



Programming with Android: User Interface & Layouts

Introduction to User Interface (UI)

- The user interface of an application is everything that the user can see and interact with.
- Android provides a variety of pre-built UI components such as structured layout objects and UI controls.
- E.g : dialogs, notifications, and menus.

Two UI Approaches

Procedural	Declarative
Write Java code.	Write XML code Similar to HTML of a webpage.

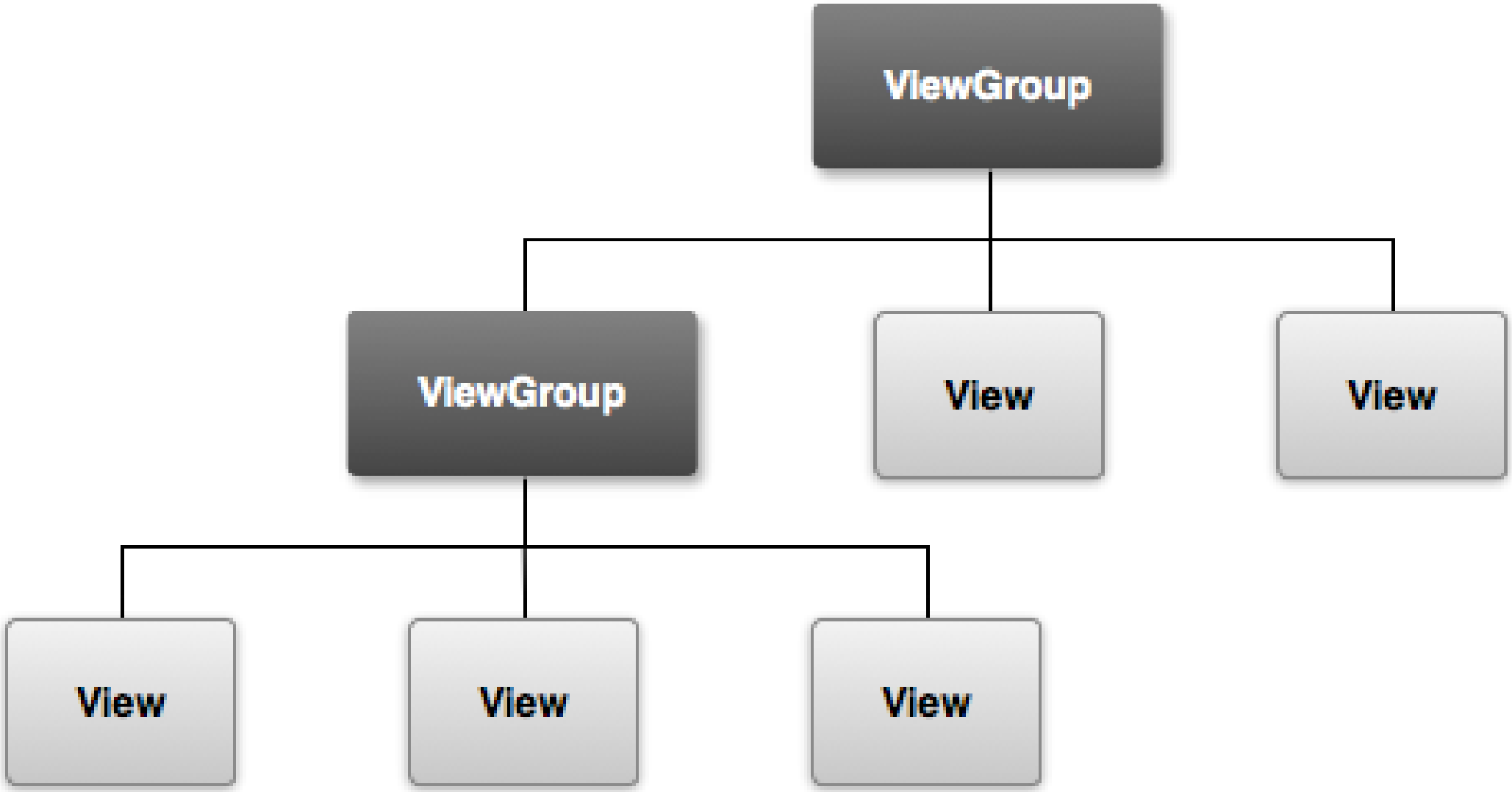
Two styles can be mixed :

- Start with XML and declare most of UI.
- Switch to Java and implement UI logic.

