Android Programming
Lecture 4

9/14/2011

Layouts

- Layouts are ViewGroups which are used to hold other Views
- Invisible
- Allow *positioning* of different elements
- Layouts can be nested inside of each other

- Common layouts:
  - FrameLayout
  - LinearLayout
  - TableLayout
  - RelativeLayout
  - Gallery

```
import android.widget.NAME;
```

http://developer.android.com/guide/topics/ui/layout-objects.html

Specifying Layouts in XML

- It is very common to specify layouts in a text instead of code format
- For main activity, layout specified in *res/layout/main.xml*

- **XML: Extensible Markup Language**
  - Similar to HTML
  - Markup tags (< >, opening, /closing), Attributes=Values (x=y), Content (text [rare actually])
  - Nesting

Specifying Layouts in XML

```java
public class LayoutExamplesActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Button buttonOne = new Button(this);
        Button buttonTwo = new Button(this);
        buttonOne.setText("Press Me!");
        buttonTwo.setText("Press Me Two!");

        LinearLayout linearLayout = new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.VERTICAL);
        linearLayout.addView(buttonOne);
        linearLayout.addView(buttonTwo);

        setContentView(linearLayout);
    }
}
```

Code and XML approaches
that generate the same interface

In XML version, the two Buttons are
Nested inside the LinearLayout
(between <LinearLayout></LinearLayout>)

```xml
<?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"
    >
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Press Me!"
    >
    </Button>
    <Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Press Me Two!"
    >
    </Button>
</LinearLayout>
```

StopWatch: XML Layout

```
<?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"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="9"
    android:text=""
    android:id="@+id/textview1"
    ></TextView>
 <Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Record Time"
    android:id="@+id/button1"
    ></Button>
</LinearLayout>
```

Layout width, height, and weight parameters set as previously written in code

Other initial features of the Views can also be set, such as the initial text

Ids are auto-generated and can be referenced in code

@+id/button1 means
- Go to the ID resource
- Add a *new* ID value for Button1

Tricky: 1) ids don't map to 0,1,2,3
2) weights in XML are integer strings, in code floats

Using the XML Spec in Code

- The XML is compiled into resources
  - The view resource created can auto-magically be "inflated" at runtime
  - Can programmatically access other definitions, such as IDs, made in XML

- To make use of XML specified layout in code:
  - Use the layout as your layout for the Activity:
    - Reference the layout Resource specified in the main.xml file

```
setContentView(R.layout.main);
```

  - Access parts of layout by id:
    - Reference the id Resource, get View associated with that id

```
button = (Button)findViewById(R.id.button1);
```

Using the XML Spec in Code

```
public class StopWatchXMLActivity extends Activity implements View.OnClickListener {

    Button button;
    TextView textView;
    Time currentTime;
    LinearLayout linearLayout;
    int count;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        button = (Button)findViewById(R.id.button1);
        textView = (TextView)findViewById(R.id.textview1);

        button.setOnClickListener(this);

        currentTime = new Time();
        count =1;

    }

    public void onClick(View arg0) {
        // TODO Auto-generated method stub
        if (arg0.getId() == R.id.button1)
        {
            currentTime.setToNow();
            if (count == 1)
                textView.setText(""+currentTime);
            else
                textView.setText(textView.getText() + "\n" + currentTime);
            count = count + 1;
        }
    }
}
```

Why Use XML?

- Separation of appearance (presentation) from actual meaningful state-changing code
- Can change interface without having to completely recompile Java code
  - Can just recompile XML
  - View Resource is inflated at runtime

Speaking of XML: Applications and Activities

- How does the Application know the initial Activity to call?
  - Stored in application manifest: AndroidManifest.xml
    - Managed by Eclipse for us

Indication that the activity is the first target

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="turkett.csc191"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".FirstActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

