Lesson 6 – User Interfaces Part 7

In this session, we're going to review some additional layouts and widgets. Specifically, we'll take a quick look at the TableLayout and RelativeLayout Layouts, as well as the RatingBar, and SeekBar.

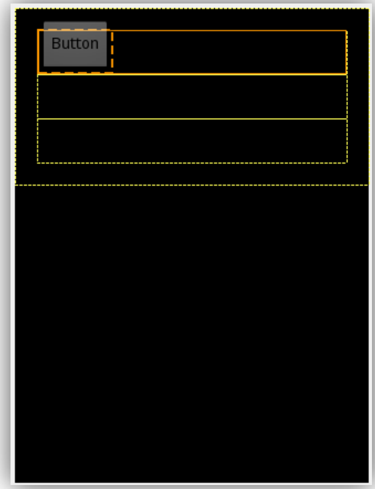
Open the existing Lesson5_UI project, and create a new Layout file (see Part 2, if you're not sure how) called **other_layouts_example.xml**. Start it off with a standard LinearLayout layout. Make sure that the LinearLayout fills the parent (match_parent or fill_parent for height / width), and make sure that its orientation is set to vertical.

Using the visual UI editor in Eclipse, drag and drop the TableLayout onto the LinearLayout. Depending on your version of Eclipse and Android SDK, the layout may come in with four TableRows, or it may simply appear as a TableLayout without any rows. Switch to the XML view and review the output. Add TableRows to your TableLayout if necessary. You'll need a total of three TableRow elements within your TableLayout element. In the end, your XML should look like this:

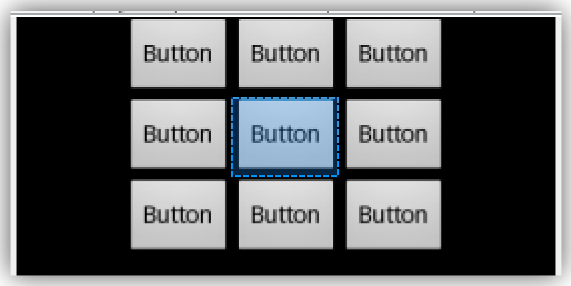
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent" android:layout_height="match_parent"
        android:orientation="vertical">
        <TableLayout android:id="@+id/tableLayout1"
                android:layout_height="wrap_content" android:layout_width="match_parent">
                <TableRow android:id="@+id/tableRow1" android:layout_width="wrap_content"
                        android:layout_height="wrap_content"></TableRow>
                <TableRow android:id="@+id/tableRow2" android:layout_width="wrap_content"
                        android:layout_height="wrap_content"></TableRow>
                <TableRow android:id="@+id/tableRow3" android:layout_width="wrap_content"
                        android:layout_height="wrap_content"></TableRow>
        </TableLayout>