UI Controls

Some of the basic UI controls are :

- TextView
- EditText
- Button
- Radio Button
- Checkbox
- Spinner


ViewGroup
ViewGroup
View
View
View
View
View

Views : Outline

- Main difference between a Drawable and a View is reaction to events
- Could be declared in an XML file
- Could also be declared inside an Activity
- Every view has a unique ID
- Use findViewById(int id) to get it
- Views can be customized

ViewGroup and layout

- ViewGroup is a view container
- It is responsible for placing other views on the display
- Every layout must extend a ViewGroup
- Every view needs to specify:
  - android:layout_height
  - android:layout_width
  - A dimension or one of match_parent or wrap_content

Android - Event Handling

- There are three concepts related to Android Event Handling:

Event Listeners − An event listener is an interface in the View class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.

Event Listeners Registration − Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.

Event Handlers − When an event happens and we have registered and event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

Android Layouts

What is a Layout?

A Layout defines the visual structure for a user interface, i.e. it handles the arrangement of components on the screen. Layouts can be declared in two ways:

1. **Declare UI elements in XML:** Android provides a straightforward method of declaring layouts in XML file.
2. **Instantiate layout elements at runtime:** An application can declare layouts (and manipulate their properties) programmatically.

Layouts

- Some layouts are pre-defined by Android
- Some of these are
  - LinearLayout
  - RelativeLayout
  - TableLayout
  - FrameLayout
  - AbsoluteLayout
- A layout could be declared inside another layout

