

4 – Layouts in Detail

Layouts arrange the elements of the user interface screen on to the phone (or tablet) display. This section introduces the horizontal and vertical layouts and shows how they can be combined with one another to create complex yet flexible displays that can work with many different screen sizes and orientations.

Lesson 2 introduced the basic three layouts:

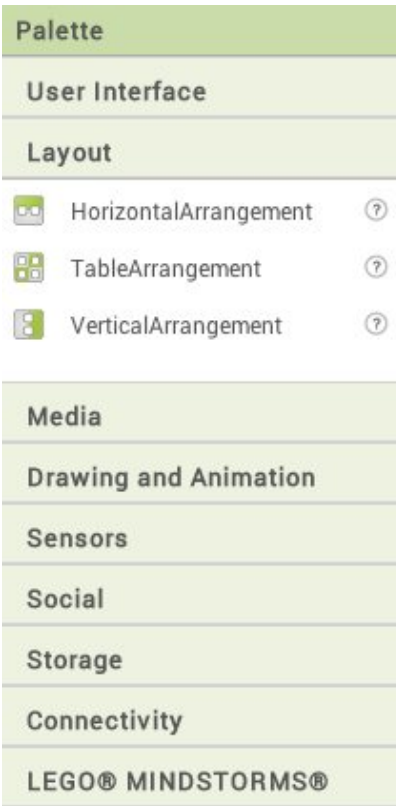
- *HorizontalArrangement*
- *VerticalArrangement*
- *TableArrangement*

These layouts may be combined in many ways. For example, a horizontal layout may appear inside a vertical layout. Or a pair of horizontal layouts may be added one on top of each other, in a vertical layout. In this way, controls can be placed in many different ways on the screen.

To understand layouts, we look at horizontal and vertical layouts and nested layouts, where one layout is inside another. The Table layout was used in Lesson 2 in the design of the calculator user interface.

Source code for each of the examples given in this lesson are in *Lesson4_Layout1*, *Lesson4_Layout2* and *Lesson4_Layout3*.

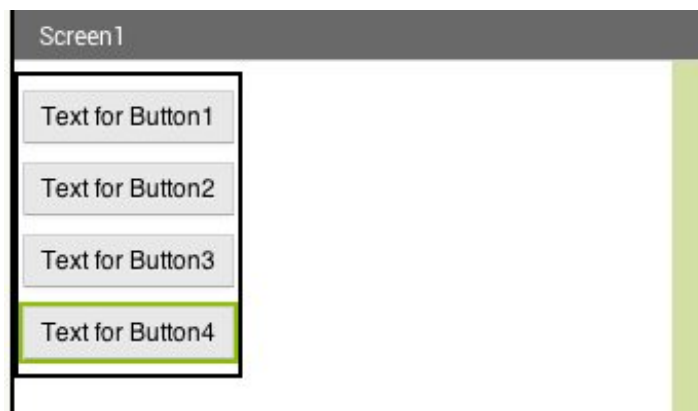
Layouts are selected from the Palette at the left of the App Inventor screen:



A layout is added to the Viewer by clicking on the desired layout and dragging to the Viewer.

The Vertical Layout

Let us start with a blank Viewer (just create a new project). Drag a *VerticalArrangement* into the Viewer. Next, drag and drop some buttons onto the layout:

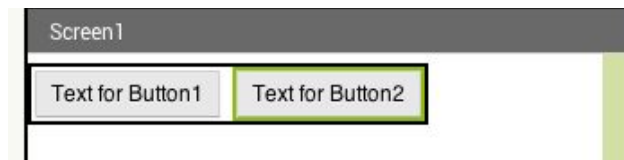


In the vertical layout, items are arranged from top to bottom. The vertical layout zone automatically grows larger as new components are added.

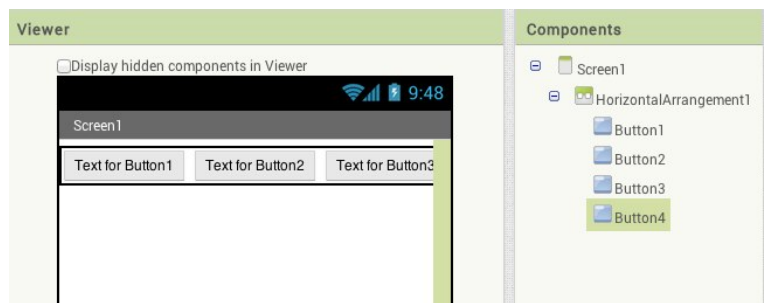
The vertical layout is easy to use but creates a user interface that is not particularly interesting as each user interface control appears in one column. The key to creating interesting layouts is to combine the vertical and horizontal layouts.

