FPT Software

# ANDROID TRAINING
# LESSON 2

Version 0.1

# Agenda

- Layout overview
- LinearLayout
- RelativeLayout
- TableLayout
- ListView
- ScrollView
- GridView
- Gallery Control
- Spinner Control

# Main Layout Strategies

- **XML-based**
  - Declare layout in res/layouts/some_layout.xml
    - Set various XML properties
    - Use visual editor in Eclipse
  - Load with setContentView(R.layout.some_layout)
- **Java-based**
  - Instantiate  layout, set properties, insert sub-layouts

    ```
    LinearLayout window = new LinearLayout(this);
    window.setVariousAttributes(…); window.addView(widgetOrLayout);
    ```
  - Load with setContentView(window)
- **This tutorial**
  - Uses XML-based approach. However, attributes can be  adapted for Java-based approach.

# XML Layout Attributes

- Idea
  - Each Layout class has an inner class called LayoutParams that defines general XML parameters that layout uses. These parameters are always named android:layout_blah, and usually have to do with sizes and margins.
  - Layout classes define more specific attributes. Many inherited from LinearLayout (which extends ViewGroup and View).
    - Not named beginning with “layout_”
- Example

```
<LinearLayout
   android:layout_width="match_parent"
    android:layout_height="wrap_content"y _ gp_
    android:gravity="center_horizontal"
    android:background="@color/color_1">…<LinearLayout>
```

# Commonly Used Attributes

- **Size**
  - android:layout_height, android:layout_width
    - match_parent: fill the parent space (minus padding)
    - wrap content: use natural size (plus padding)
    - An explicit size with a number and a dimension. See margins on next slide.
  - android:layout_weight
    - A number that gives proportional sizes. See example.

# Commonly Used Attributes

- Alignment
  - android:layout_gravity
    - How the View is aligned within containing View.
  - android:gravity
    - How the text or components inside the View are aligned.
  - Possible values
    - top, bottom, left, right, center_vertical, center_horizontal, center (i.e., center  both ways), fill_vertical, fill_horizontal, fill (i.e., fill both directions),  clip_vertical, clip_horizontal

# Commonly Used Attributes

- Margins (blank space outside)
  - android:layout_marginBottom, android:layout_marginTop, android:layout_marginLeft, android:layout_marginRightandroid:layout marginRight
- Padding (blank space inside)
  - android:paddingBottom, android:paddingTop, android:paddingLeft, android:paddingRight
- Units (e.g., "14.5dp")
  - dp: density-independent pixels (scaled by device resol.)
  - sp: scaled pixels (scaledbased on preferred font size)
  - px: pixels
  - in: inches
  - mm: millimeters

# Commonly Used Attributes

- ID
  - android:id
    - Used if the Java code needs a reference to View
    - Used in RelativeLayout so XML can refer to earlier ids
- Colors
  - android:background (color or image, for any Layout)
  - android:textColor (e.g., for TextView or Button)
  - Common color value formats
    - "#rrggbb" "#aarrggbb", "@color/color name“
- Click handler
  - android:onClick
    - Should be a public method in main Activity that takes a  View (the thing clicked) as argument and returns void

# LinearLayout

- Idea
  - Put components in a single row or single column
  - By nesting, can have rows within columns, etc.
- Most important XML attributes
  - android:orientation
    - "horizontal" (a row) or "vertical" (a column)
    - horizontal is the default, so can be omitted for rows
  - android:gravity
    - How the Views inside are aligned
    - Possible values
    - top, bottom, left, right, center_vertical, center_horizontal, center (i.e., center both ways), fill_vertical, fill_horizontal, fill (i.e., fill both directions), clip_vertical, clip_horizontal

# LinearLayout

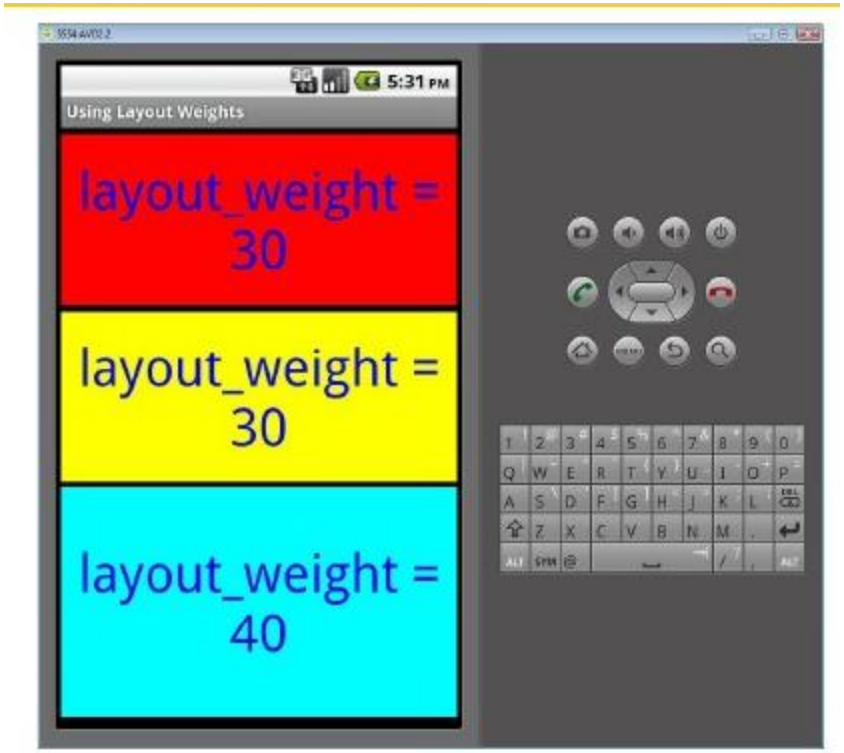
- android:layout_weight
  - Assign android:layout_height to 0dp
  - Use relative values for android:layout_weight
  - Analogous approach to set widths

