Relative Layout

After linear layouts, which display controls in a single row or column, relative layouts are one of the more common types of layouts used by Android user interface designers.

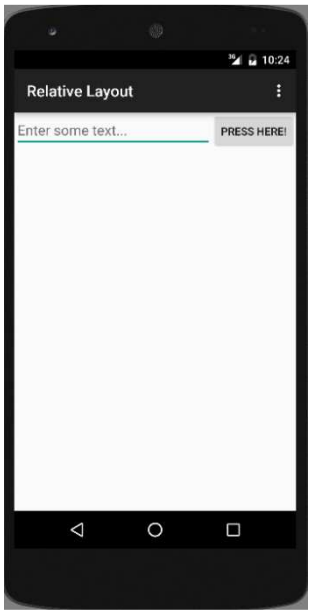
The relative layout works much as its name implies: it organizes controls relative to one another, or to the parent control itself. What does this mean? It means that child controls, such as ImageView, TextView ,and Button controls, can be placed above, below, to the left or right, of one another. Child controls can also be placed in relation to the parent (the relative layout container), including placement of controls aligned to the top, bottom, left or right edges of the layout.

Relative layout child control placement is defined using rules. These rules define how the controls within the relative layout are displayed. For the complete list of rules for relative layouts, see http://developer.android.com/reference/android/widget/RelativeLayout.html.

A Simple Relative Layout

Relative layouts are best explained using an example. Let's say we want to design a screen with an EditText control and a Button control. We want the Button to display to the right of the EditText control. Therefore, we could define a relative layout with two child controls: the EditText and the Button. The EditText control might have a rule that says: align this control to the left-hand side of the parent control (the layout) and to the left of a second control (the Button control). Meanwhile, the Button control might have a rule that says: align this control to the right-hand side of the parent control (the layout).

The following figures show just such a relative layout. The relative layout has two child controls: an EditText control and a Button control.



Defining an XML Layout Resource with a Relative Layout

The most convenient and maintainable way to design application user interfaces is by creating XML layout resources. This method greatly simplifies the UI design process, moving much of the static creation and layout of user interface controls and definition of control attributes, to the XML, instead of littering the code.

XML layout resources must be stored in the /res/layout project directory hierarchy. This layout resource file, aptly named /res/layout/relative.xml, is defined in XML as follows:

```
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_height="fill_parent"
   android:layout_width="fill_parent">
   <EditText
      android:id="@+id/EditText01"
      android:hint="Enter some text..."
      android:layout_alignParentLeft="true"
      android:layout_width="fill_parent"
      android:layout_toLeftOf="@+id/Button01"
      android:layout_height="wrap_content"></EditText>
   <Button
      android:id="@+id/Button01"
      android:text="Press Here!"
      android:layout_width="wrap_content"
      android:layout_alignParentRight="true"
      android:layout_height="wrap_content"></Button>
</RelativeLayout>
```

Exploring the Important Relative Layout Properties and Attributes

Now let's talk a bit about the attributes that help configure a relative layout and its child controls. Some specific attributes apply to relative layouts-namely the child rules, including:

- Rules for child control centering within the parent layout, including: center horizontally, center vertically, or both.
- Rules for child control alignment within the parent layout, including: align with top, bottom, left or right edge of another control.
- Rules for child control alignment in relation to other child controls, including: align with top, bottom, left or right edge.
- Rules for child control placement in relation to other child controls, including: placement to the left or right of a specific control, or above or below another control.

Also, general ViewGroup-style attributes apply to relative layouts. These include:

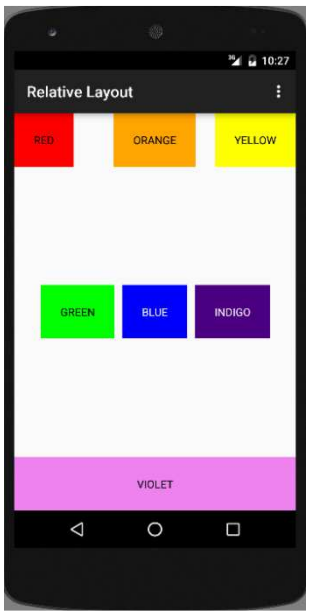
- Generic Layout Parameters such as layout_height (required) and layout_width (required) (class: ViewGroup.LayoutParams)
- Margin Layout Parameters such as margin_top, margin_left, margin_right and margin_bottom (class: ViewGroup. MarginLayoutParams)

- Layout Parameters such as layout_height and layout_width (class: ViewGroup.LayoutParams)

Working with Layout Rules

Let's look at a more complex screen design. For the purposes of this exercise, we will start by looking at the final screen design, and then will work backwards, discussing the relative layout features and rules used to achieve this final result.