The Horizontal Layout

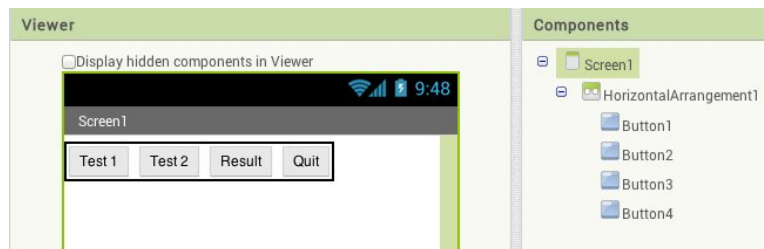
Create a horizontal layout by dragging the *HorizontalArrangement* from the Layout palette to the Viewer. Next, drag a button control in to the horizontal layout. To position a second button in to this layout, drag the button and *drop it directly on top of the first button*. Now, the two buttons are arranged horizontally:



As components are added, they may eventually extend across the Viewer and off the right side, as shown below. The Components list shows there are four buttons but we can only see three of them in the Viewer and the third is partially cut off.

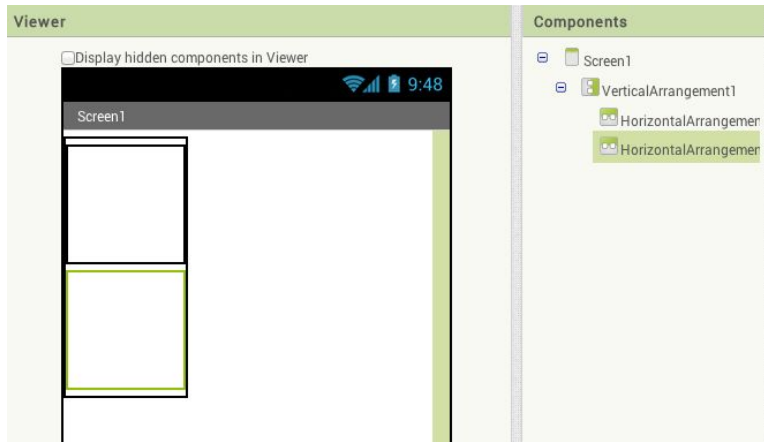


Resolve this by selecting each button and use the Properties | Text field to enter a shorter name. With short names, all four buttons are visible:

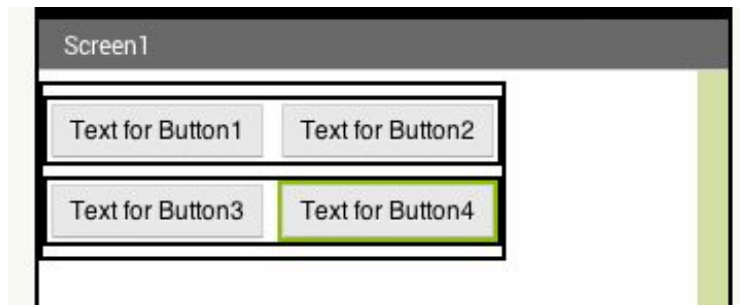


Nested Layouts

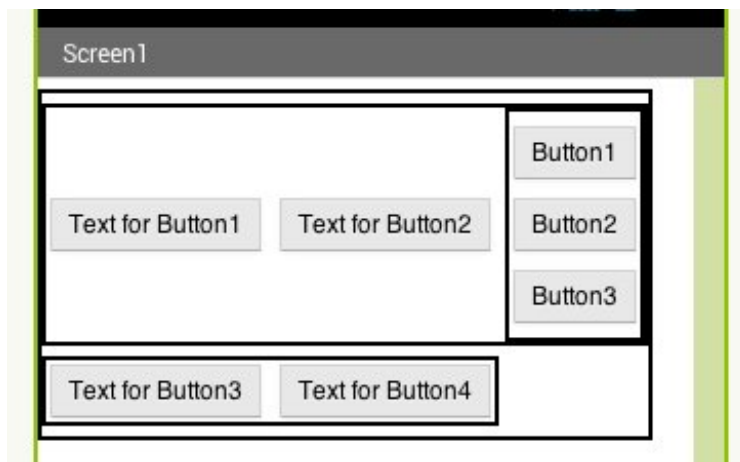
Layouts may be nested within one another. For example, a Vertical layout was placed on the Viewer and then two Horizontal layouts were added inside the Vertical layout.



