B4X Booklets

B4X Visual Designer

 Edition 2.2

Last update: 2021.07.12

Main contributors: Klaus Christl (klaus), Erel Uziel (Erel)

To search for a given word or sentence use the Search function in the Edit menu.

All the source code and files needed (layouts, images etc.) of the example projects in this guide are included in the SourceCode folder.

Updated for:
B4A version 11.0
B4i version 7.50
B4J version 9.10

B4X Booklets:
B4X Getting Started
B4X Basic Language
B4X IDE Integrated Development Environment
B4X Visual Designer
B4X Help tools

B4XPages Cross-platform projects
B4X CustomViews
B4X Graphics
B4X XUI B4X User Interface
B4X SQLite Database
B4X JavaObject NativeObject

B4R Example Projects

You can consult these booklets online in this link [B4X] Documentation Booklets.
Be aware that external links don't work in the online display.

1 B4X platforms

B4X is a suite of BASIC programming languages for different platforms.

B4X suite supports more platforms than any other tool
ANDROID | IOS | WINDOWS | MAC | LINUX | ARDUINO | RASPBERRY PI | ESP8266 | AND MORE...

- **B4A**  **Android**

  B4A is a **100% free** development tool for Android applications, it includes all the features needed to quickly develop any type of Android app.

- **B4i**  **iOS**

  B4i is a development tool for native iOS applications.
  B4i follows the same concepts as B4A, allowing you to reuse most of the code and build apps for both Android and iOS.

- **B4J**  **Java / Windows / Mac / Linux / Raspberry PI**

  B4J is a **100% free** development tool for desktop, server and IoT solutions.
  With B4J you can easily create desktop applications (UI), console programs (non-UI) and server solutions.
  The compiled apps can run on Windows, Mac, Linux and ARM boards (such as Raspberry Pi).

- **B4R** 

Arduino / ESP8266

  B4R is a **100% free** development tool for native Arduino and ESP8266 programs.
  B4R follows the same concepts of the other B4X tools, providing a simple and powerful development tool.
  B4R, B4A, B4J and B4i together make the best development solution for the Internet of Things (IoT).

- **B4XPages**

  B4XPages is an internal library for B4A, B4i and B4J allowing to develop easily cross-platform programs.
  B4XPages is explained in detail in the B4XPages Cross-platform projects booklet.
  Even, if you want to develop only in one platform it is interesting to use the B4XPages library it makes the program flow simpler especially for B4A.

2 Layouts

Designing layouts is a major concern for developers.
A well organized and nice-looking user interface makes a program being accepted immediately by the users or not, and this on different devices with different screen sizes.
Most users, when they look at a new application, decide in the first minutes if they will go further or not! Me too, when I download an application and there are several with the same purpose, the first impression is crucial. If I don't like the layout I don't keep it.
You should have a look at:

- Androids' guidelines
- Apples' guidelines about how to design UI Design Basics.

For the navigation between different pages you should use the standard OS objects instead of reinventing the wheel. Users are used to them and feel directly 'at home'!
Android users are used to UI with Android look whereas Apple users may prefer the Apple look.

It's up to you to define what layout you want, what you want to display at the same time and how you want to navigate through the different displays.
In some cases, it might be better to have one or two layouts (portrait and / or landscape) phones and one or two layouts for tablets. Use as little layout variants as needed and use the different tools to adapt them to fit the different screen sizes.
As tablets have bigger screens than phones it could be interesting to display more information on tablets than on phones.
Depending on the application, it could be interesting to display one panel on pages in portrait on phones and display two panels side by side on a same page on tablets.

There are no general rules nor templates for user interfaces. They depend on the kind of application, the kind of information to display on what screen size, the number of different pages depending on the screen size and the information, etc.

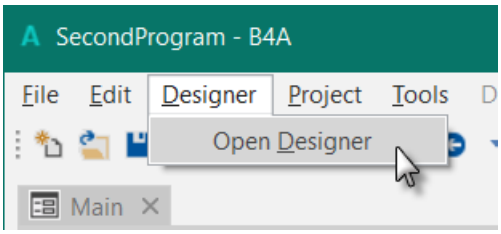
Several tools are at your disposal to design the layouts, these are explained in the following chapters.

