Trường Đại học Bách Khoa Hà Nội
Hanoi University of Science and Technology

Chapter 4. Graphical User Interfaces

Hanoi University of Science and Technology

Nguyen Hong Quang
2/9/2012

Content

- 4.1a. Layout
  - LinearLayout
  - RelativeLayout
  - TableLayout
  - ScrollView
- 4.1b. UI Events Handling
  - Listener Introduction
  - onClickListener
  - onKeyListener, onFocusChangeListener, onLongClickListener, onTouchListener

4.1. Layout

- Most GUI toolkits have some notion of layout management, frequently organized into containers.
- Android's Layouts:
  - LinearLayout : the box model
  - RelativeLayout : a rule-based model
  - TableLayout : the grid model

4.1.1. LinearLayout

- LinearLayout is a box model
  - widgets or child containers are lined up in a column or row, one after the next.
- Configure a LinearLayout:
  - Orientation
  - fill model
  - weight
  - gravity
  - padding

4.1.1. LinearLayout Orientation

- indicates whether the LinearLayout represents a row or a column.
- android:orientation
  - horizontal
  - vertical

4.1.1. LinearLayout Fill Model

- android:layout_width and android:layout_height properties
- match_parent/wrap_content/125dip

4.1.1. LinearLayout Gravity

- By default, everything in a LinearLayout is left- and top-aligned.
- Common gravity values are :
  - left, center_horizontal, and right for left-aligned, centered, and right-aligned widgets respectively.

4.1.1. Margins

- set in XML, either on a per-side basis (e.g., android:layout_marginTop) or on all sides via android:layout_margin.

4.1.1. LinearLayout Weight

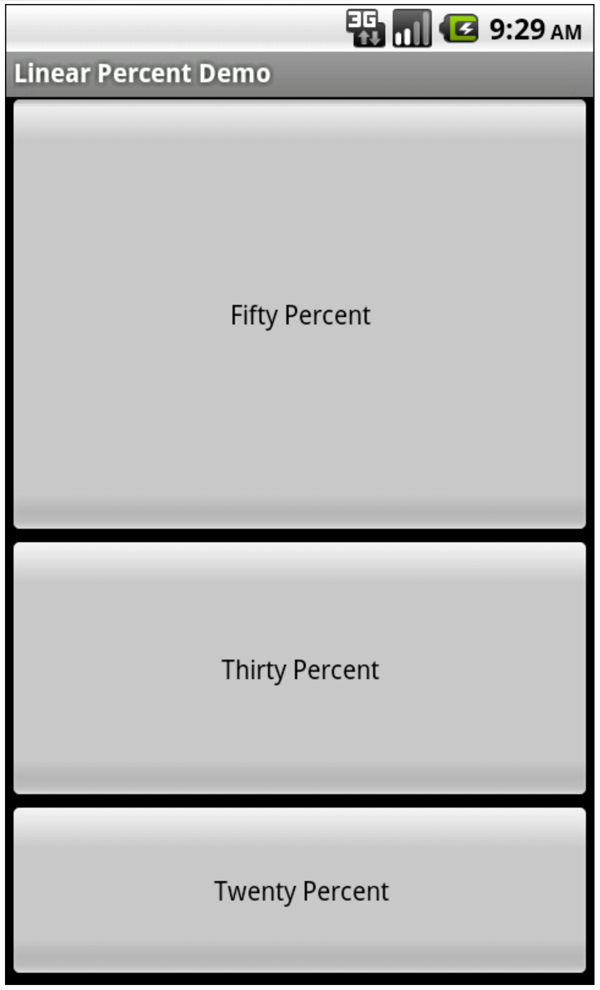
- For example, having two multi-line fields in a column
- First methods:
  - setting android:layout_width to match_parent, android:layout_weight to be the same non-zero value for a pair of widgets
  - If setting 1 for one widget and 2 for another widget, the second widget will use up twice the free space that the first widget does.

4.1.1. LinearLayout Weight

- Second methods: allocate sizes on a percentage basis
  - Set all the android:layout_width values to be 0 for the widgets in the layout
  - Set the android:layout_weight values to be the desired percentage size for each widget in the layout
  - Make sure all those weights add up to 100

4.1.1. LinearLayout Examples

```
<LinearLayout>
    <Button
      android:layout_height="0dip"
      android:layout_weight="50“>
    <Button
      android:layout_height="0dip"
      android:layout_weight="30“>
    <Button
      android:layout_height="0dip"
      android:layout_weight="20” />
</LinearLayout>
```



4.1. Layout
4.1.2. RelativeLayout

- Lays out widgets based upon their relationship to other widgets in the container and the parent container
- You can place Widget X below and to the left of Widget Y , or have Widget Z's bottom edge align with the bottom of the container, and so on.