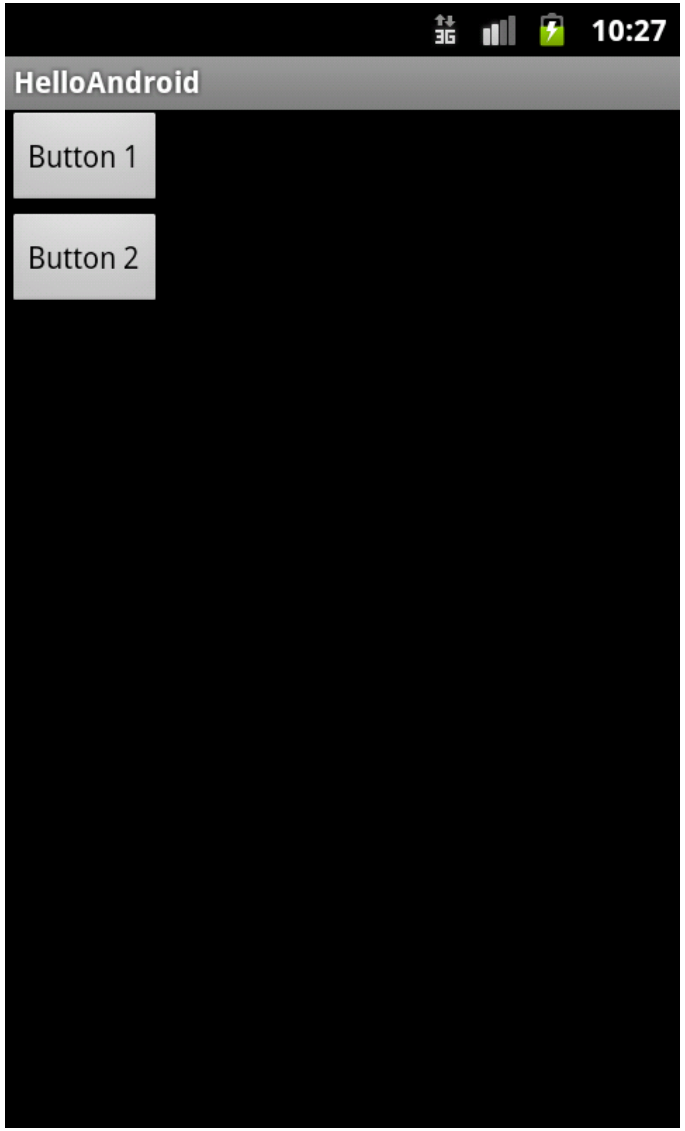
LinearLayout

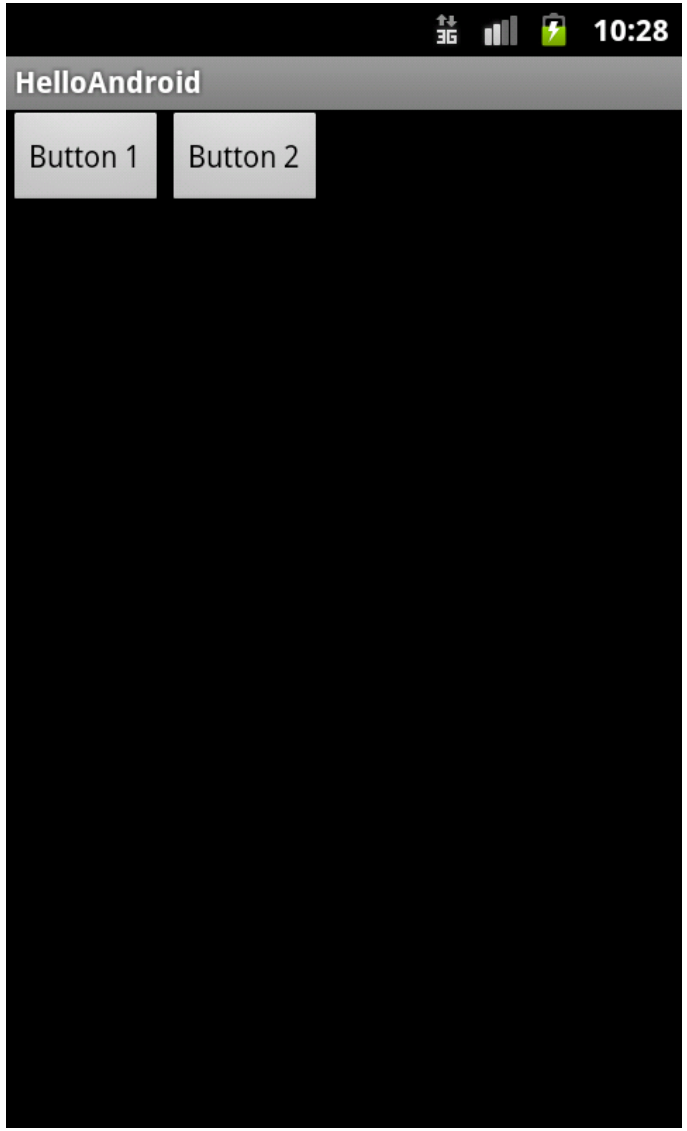
- Dispose views on a single row or column, depending on android:layout_orientation
- The orientation could also be declared via setOrientation(int orientation)
  - orientation is one of HORIZONTAL or VERTICAL
- Has two other attributes:
  - gravity
  - weight

LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >                <!-- Also horizontal -->