Speaking of XML: Applications and Activities

- A manifest for an Application with two Activity components

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="turkett.android.ridethewake"
      android:versionCode="3"
      android:versionName="1.2">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".StartActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
         android:name=".SettingsActivity"
         android:label="@string/settings_name" />
    <uses-library android:name="com.google.android.maps" android:required="true"></uses-library>
</application>
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="8" />
</manifest>
```

RelativeLayout

- A Layout where the location for Views that are added can be described:
  - Relative to other Views added (“to the left of X”)
  - Relative to the RelativeLayout container (“aligned to the container bottom”)
- Very flexible
- Suggested for use over nested LinearLayouts
  - More complex the nesting of a layout, the longer to inflate

RelativeLayout: Layout Parameters

android:layout_above		Positions the bottom edge of this view above the given anchor view ID.
android:layout_alignBaseline		Positions the baseline of this view on the baseline of the given anchor view ID.
android:layout_alignBottom		Makes the bottom edge of this view match the bottom edge of the given anchor view ID.
android:layout_alignLeft		Makes the left edge of this view match the left edge of the given anchor view ID.
android:layout_alignParentBottom		If true, makes the bottom edge of this view match the bottom edge of the parent.
android:layout_alignParentLeft		If true, makes the left edge of this view match the left edge of the parent.
android:layout_alignParentRight		If true, makes the right edge of this view match the right edge of the parent.
android:layout_alignParentTop		If true, makes the top edge of this view match the top edge of the parent.
android:layout_alignRight		Makes the right edge of this view match the right edge of the given anchor view ID.
android:layout_alignTop		Makes the top edge of this view match the top edge of the given anchor view ID.
android:layout_alignWithParentIfMissing		If set to true, the parent will be used as the anchor when the anchor cannot be be found for layout_toLeftOf, layout_toRightOf, etc.
android:layout_below		Positions the top edge of this view below the given anchor view ID.
android:layout_centerHorizontal		If true, centers this child horizontally within its parent.
android:layout_centerInParent		If true, centers this child horizontally and vertically within its parent.
android:layout_centerVertical		If true, centers this child vertically within its parent.
android:layout_toLeftOf		Positions the right edge of this view to the left of the given anchor view ID.
android:layout_toRightOf		Positions the left edge of this view to the right of the given anchor view ID.

Remember, two styles:

Relative to some other view (layout_below)

Relative to RelativeLayout container parent (alignParentLeft)

ClearingStopWatch: LinearLayout (Homework Review)

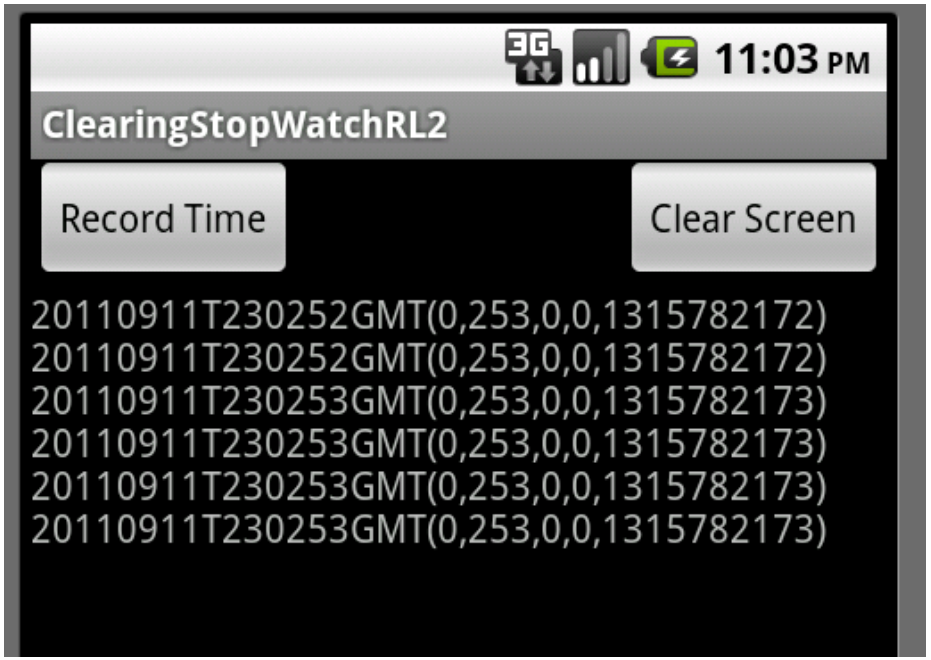
```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.main);

    button = new Button(this);
    clearButton = new Button(this);
    textView = new TextView(this);

    linearLayout = new LinearLayout(this);
    linearLayout.setOrientation(LinearLayout.VERTICAL);
    LinearLayout.LayoutParams buttonParams =
            new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,LinearLayout.LayoutParams.WRAP_CONTENT, 0.1f);
    linearLayout.addView(button, buttonParams);
    linearLayout.addView(clearButton, buttonParams);
    LinearLayout.LayoutParams textParams =
            new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,LinearLayout.LayoutParams.FILL_PARENT, 0.9f);
    linearLayout.addView(textView, textParams);


    button.setText("Record Time");
    button.setId(1);
    button.setOnClickListener(this);
    clearButton.setText("Clear Screen");
    clearButton.setId(2);
    clearButton.setOnClickListener(this);

    currentTime = new Time();
    count = 1;
    setContentView(linearLayout);
}
```

ClearingStopWatch: RelativeLayout (Homework Review)

```
button = new Button(this);
button.setId(1);
button.setText("Record Time");
clearButton = new Button(this);
clearButton.setId(2);
clearButton.setText("Clear Screen");
textView = new TextView(this);
textView.setText("Hello?!?");
textView.setId(3);

relativeLayout = new RelativeLayout(this);
RelativeLayout.LayoutParams buttonParams =
        new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.FILL_PARENT,RelativeLayout.LayoutParams.WRAP_CONTENT);
buttonParams.addRule(RelativeLayout.ALIGN_PARENT_TOP);

relativeLayout.addView(button, buttonParams);
RelativeLayout.LayoutParams clearButtonParams =
        new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.FILL_PARENT,RelativeLayout.LayoutParams.WRAP_CONTENT);
clearButtonParams.addRule(RelativeLayout.BELOW,1);
relativeLayout.addView(clearButton, clearButtonParams);