4.1.2. RelativeLayout Positions Relative to Container

- android:layout_alignParentTop
- android:layout_alignParentBottom
- android:layout_alignParentLeft
- android:layout_alignParentRight
- android:layout_centerHorizontal
- android:layout_centerVertical
- android:layout_centerInParent

- Values: true/false

4.1.2. RelativeLayout Relative Notation in Properties

- Control position of a widget vis a vis other widgets:
  - android:layout_above
  - android:layout_below
  - android:layout_toLeftOf
  - android:layout_toRightOf

4.1.2. RelativeLayout Relative Notation in Properties

- Control one widget's alignment relative to another:
  - android:layout_alignTop
  - android:layout_alignBottom
  - android:layout_alignLeft
  - android:layout_alignRight
  - android:layout_alignBaseline
    - indicates that the baselines of the two widgets should be aligned (where the "baseline" is that invisible line that text appears to sit on)

4.1.2. RelativeLayout Identity of a widget

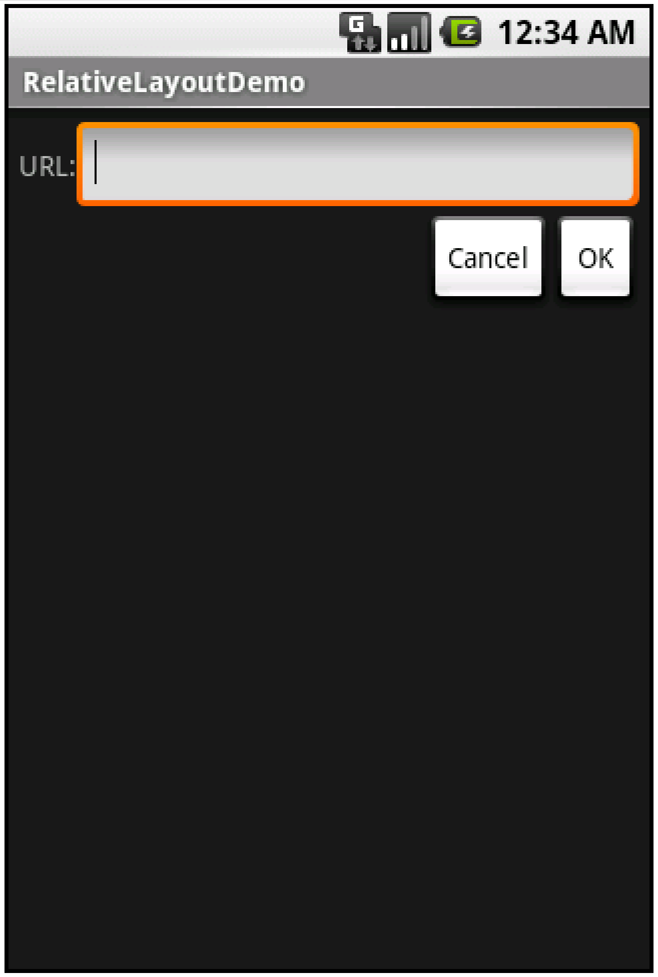
- Properties of relevance to RelativeLayout take as a value the identity of a widget in the container.
- To do this:
  - 1. Put identifiers (android:id attributes) on all elements that you will need to address
  - 2. Reference other widgets using the same identifier value

4.1.2. RelativeLayout Order of Evaluation

- Android uses two passes to process the rules, so you can now safely have forward references to as-yet-undefined widgets.

4.1.2. RelativeLayout Examples

```
<RelativeLayout>
    <TextView
        android:layout_alignBaseline=
                "@+id/entry”
        android:layout_alignParentLeft="true“/>
    <EditText
        android:layout_toRightOf="@id/label"
        android:layout_alignParentTop="true"/>
    <Button
        android:layout_below="@id/entry"
        android:layout_alignRight="@id/entry“/>
    <Button
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok”/>
</RelativeLayout>
```