</LinearLayout>
```

Now, drag and drop three buttons into each row.



After adding the buttons, use the Gravity property on each TableRow to center the three buttons horizontally in the each row.



TableLayout is much like an Excel grid. Resizing of any cell within a column will extend the content across all columns. So, for example, if we set the width of the center button to be 150dp, buttons in the top center and bottom center will also expand. Give that a try. Highlight the center button, and edit its width property, setting it to 150dp. Note the results.

Add a LinearLayout to your UI, placing it below the TableLayout (not inside any of the rows). Make sure that its orientation is set to vertical, and that its layout_height is set to wrap_content, and layout_width to match _parent (or fill_parent). Set the padding property for this layout to 10dp, and its layout_gravity to center_horizontal.

To that LinearLayout, add a SeekBar widget, followed by a RatingBar widget.

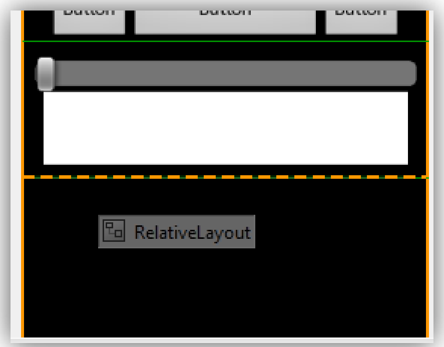
Customize the SeekBar, by setting the following properties:

- Max: 50

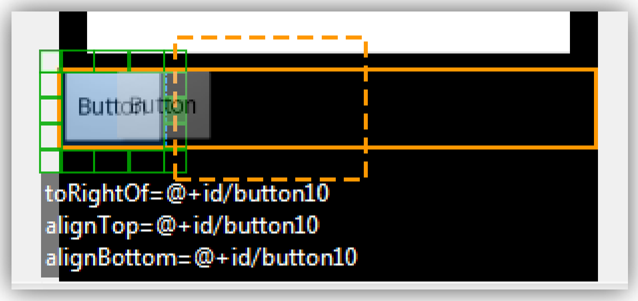
And the RatingBar:

- Max: 5
- Num Stars: 5
- Step Size: 1

Below the LinearLayout housing our new widgets, add a RelativeLayout. A RelativeLayout allows us to position widgets in relation to other widgets in our UI.



Add a button to the RelativeLayout. Try to add a second button to the layout. Notice what happens in the UI. You should see a set of docking boxes available for the second button surrounding the first.



toRightOf=@+id/button10
alignTop=@+id/button10
alignBottom=@+id/button10

Dock the second button so it's "toRightOf" your first button, and alignTop and alignButtom are identified by the first button as well (@+id/button10).

Now, modify the layout_height property of the first button (button10), setting it to 60dp. You'll see that the height of the second button also changes. Modify the layout_width of the first button (120dp), and you'll see that the width of the second button doesn't change. Why? Simply put, alignTop and alignBottom properties force the height dimensions to be the same between the two buttons, but not the width. If we were to place button11 below button10, we could use alignLeft, and alignRight to set's width to be the same as button10, but we'd lose the 60dp height for button11.

Save your layout and close it.

In Eclipse, click on the "New Java Class" option to create a new class file called Other_Layouts_Example, which inherits from android.app.Activity.

Open the class and add the following code:

```
public class Other_Layouts_Example extends Activity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                // TODO Auto-generated method stub
                super.onCreate(savedInstanceState);
                setContentView(R.layout.other_layouts_example);

                final RatingBar oRate = (RatingBar) findViewById(R.id.ratingBar1);
                final SeekBar oSeek = (SeekBar) findViewById(R.id.seekBar1);

                oRate.setOnRatingBarChangeListener(new RatingBar.OnRatingBarChangeListener() {

                        @Override
                        public void onRatingChanged(RatingBar ratingBar, float rating,
                                        boolean fromUser) {
                                Toast.makeText(
                                                getApplicationContext(),
                                                "SeekBar value = " + oSeek.getProgress() + "\n"
                                                                + "Rating is: " + oRate.getRating(),
                                                Toast.LENGTH_SHORT).show();

                        }
                });

        }
}
```

In the above code, we set call our "other_layouts_example" layout first, then grab a reference to the seekbar and ratingbar widgets. We use the rating bar's setOnRatingBarChangeListener to toast a message that shows us the current values of the rating bar and seekbar.

Open your Lesson5UI class file. Add the following code to allow us to get to the newly created class file:

```
public class Lesson5UI extends ListActivity {

        String[] GUIExamples = { "Buttons", // 0
                        "EditText and TextView", // 1
                        "Checkbox and RadioButtons", // 2
                        "Spinner (Dropdown)", // 3
                        "WebView", // 4
                        "Other Layouts" //5
        };

CODE REMOVED TO SAVE SPACE

                case 4: // WebView
                        startActivity(new Intent(this, WebView_Example.class));
                        break;
                case 5: // Other
                        startActivity(new Intent(this, Other_Layouts_Example.class));
                        break;
                }
```

Add the Other_Layouts_Example class to the Android Manifest file, then save and run your program. Review the video for this lesson.