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString2" />
</LinearLayout>
```

LinearLayout

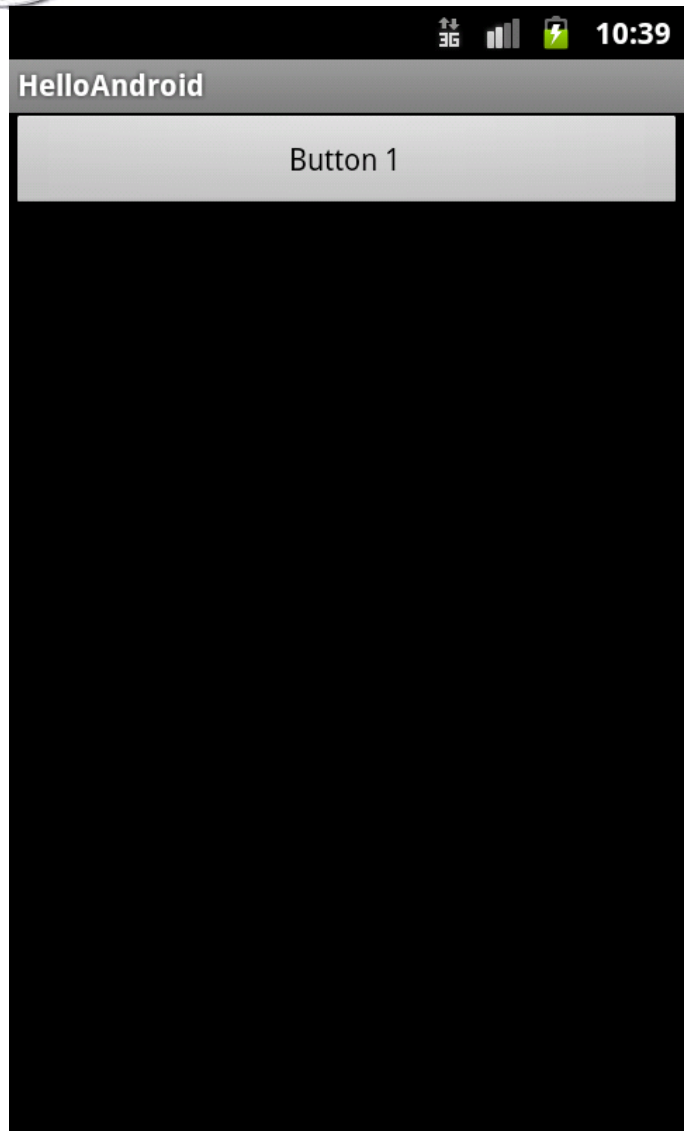


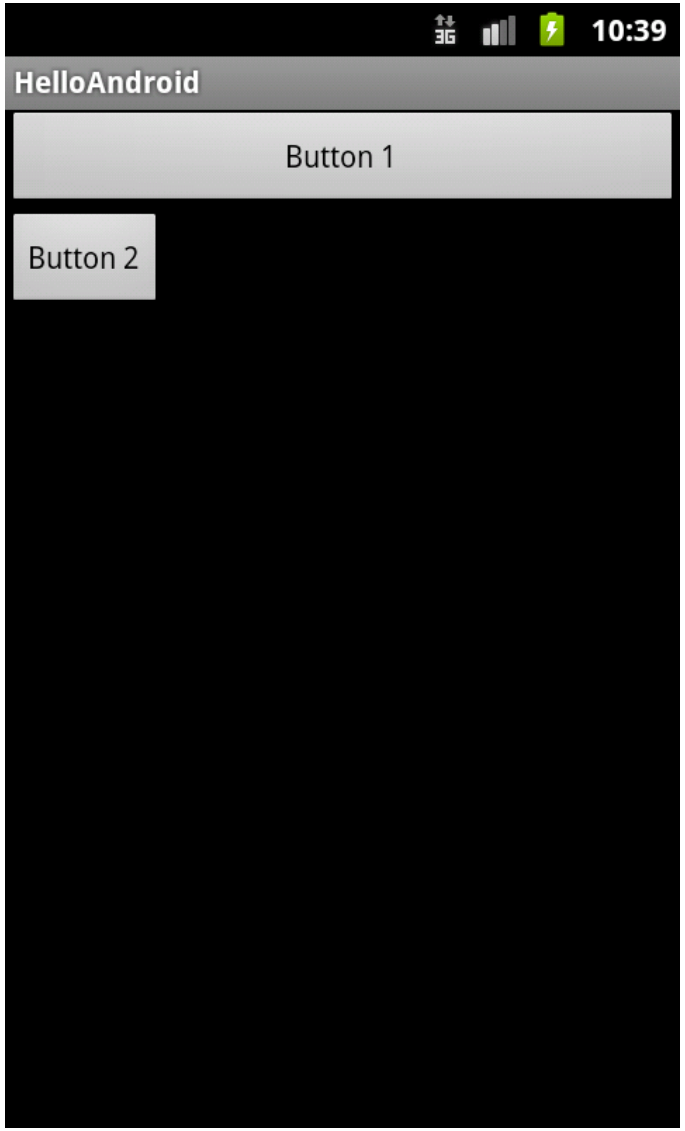