Let's say we want to design a screen that looks like this:



First, define a relative layout in your XML resource file. Since you want this layout to control the contents of the entire screen, set its height and width attributes to fill_parent. Your XML resource file should now look something like this:

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

Next, we determine what child controls we need. In this case, we need seven TextView controls (one for each color). Configure them as you normally would, setting text attributes to strings, background colors, font sizes, etc. Place each of these controls within your relative layout.

Next, we define the rules for each child control, in order to get them to draw in the appropriate places:

- The RED TextView control has no specific settings configured. By default, this control will be drawn in the upper left-hand corner of the parent layout.

- The ORANGE TextView control is centered horizontally in the parent layout. Because all controls default to the top left-hand corner of the screen, this effectively anchors the control to the top middle edge of the parent layout.
- The YELLOW TextView control is aligned to the right-hand edge of the parent layout. Because all controls default to the top left-hand corner of the screen, this effectively anchors the control to the top right corner of the parent layout.
- The GREEN TextView control is centered vertically within the parent layout and configured to display to the left of the BLUE TextView control.
- The BLUE TextView control is aligned to the center (horizontally and vertically) of the parent control. This displays it in the middle of the screen.
- The INDIGO TextView control is centered vertically within the parent layout and configured to display to the right of the BLUE TextView control.
- The VIOLET TextView control is aligned to the bottom edge of the parent layout. It's width is also set to fill the parent, allowing it to stretch across the bottom edge of the screen.

If you define these rules in your XML resource file, it should now look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_height="fill_parent"
   android:layout_width="fill_parent">
   <TextView
      android:text="RED"
      android:id="@+id/TextView01"
      android:layout_height="wrap_content"
      android:background="#f00"
      android:gravity="center"
      android:textColor="#000"
      android:layout_width="wrap_content"
      android:padding="25dp"></TextView>
   <TextView
      android:text="ORANGE"
      android:layout_height="wrap_content"
      android:background="#ffa500"
      android:gravity="center"
      android:textColor="#000"
      android:id="@+id/TextView02"
      android:layout_width="wrap_content"
      android:layout_centerHorizontal="true"
      android:padding="25dp"></TextView>
   <TextView
      android:text="YELLOW"
      android:layout_height="wrap_content"
      android:background="#ffff00"
      android:gravity="center"
      android:textColor="#000"
      android:id="@+id/TextView03"
      android:layout_width="wrap_content"
      android:layout_alignParentRight="true"
      android:padding="25dp"></TextView>
   <TextView
      android:text="GREEN"
      android:layout_height="wrap_content"
      android:background="#0f0"
```

```
        android:gravity="center"
        android:textColor="#000"
        android:id="@+id/TextView04"
        android:layout_width="wrap_content"
        android:layout_toLeftOf="@+id/TextView05"
        android:padding="25dp"
        android:layout_centerVertical="true"></TextView>
    <TextView
        android:text="BLUE"
        android:layout_height="wrap_content"
        android:background="#00f"
        android:gravity="center"
        android:textColor="#fff"
        android:id="@+id/TextView05"
        android:layout_width="wrap_content"
        android:layout_centerInParent="true"
        android:layout_margin="10dp"
        android:padding="25dp"></TextView>
    <TextView
        android:text="INDIGO"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="#fff"
        android:id="@+id/TextView06"
        android:layout_width="wrap_content"
        android:layout_toRightOf="@+id/TextView05"
        android:background="#4b0082"
        android:padding="25dp"
        android:layout_centerVertical="true"></TextView>
    <TextView
        android:text="VIOLET"
        android:layout_height="wrap_content"
        android:background="#ee82ee"
        android:gravity="center"
        android:textColor="#000"
        android:id="@+id/TextView07"
        android:layout_alignParentBottom="true"
        android:layout_width="fill_parent"
        android:padding="25dp"></TextView>
</RelativeLayout>
```