RelativeLayout.LayoutParams textParams =
        new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.FILL_PARENT,RelativeLayout.LayoutParams.FILL_PARENT);
textParams.addRule(RelativeLayout.BELOW,2);
relativeLayout.addView(textView, textParams);
```

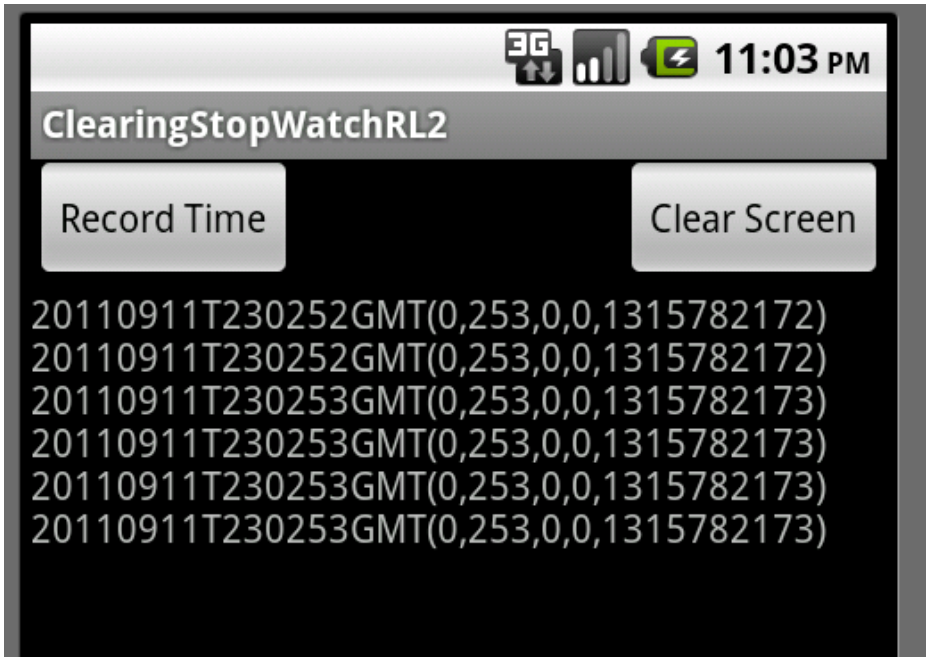
ClearingStopWatch: RelativeLayout Exploiting Relative Controls

- Why is this layout difficult with a linear layout?



ClearingStopWatch: RelativeLayout Exploiting Relative Controls

- Why is this layout difficult with a linear layout?
- Would need to nest two layouts
  - Row of Buttons
  - TextView
- If row of Buttons uses horizontal LinearLayout, would be next to each other instead of pushed to borders of screen



ClearingStopWatch: RelativeLayout Exploiting Relative Controls

```
 button = new Button(this);
 button.setId(1);
 button.setText("Record Time");
 clearButton = new Button(this);
 clearButton.setId(2);
 clearButton.setText("Clear Screen");
 textView = new TextView(this);
 textView.setText("Hello?!?");
 textView.setId(3);

relativeLayout = new RelativeLayout(this);
RelativeLayout.LayoutParams buttonParams =
        new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,RelativeLayout.LayoutParams.WRAP_CONTENT);
buttonParams.addRule(RelativeLayout.ALIGN_PARENT_TOP);
buttonParams.addRule(RelativeLayout.ALIGN_PARENT_LEFT);

relativeLayout.addView(button, buttonParams);
RelativeLayout.LayoutParams clearButtonParams =
        new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,RelativeLayout.LayoutParams.WRAP_CONTENT);
clearButtonParams.addRule(RelativeLayout.ALIGN_PARENT_TOP);
clearButtonParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
relativeLayout.addView(clearButton, clearButtonParams);

RelativeLayout.LayoutParams textParams =
        new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.FILL_PARENT,RelativeLayout.LayoutParams.FILL_PARENT);
textParams.addRule(RelativeLayout.BELOW,2);
relativeLayout.addView(textView, textParams);
```

ClearingStopWatch: RelativeLayout XML

ClearingStopWatchRLXMLActivity.java | main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:text="Record Time"
    >
</Button>
<Button
    android:id="@+id/clearButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
    android:text="Clear Screen"
    />
<TextView
    android:id="@+id/textview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text=""
    android:layout_below="@id/clearButton"
    >
</TextView>
</RelativeLayout>
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);
    button = (Button)findViewById(R.id.button);
    clearButton=(Button)findViewById(R.id.clearButton);
    textView = (TextView)findViewById(R.id.textview);

    button.setOnClickListener(this);
    clearButton.setOnClickListener(this);

    currentTime = new Time();
    count = 1;

}

@Override
public void onClick(View arg0) {
    // TODO Auto-generated method stub
    if (arg0.getId() == button.getId())
    {
        currentTime.setToNow();
        if (count == 1)
            textView.setText(""+currentTime);
        else
            textView.setText(textView.getText() + "\n" + currentTime);
        count = count + 1;
    }
    else if (arg0.getId() == clearButton.getId())
    {
        textView.setText("");
        count = 1;
    }
}
```