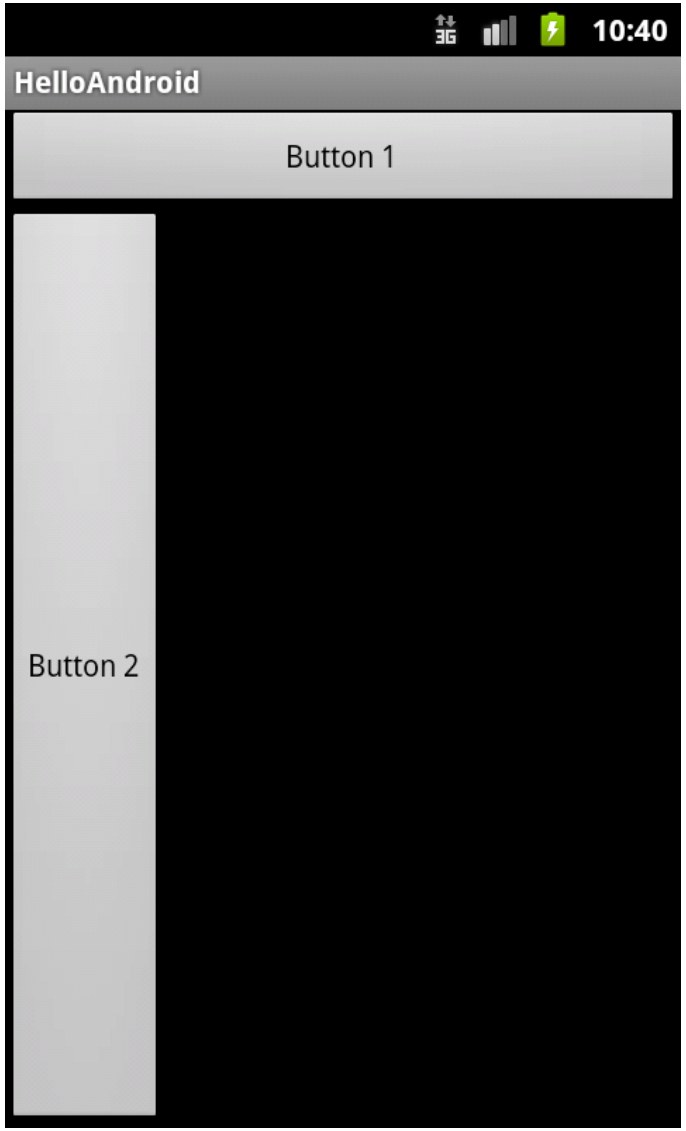
LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/buttonString1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:text="@string/buttonString2" />
</LinearLayout>
```

