Linear Layout

LinearLayout is a ViewGroup that displays child View elements in a linear direction, either vertically or horizontally.

You should be careful about over-using the LinearLayout. If you begin nesting multiple LinearLayouts, you may want to consider using a RelativeLayout instead.

1. Start a new project named *HelloLinearLayout*.
2. Open the `Resources/Layout/Main.axml` file and insert the following:

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

  <LinearLayout
      android:orientation=    "horizontal"
      android:layout_width=    "fill_parent"
      android:layout_height=    "fill_parent"
      android:layout_weight=    "1"    >
      <TextView
          android:text=    "red"
          android:gravity=    "center_horizontal"
          android:background=    "#aa0000"
          android:layout_width=    "wrap_content"
          android:layout_height=    "fill_parent"
          android:layout_weight=    "1"    />
      <TextView
          android:text=    "green"
```

```
        android:gravity=    "center_horizontal"
        android:background=    "#00aa00"
        android:layout_width=    "wrap_content"
        android:layout_height=    "fill_parent"
        android:layout_weight=    "1"    />
    <TextView
        android:text=    "blue"
        android:gravity=    "center_horizontal"
        android:background=    "#0000aa"
        android:layout_width=    "wrap_content"
        android:layout_height=    "fill_parent"
        android:layout_weight=    "1"    />
    <TextView
        android:text=    "yellow"
        android:gravity=    "center_horizontal"
        android:background=    "#aaaa00"
        android:layout_width=    "wrap_content"
        android:layout_height=    "fill_parent"
        android:layout_weight=    "1"    />
</LinearLayout>

<LinearLayout
  android:orientation=    "vertical"
  android:layout_width=    "fill_parent"
  android:layout_height=    "fill_parent"
  android:layout_weight=    "1"    >
  <TextView
      android:text=    "row one"
      android:textSize=    "15pt"
      android:layout_width=    "fill_parent"
      android:layout_height=    "wrap_content"
      android:layout_weight=    "1"    />
    <TextView
        android:text=    "row two"
        android:textSize=    "15pt"
        android:layout_width=    "fill_parent"
        android:layout_height=    "wrap_content"
        android:layout_weight=    "1"    />
    <TextView
        android:text=    "row three"
        android:textSize=    "15pt"
        android:layout_width=    "fill_parent"
        android:layout_height=    "wrap_content"
        android:layout_weight=    "1"    />
    <TextView
        android:text=    "row four"
        android:textSize=    "15pt"
        android:layout_width=    "fill_parent"
        android:layout_height=    "wrap_content"
       android:layout_weight=    "1"    />
  </LinearLayout>

</LinearLayout>
```

Carefully inspect this XML. There is a root `LinearLayout` that defines its orientation to be vertical—all child `View`s (of which it has two) will be stacked vertically. The first child is another `LinearLayout` that uses a horizontal orientation and the second child is a `LinearLayout` that uses a vertical orientation. Each of these nested `LinearLayout`s contain several `TextView` elements, which are oriented with each other in the manner defined by their parent `LinearLayout`.

3. Now open `HelloLinearLayout.cs` and be sure it loads the `Resources/Layout/Main.axml` layout in the `OnCreate()` method:

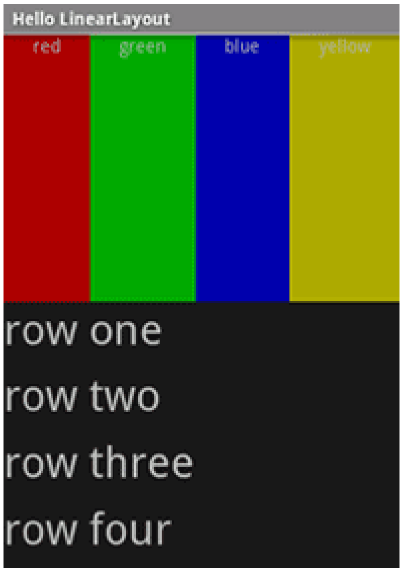
```
protected override void OnCreate (Bundle savedInstanceState)
```

```
{
    base.OnCreate (savedInstanceState);
    SetContentView (Resource.Layout.Main);
}
```

The `SetContentView(int)` method loads the layout file for the `Activity`, specified by the resource ID — `Resources.Layout.Main` refers to the `Resources/Layout/Main.axml` layout file.

4. Run the application.

You should see the following:



Notice how the XML attributes define each View's behavior. Try experimenting with different values for `android:layout_weight` to see how the screen real estate is distributed based on the weight of each element. See the Common Layout Objects document for more about how `LinearLayout` handles the `android:layout_weight` attribute.

References

- `LinearLayout`
- `TextView`