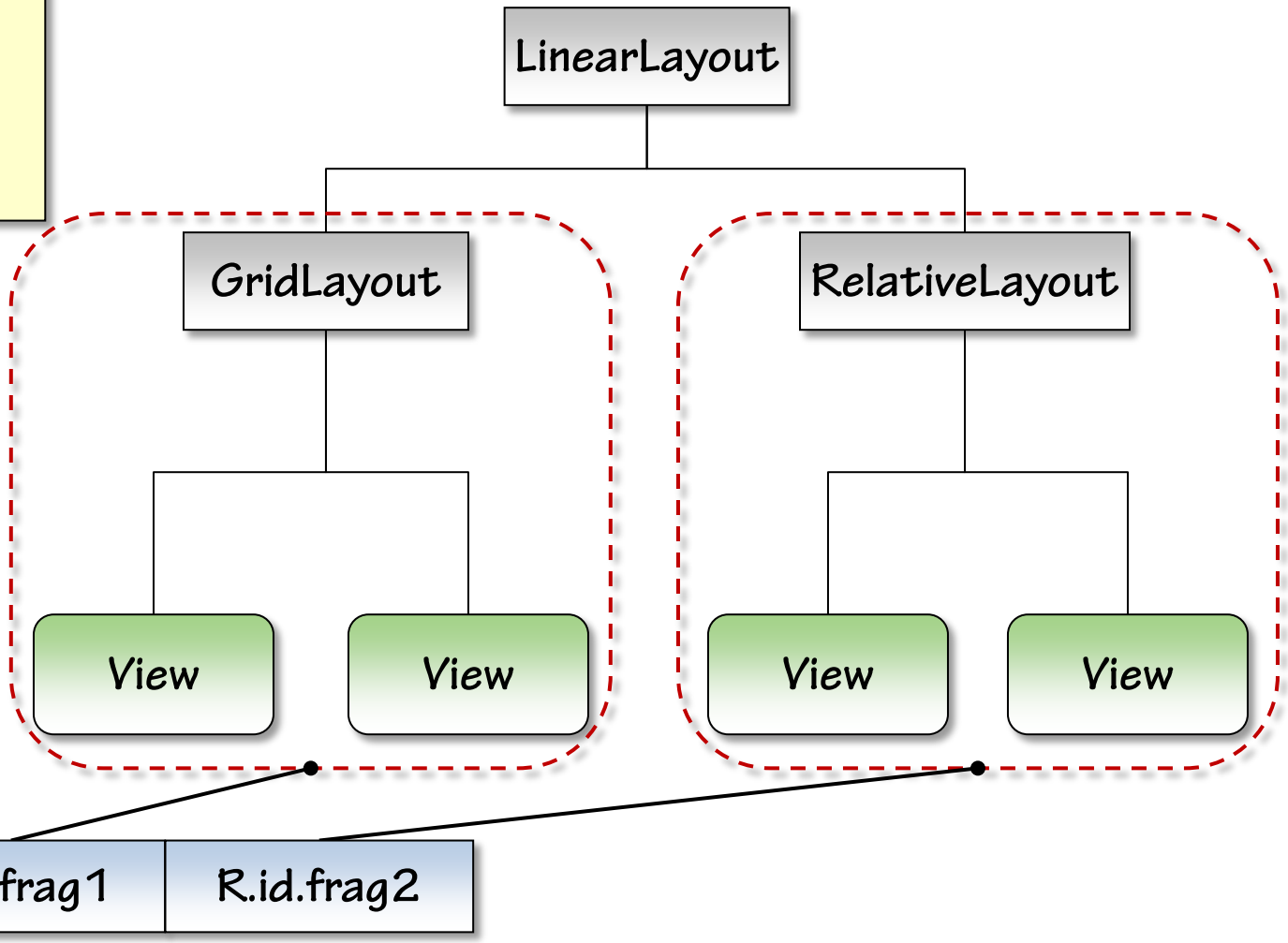
Creating Fragments

- **Creating a Fragment requires a few simple steps**
  - Declare a class that inherits from Fragment class
  - Provide the fragment's display contents
    - Involves overriding onCreateView and/or onCreate
    - Often uses an XML-based layout description much like an Activity
    - Specialized Fragment-derived classes simplify special cases
      - More to come on this
  - Attach your fragment to an Activity
    - Use the <fragment> element in the Activity's XML layout
      - The "class" attribute identifies the fully qualified name of fragment class to create
    - You can attach as many fragments to a view as you'd like



Structure of an Activity containing Fragments

```
<LinearLayout ...>
  <fragment
    name="com.ps.frag1"
    id="@+id/frag1" .../>
  <fragment
    name="com.ps.frag2"
    id="@+id/frag2" .../>
</LinearLayout>
```



FragmentManager