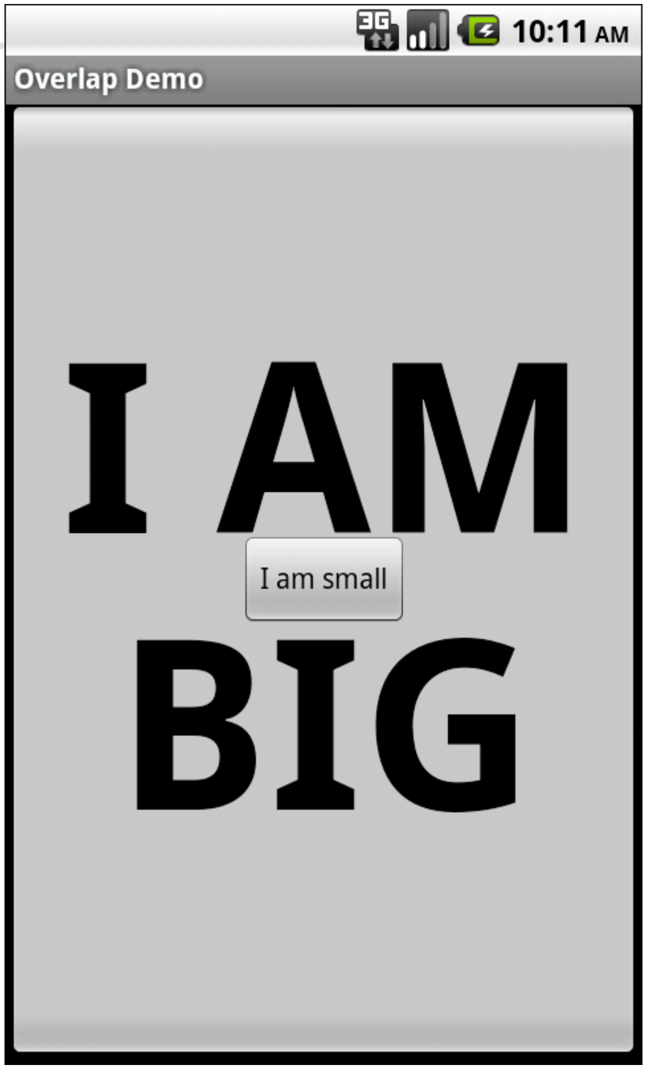


4.1.2. RelativeLayout Overlap

- RelativeLayout also has a feature that LinearLayout lacks – the ability to have widgets overlap one another.
- Later children of a RelativeLayout are "higher in the Z axis" than are earlier children, meaning that later children will overlap earlier children if they are set up to occupy the same space in the layout.

4.1.2. RelativeLayout Examples

```
<RelativeLayout>
    <Button
        android:textSize="120dip"
        android:textStyle="bold"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
    />
</RelativeLayout>
```



4.1.3. TableLayout

- Position widgets in a grid which columns might shrink or stretch to accommodate their contents, and so on.
- Rows are declared by the developer, by putting widgets as children of a TableRow inside the overall TableLayout.
- The number of columns are determined by Android

4.1.3. TableLayout Number of columns

- First, there will be at least one column per widget in your longest row.
- So if we have three rows, one with two widgets, one with three widgets, and one with four widgets, there will be at least four columns.

4.1.3. TableLayout

- A widget can take up more than one column by including the android:layout_span property, indicating the number of columns the widget spans.
- Put a widget into a different column via the android:layout_column property (columns are counted starting from 0)

Non-Row Children of TableLayout

- It is possible to put other widgets in between rows.
- For those widgets, TableLayout behaves a bit like LinearLayout with vertical orientation.
- The widgets automatically have their width set to fill_parent, so they will fill the same space that the longest row does.