LinearLayout



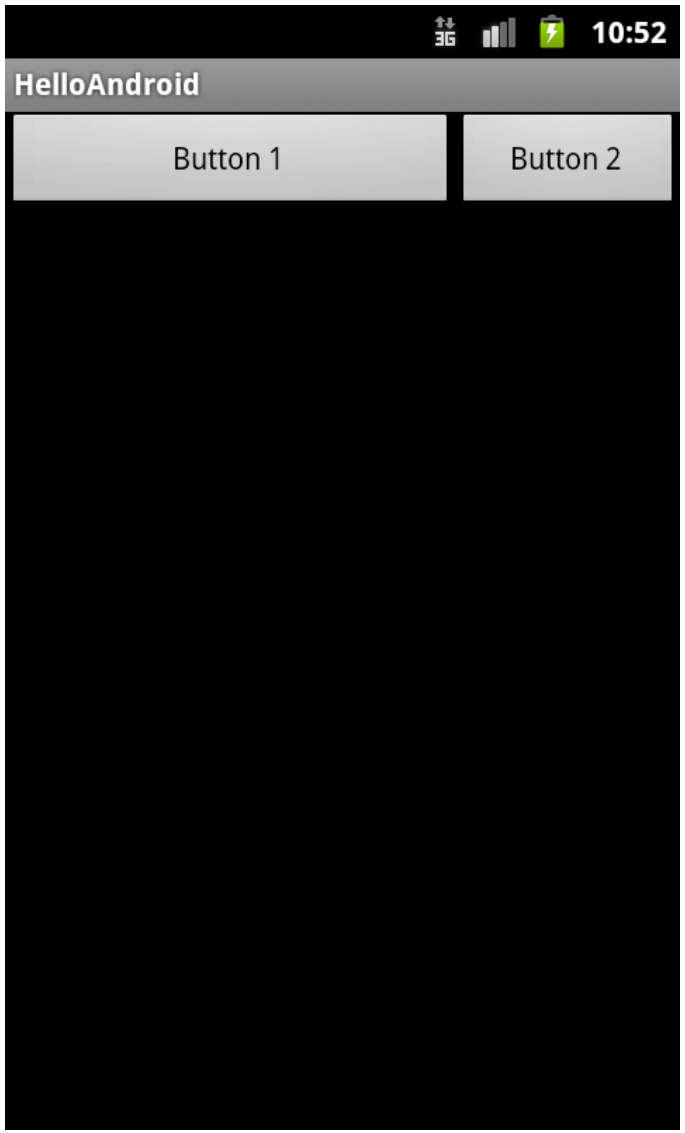


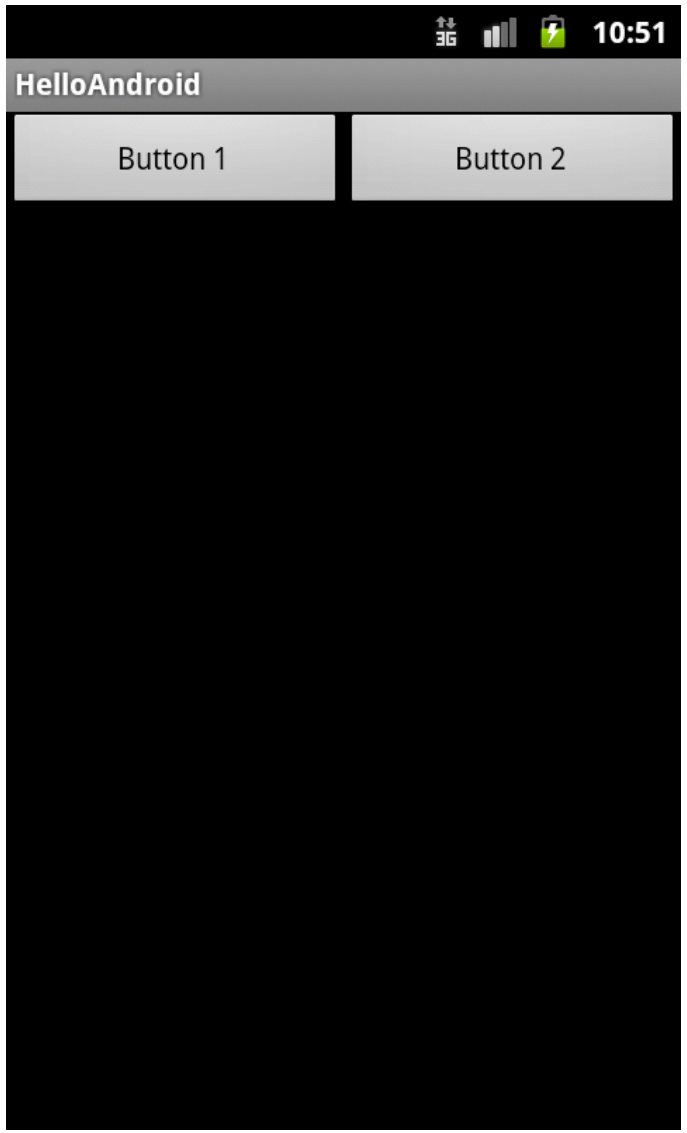


LinearLayout weight

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"    android:layout_height="fill_parent"    android:orientation="horizontal" >