The nature of this layout is apparent when two buttons are added to each of the horizontal layouts:



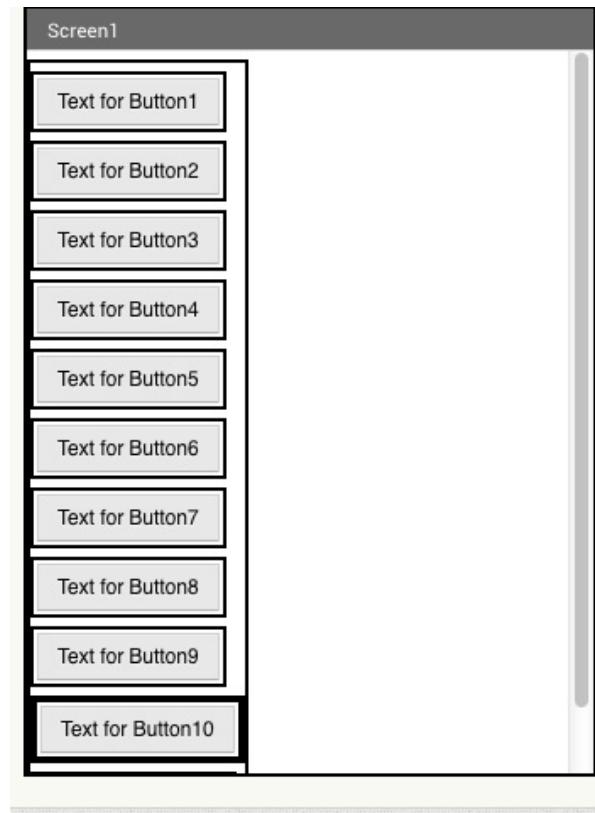
The nesting levels may be placed in any arbitrary fashion. A vertical layout is added to the right side of the first horizontal layout and then three buttons are added, giving this result:



All layouts can be placed inside other layouts. A horizontal layout may be placed inside a Table layout cell, for example. Or a table may be placed inside a vertical layout. And of course, the layouts may include components other than buttons including text boxes, labels and other user interface controls.

Layout Screen Size and Scrolling

You can add layouts and controls even beyond the bottom of the visible Viewer area on screen. When you add additional layouts or components, a scroll bar appears at the right side of the Viewer, as shown in this Viewer screen:



Use “long” screens, as needed, to display large amounts of content or to handle complex data entry.

Tip – Consider avoiding “long” Screens

Users may find several short screens easier to use than one very long screen but it will depend on the application and the users’ needs. There is no hard rule on this but keep this in mind when designing and experimenting with your app’s user interface. In some cases it may be simpler to use two screens and a “Next page” button to advance to the next page, than to create a long, scrolling screen. How to use multiple screens is explained later in Lesson 5.

Restrictions on Layouts

Complex nested layouts require additional computing resources when Android converts the layout in to the screen representation. A layout is “nested” when one layout appears inside another. Google recommends limiting the depth of nested layouts to 10 levels. That’s a guideline, not a hard number.

Restriction on Screen Complexity

In Android user interfaces every component – button, text box, check box, label and so forth – is known as a *View*. Inside Android, a *View* is like a template for most anything that appears on the screen. (Technically, a *View* is a pre-defined *class* in Android. User interfaces are sub-classes of *View*. But this only matters if you are creating Android apps in the Java programming language.) The only reason you, as an App Inventor programmer, need to know this is that Google recommends having no more than 80 *views* on a single activity screen. This too is a guideline and not a hard rule.

Responsive Design Layout

“Responsive Design” refers to features that enable the user interface to adapt to different screen sizes. The use of horizontal and vertical arrangement layouts help to organize the general screen layout for different types of screen shapes and sizes. However, the size of individual controls - such as buttons or text boxes - may also need to adapt to screen sizes.

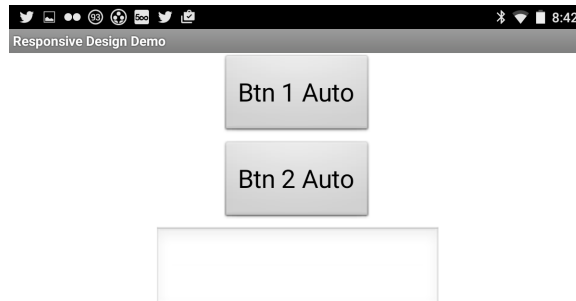
For example, a text box designed to fit on a small smart phone screen might appear very small on a larger tablet screen. But with responsive design, the text box can automatically adapt to an appropriate size for the device.

Using these features, you can create a single app that runs on both a smart phone and a tablet, yet still displays proportional user interface controls.