3 Visual Designer

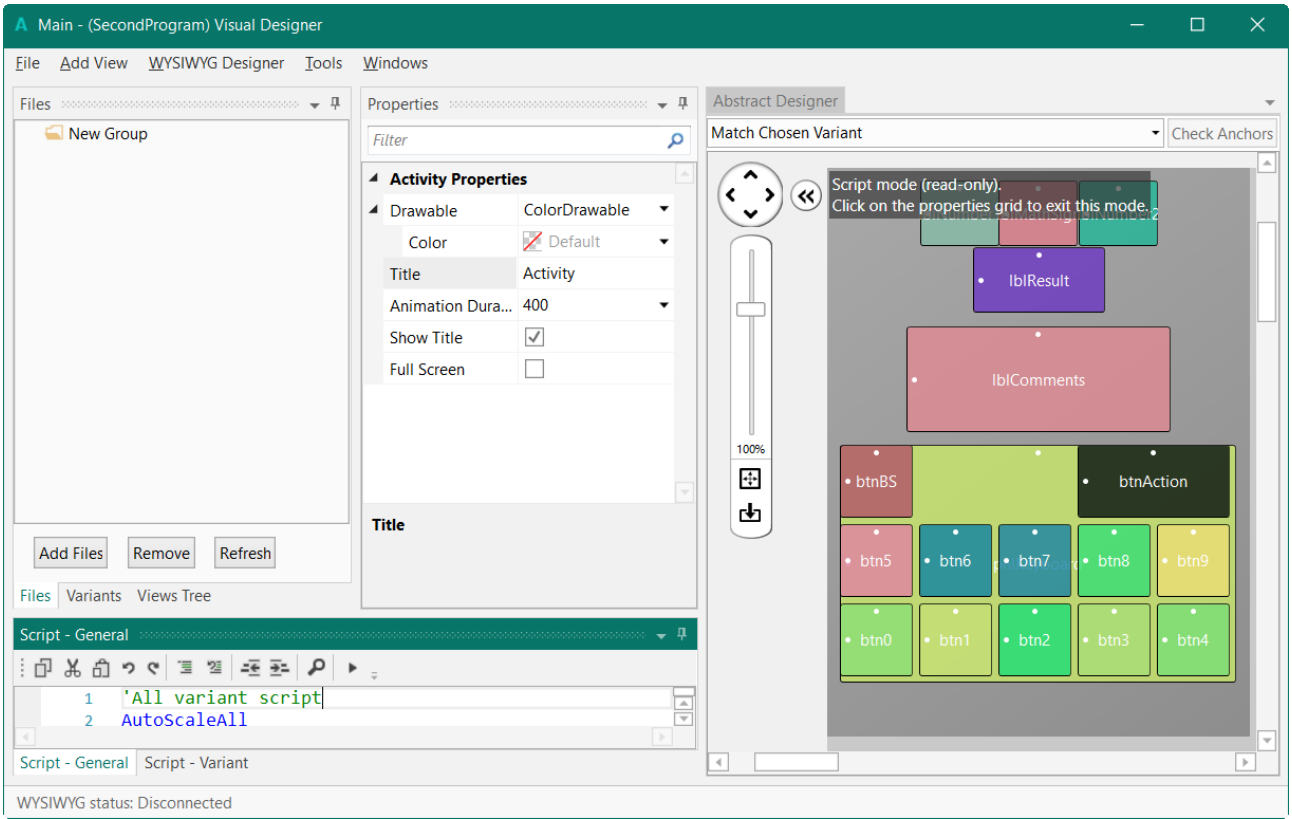
The Visual Designer allows generating layouts with either the Abstract Designer or with a real device. You can also use Emulators but not recommended.

All the images in this booklet are made with the B4A Designer, but the others look similar with different themes.
Specific images for B4i or B4J are shown when needed.

Launch the Designer in the IDE Menu Designer.

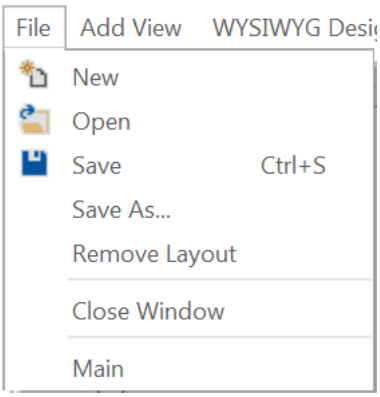


The default Visual Designer looks like this, the layout in the Abstract Designer is from the SecondProgram project.



3.1 The menu

3.1.1 File menu

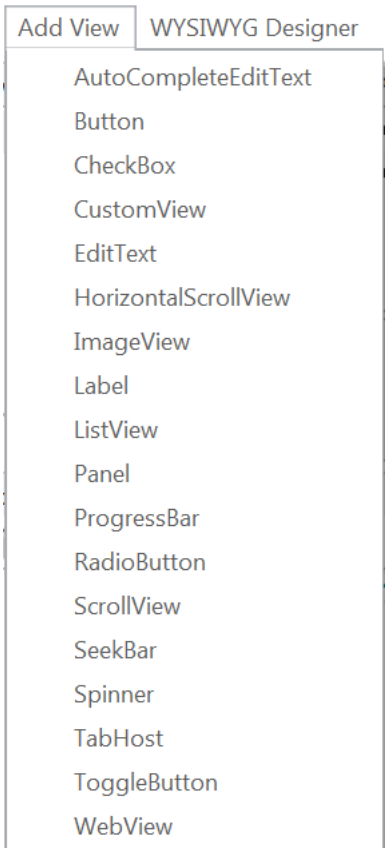


New	Opens a new empty layout.
Open	Opens an existing layout.
Save	Saves the current layout.
Save As…	Saves the current layout with a new name.
Remove Layout	Removes the layout from the Files directory.
Close Window	Closes the Visual Designer.
Main	Layout file list, in this case only one file, 'Main'.