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/buttonString1"
        android:layout_weight="1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/buttonString2"
        android:layout_weight="2"        />
</LinearLayout>
```

LinearLayout weight

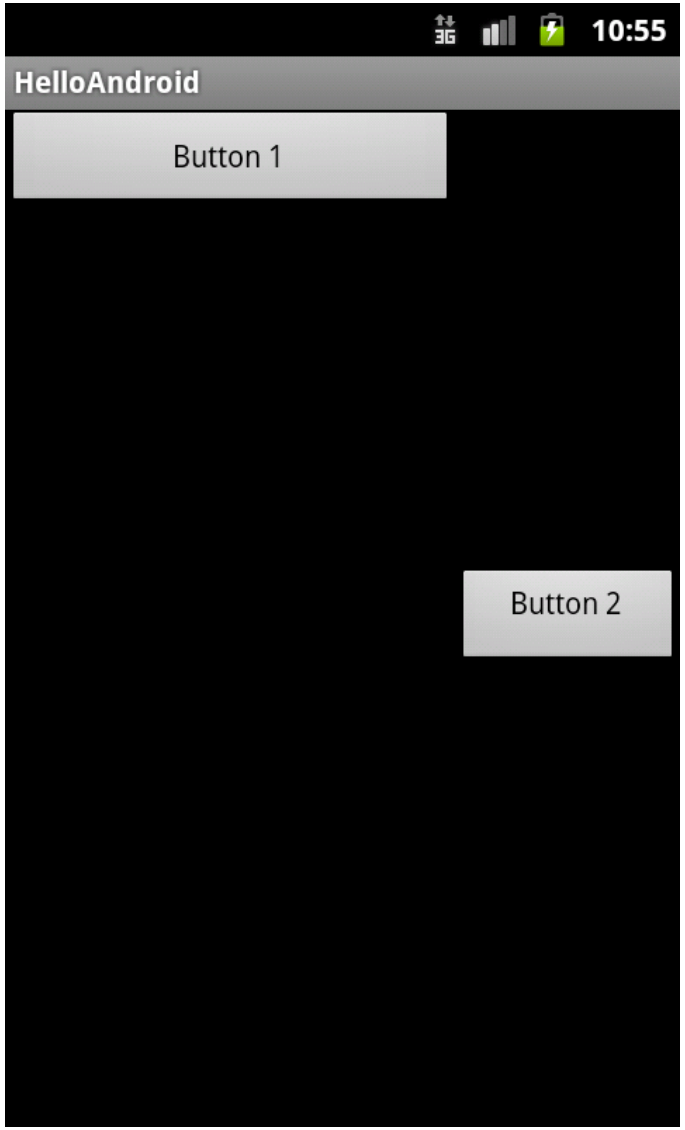




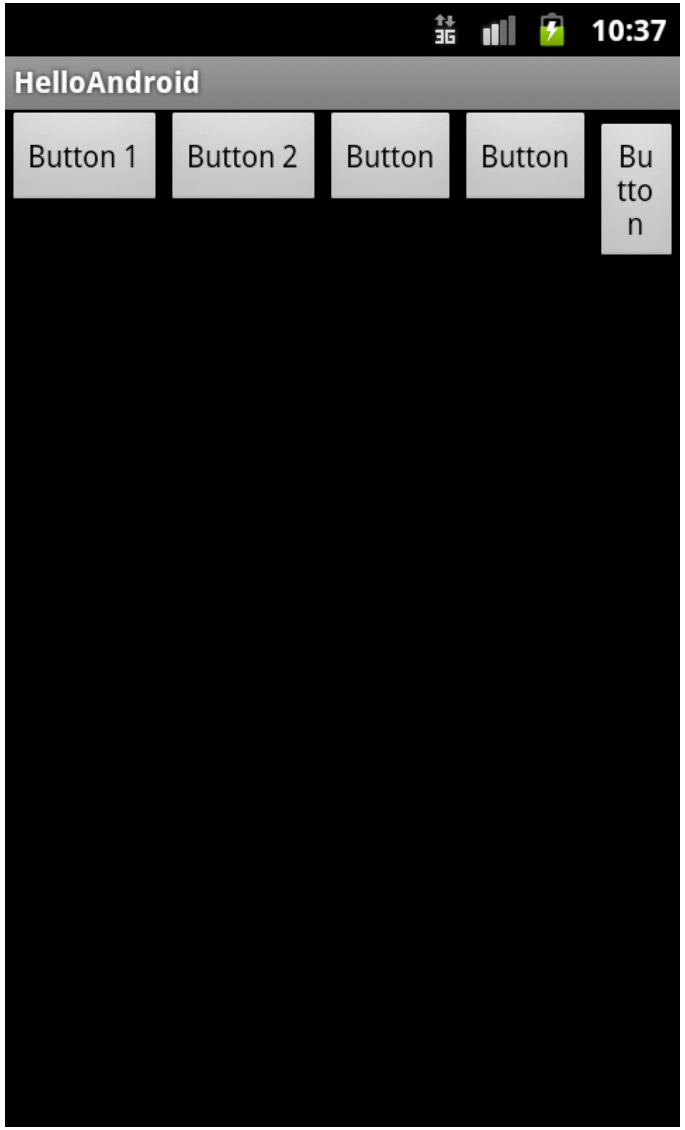
LinearLayout gravity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"    android:layout_height="fill_parent"    android:orientation="horizontal" >

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"        android:layout_height="wrap_content"
        android:text="@string/buttonString1"
        android:layout_weight="1" />
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"        android:layout_height="wrap_content"
        android:text="@string/buttonString2"
        android:layout_weight="2"
        android:layout_gravity="center_vertical"
        android:gravity="top|center" />

</LinearLayout>
```

LinearLayout gravity



LinearLayout problem



RelativeLayout

- Disposes views according to the container or according to other views
- The gravity attribute indicates what views are more important to define the layout
- Useful to align views

RelativeLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"    android:layout_height="match_parent" >

    <EditText
        android:id="@+id/username"        android:text="username"
        android:inputType="text"
        android:layout_width="wrap_content"        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_toRightOf="@+id/usernameLabel" >
    </EditText>

    <TextView
        android:id="@+id/usernameLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/username"
        android:text="Username" />
```

RelativeLayout

```
<EditText
    android:id="@+id/password"        android:text="password"
    android:inputType="textPassword"
    android:layout_below="@+id/username"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/username"
    android:layout_alignParentRight="true"
    android:layout_toRightOf="@+id/usernameLabel" >
  </EditText>