4.1.3. TableLayout Stretch, Shrink, and Collapse

- android:stretchColumns
- android:shrinkColumns
- android:collapseColumns

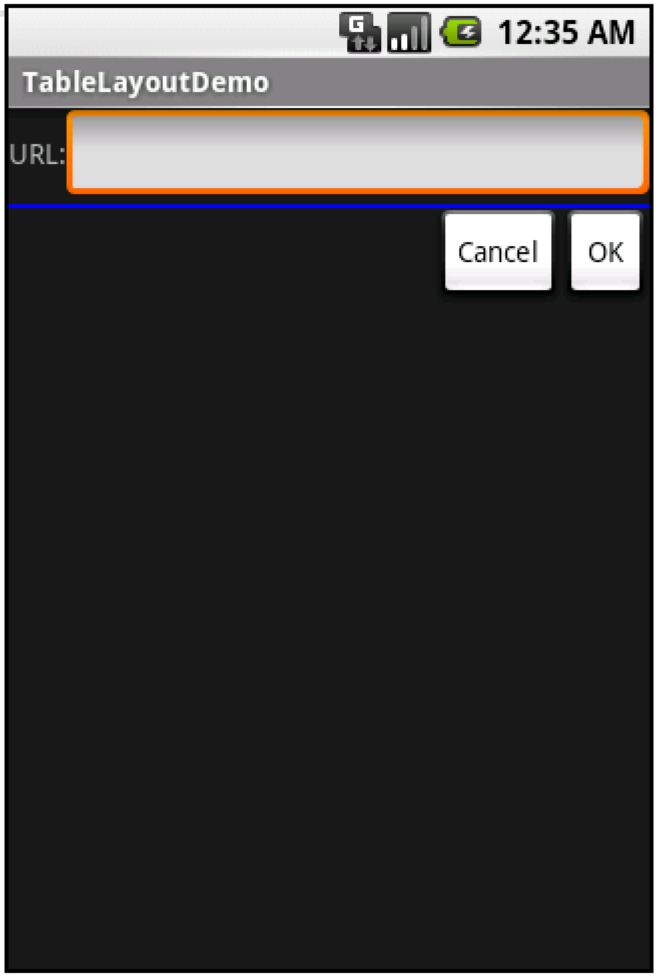
- Values:
  - this should be a single column number or a commadelimited list of column numbers

4.1.2. RelativeLayout Examples

```
<RelativeLayout>
    <Button
        android:textSize="120dip"
        android:textStyle="bold"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
    />
</RelativeLayout>
```

4.1.2. RelativeLayout Examples

```
<TableLayout>
     <TableRow>
          <TextView/>
          <EditText
                    android:layout_span="3"/>
    </TableRow>
    <TableRow>
          <Button
                    android:layout_column="2" />
          <Button/>
    </TableRow>
</TableLayout>
```



4.1.3. ScrollView

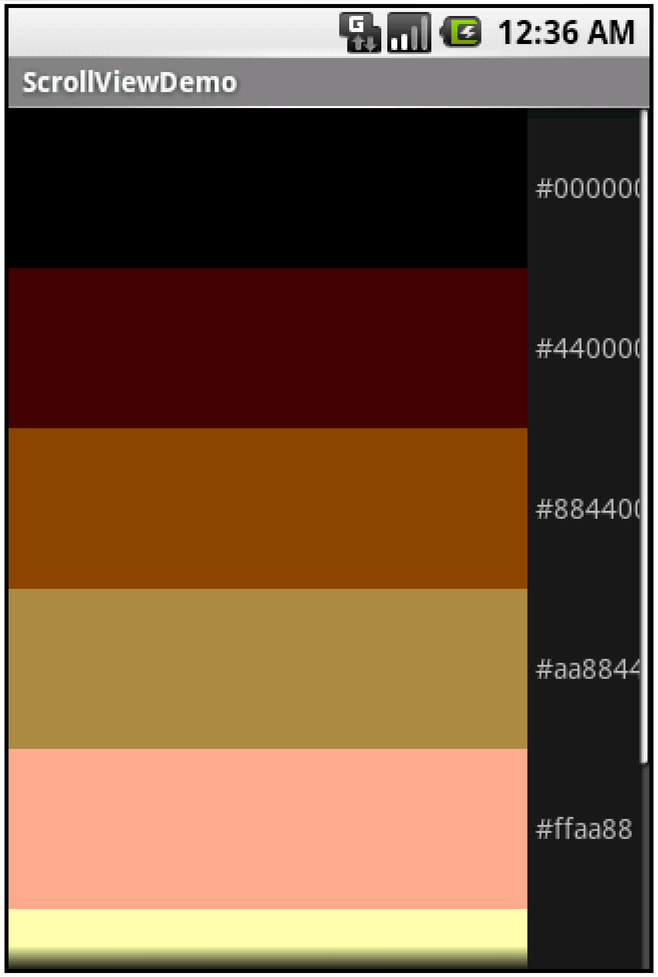
- Only part of the information is visible at one time, the rest available via scrolling up or down.

4.1.2. RelativeLayout Examples

```
<TableLayout>
    <TableRow>
        <TextView/>
        <EditText
            android:layout_span="3"/>
    </TableRow>
    <TableRow>
        <Button
            android:layout_column="2" />
        <Button/>
    </TableRow>
</TableLayout>
```

4.1.2. RelativeLayout Examples

```
<ScrollView>
<TableLayout>
   android:stretchColumns="0"
    <TableRow>
      <View
        android:layout_height="80dip"
        android:background="#000000"/>
      <TextView android:text="#000000"
           android:paddingLeft="4dip"
         android:layout_gravity="center_vertical"
      />
    </TableRow> ....
    </TableLayout>
</ ScrollView>
```



End of Lecture

Q&A