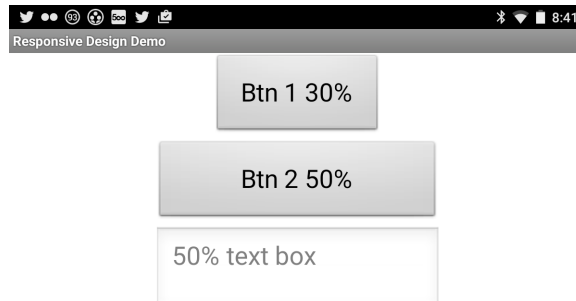
The name “responsive” comes from the ability of the app to “respond” to the size of the device and to change the size of controls so they maintain a similar size on each device.

How This Works

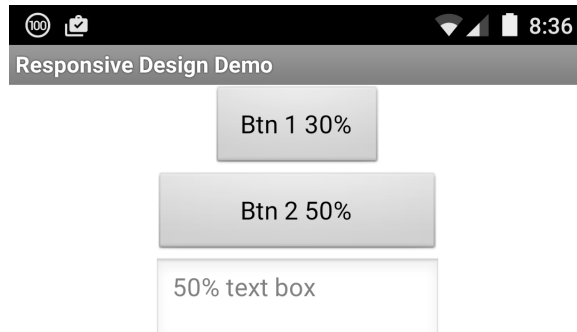
Here is a screen shot from a Nexus 7 tablet of an app with 2 buttons and 1 text box. Each component has its vertical and horizontal width set to “Automatic”.



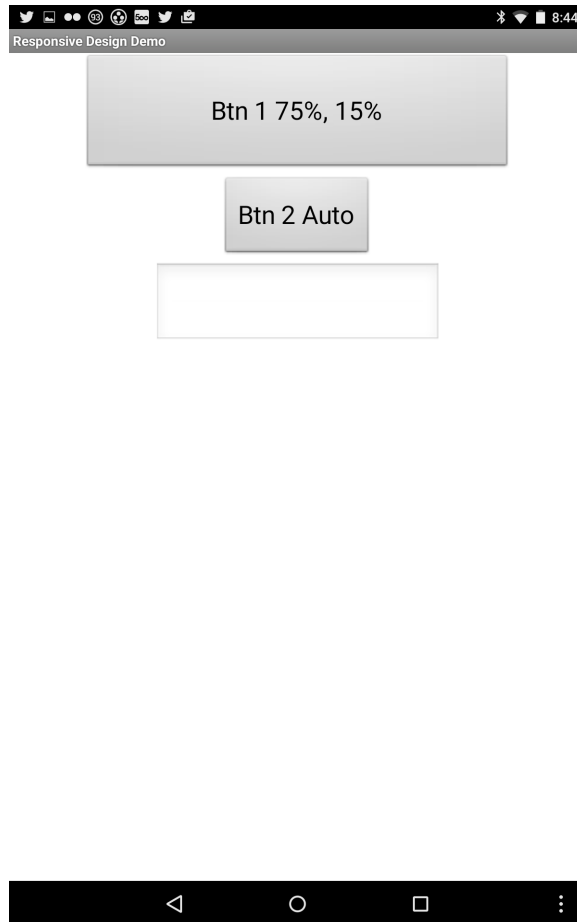
We can change the button properties to specify a width in percent, rather than automatic or a “Fixed” (one size only). Here, button 1 is changed to a 30% width and button 2 is changed to a 50% width – note how the 50% sized button uses up half the screen width on this Nexus 7 screen:



Here is the same app running on a Nexus 5 smart phone – note that the sizes of the buttons are proportionately equal on the smart and the Nexus 7 tablet above and the button set to a “50%” width remains at 50% the width of the larger tablet screen.

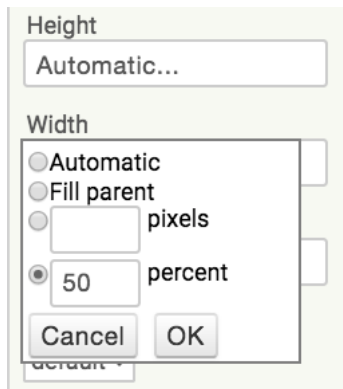


Both the height and width may be set as a percentage – in this example, the width of button 1 has been set to 75% and the height has been set to 15%, creating a taller button than the normal button size:



Setting the Height and Width Properties

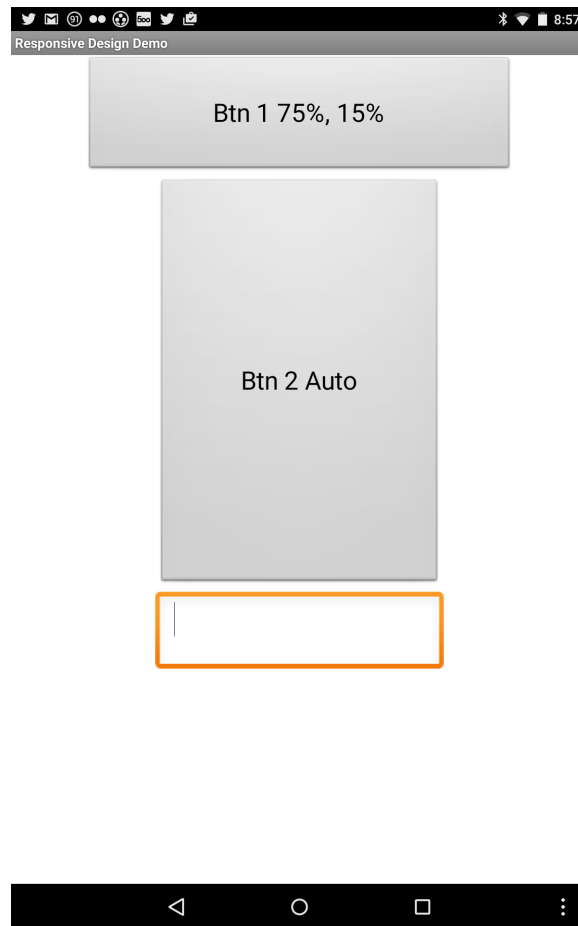
Select a control and then refer to the *Height* and *Width* properties for the component. In this example, the width is set to 50% of the parent (which could be a vertical or horizontal layout):



These properties may also be set programmatically. For example, this blocks code changes the size of a button while the app is running!

```
when Button2 .Click
do
  set Button2 . HeightPercent to 50
  set Button2 . WidthPercent to 50
```

Which produces this funny result when button 2 is pressed:



Summary

“Layouts” are the primary method of arranging items on the user interface screen. In App Inventor, the three available layouts are:

- HorizontalArrangement
- VerticalArrangement

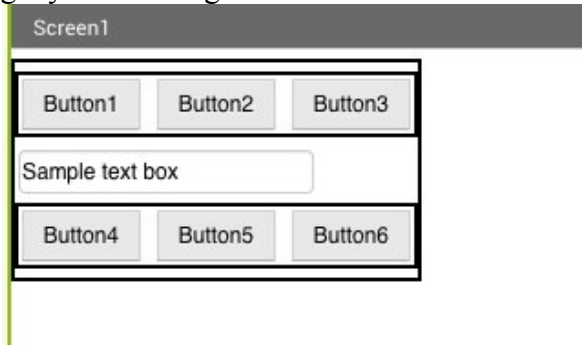
- TableArrangement

Layouts can be used by themselves but are more commonly “nested” within one another to create more complex arrangements of user interface features.

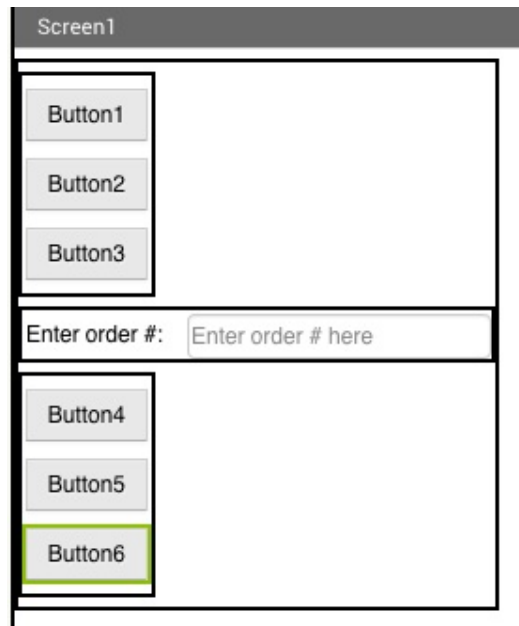
Practice

Practice using layouts to arrange user interface items on the Viewer.

1. Create the following layout in Designer



2. Create the following layout in Designer. This practice introduces the user interface part called “label”. A label is just text such as used here to indicate what should be entered in the text box.



3. Create a layout and screen design of your choice but build it so that it extends beyond the bottom of the Viewer. You'll need to add sufficient layouts or components into the Viewer before the scroll bar appears.

4. Using the layout features revise the user interface of your own app (assuming you have started one!) If you have not started your own app, now would be a great time to get that underway – you have learned enough user interface components to lay out most of the types of applications that are supported in App Inventor.