  <TextView
    android:id="@+id/passwordLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/password"
    android:text="Password" />
</RelativeLayout>
```

RelativeLayout

TableLayout

- As the name say, similar to a Table
- Has some attributes to customize the layout:
  - android:layout_column
  - android:layout_span
  - android:stretchColumns
  - android:shrinkColumns
  - android:collapseColumns
- Each row is inside a <TableRow> element

TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout android:layout_width="fill_parent"
    android:layout_height="fill_parent" xmlns:android="http://schemas.android.com/apk/res/android"    android:id="@+id/tableLayout">

  <TableRow android:layout_width="wrap_content" android:layout_height="wrap_content“ android:id="@+id/firstRow">
    <Button      android:id="@+id/button1“
                 android:layout_width="wrap_content“
                 android:layout_height="wrap_content“
          android:text="Button" />
    <Button android:id="@+id/button2“
          android:layout_width="match_parent"
                 android:layout_height="match_parent"
                 android:text="Button" />
    <Button android:id="@+id/button3"
          android:layout_width="match_parent"
                 android:layout_height="match_parent"
                 android:text="Button" />
  </TableRow>
```

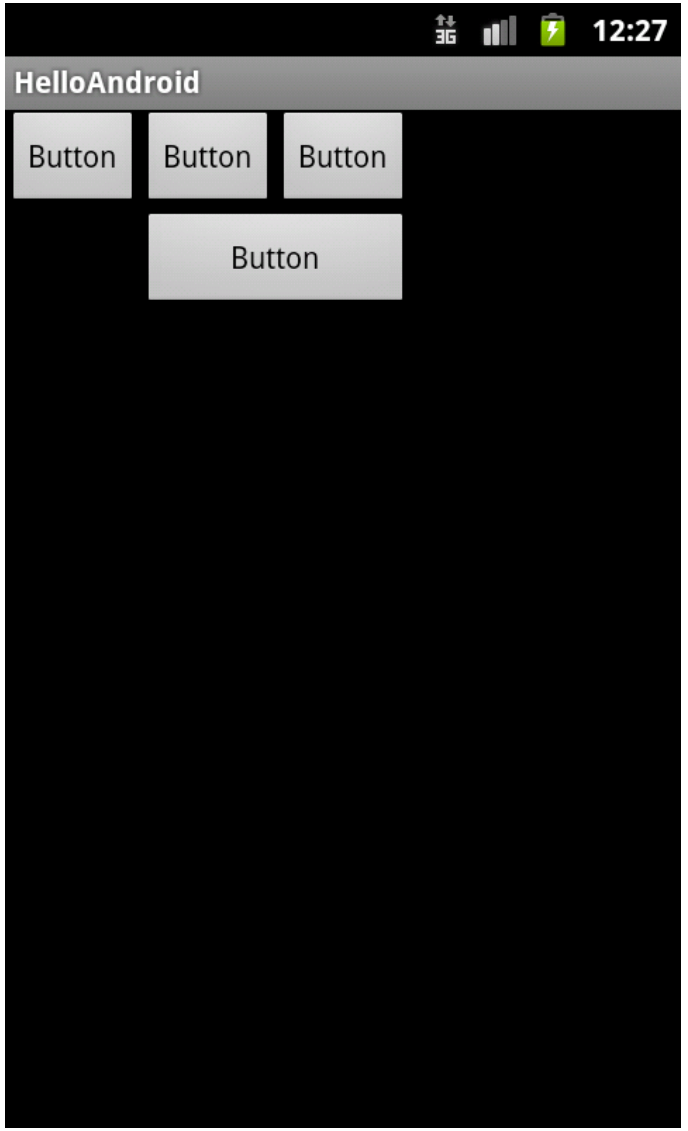
TableLayout

```
<TableRow
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/secondRow">

    <Button  android:layout_column="1"
        android:layout_span="2"
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button">
    </Button>
</TableRow>

</TableLayout>
```

TableLayout

```
<TableLayout ...>
   <TableRow..>
   <!-- Column 1 : Rowspan 2 -->
      <TextView .../>
   <!-- Column 2 : 2 Rows -->
      <TableLayout ...>
         <TableRow..>
            <TextView .../>
         </TableRow>
         <TableRow..>
            <TextView .../>
         </TableRow>
      </TableLayout>
   </TableRow>
</TableLayout>
```

TableLayout



FrameLayout and AbsoluteLayout

- FrameLayout
  - Adds an attribute, android:visibility
  - Makes the user able to define layouts managing the visibility of views
- AbsoluteLayout
  - Deprecated
  - Specify position with x and y
  - Pay attention to different resolutions