- **Example**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="…"xmlns:android …
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
      <TextView
    android:layout_width="match_parent"

      android:layout_height="0dp“
      android:layout_weight="30"  .../>
      <TextView
    android:layout_width="match_parent“
      android:layout_height="0dp“
      android:layout_weight="30“  .../>
      <TextView android:layout_width="match_parent"
      android:layout_height="0dp"
       android:layout_weight="40"   .../>
</LinearLayout>
```

# LinearLayout



# RelativeLayout

- Idea
  - Give ids to 1 or more key components (id="@+id/blah")
  - Position other components relative to those components
- Most important XML attributes
  - Aligning with container
    - android:layout_alignParentBottom (and Top, Right, Left)
    - android:layout_centerInParent (and centerHorizontal, centerVertical)
    - These all take "true" or "false" as values
  - Aligning with other component
    - android:layout_alignBottom (and Top, Right, Left)
    - android:layout_toLeftOf (and toRightOf), android:layout_above (and below)
    - These all take existing ids as values

# RelativeLayout

- First component
  - <Button id="@+id/button_1" android:layout_alignParentRight="true" …/>
- Second component
  - <Button android:layout_toLeftOf="@id/button 1" />
- Result

# TableLayout

- Idea
  - Put widgets or nested layouts in a grid. No borders.
  - Like HTML tables, the number of rows and columns is determined automatically, not explicitly specified.
  - Components are usually placed inside TableRow
- Most important XML attributes (TableLayout)
  - android:stretchColumns
    - An index or comma-separated list of indexes. Specifies the column or columns that should be stretched wider if the table is narrower than its parent. Indexes are 0-based.
  - android:shrinkColumns
    - Column(s) that should be shrunk if table is wider than parent.
  - android:collapseColumns
    - Column(s) to be totally left out. Can be programmatically put back in later.

# TableRow

- Idea
  - Goes inside TableLayout to define a row.
    - Technically, elements between rows are permitted, but you can achieve same effect with a TableRow and android:layout_span.
- Most important XML attributes of elements inside a TableRow
  - android:layout_column
    - Normally, elements are placed in left-to-right order. However, you can use android:layout_column to specify an exact column, and thus leave earlier columns empty.
  - android:layout_span
    - The number of columns the element should straddle. Like colspan for HTML tables.
    - There is nothing equivalent to HTML's rowspan; you must use nested tables instead

# ListView control

- Idea:
  - The ListView control displays a list of items vertically.
- The usage pattern for ListView:
  - Writing a new activity that extends android.app.ListActivity. ListActivity contains a ListView.
  - setListAdapter(): set the data for the ListView

# ScrollView

- In the case: you cannot fit all the views on one screen, it's often useful to allow scrolling in order to fit more elements in a single activity.
- Wrap the existing LinearLayout contents in a ScrollView:

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true" >
      <LinearLayout>
      <!-- Rest of code here -->
      </LinearLayout>
</ScrollView>
```

- Fillviewport Attribute
  - cause the child views of a ScrollView to expand to the size of the display

# GridView control

- Idea:
  - displays information in a grid.
- The usage pattern for the GridView:
  - to define the grid in the XML layout and then bind the data to the grid using an android.widget.ListAdapter.

# The Gallery Control

- Idea:
  - Create a horizontally scrollable list control that always focuses at the center of the list.
- The usage
  - instantiate a Gallery either via XML layout or code:

```
<Gallery
   android:id="@+id/galleryCtrl"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
/>
```

  - Using the Gallery control is similar to using a list control: call the setAdapter() method to populate data

# The Spinner Control

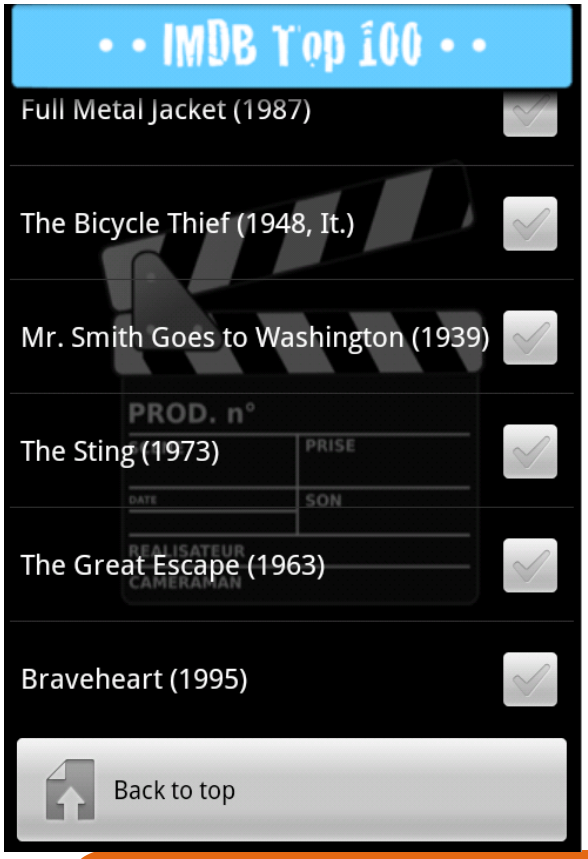
- Idea: like a dropdown menu.
- Usage:
  - You can instantiate a Spinner either via XML layout or code:

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
```

  - Using the Spinner control is also similar to using a list control: call the setAdapter() method to populate data

# Homework

- Create a application as following Layout



# Thank you!