3.2 AddView menu

3.2.1 B4A AddView menu

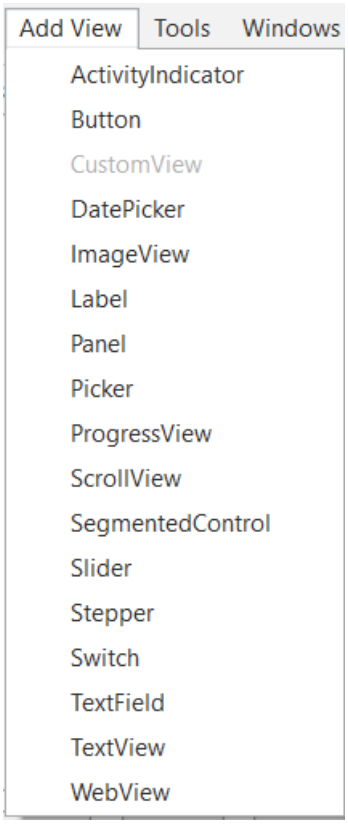
This menu allows you to add views to the current layout.



AutoCompleteEditText	adds an AutoCompleteEditText
Button	adds a Button
CheckBox	adds a CheckBox
CustomView	adds a CustomView
EditText	adds an EditText
HorizontalScrollView	adds a HorizontalScrollView
ImageView	adds an ImageView
Label	adds a Label
ListView	adds a ListView
Panel	adds a Panel
ProgressBar	adds a ProgressBar
RadioButton	adds a RadioButton
ScrollView	adds a Scrollview
SeekBar	adds a SeekBar
Spinner	adds a Spinner
TabHost	adds a TabHost
ToggleButton	adds a ToggleButton
WebView	adds a WebView

3.2.2 B4i AddView menu

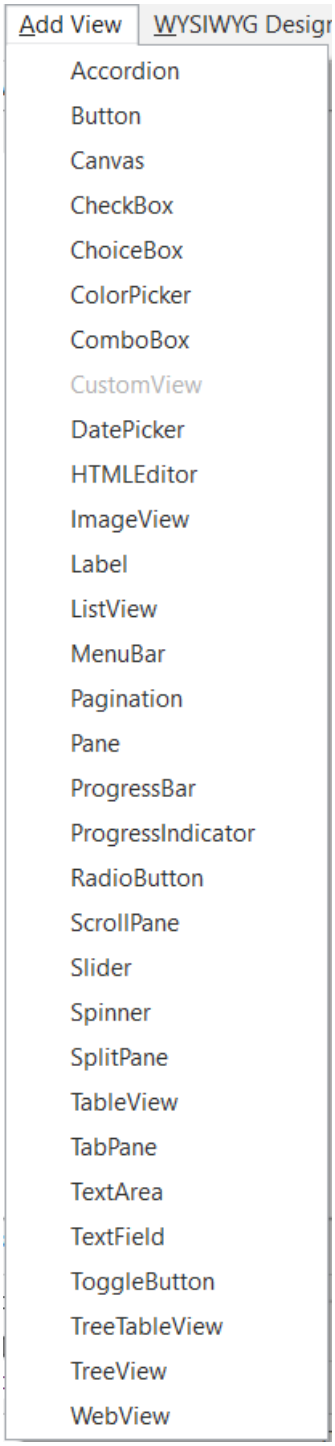
This menu allows you to select the view you want to add to the current layout.



ActivityIndicator	adds an ActivityIndicator
Button	adds a Button
CustomView	adds a CustomView if there are any.
ImageView	adds an ImageView
Label	adds a Label
Panel	adds a Panel
Picker	adds a Picker
ProgressView	adds a ProgressView
ScrollView	adds a ScrollView
SegmentedControl	adds a SegmentedControl
Slider	adds a Slider
Stepper	adds a Stepper
Switch	adds a Switch
TextField	adds a TextField
TextView	adds a TextView
WebView	adds a WebView

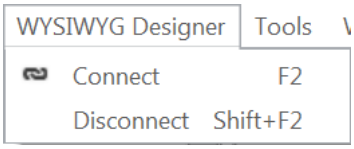
3.2.3 B4J AddView menu

This menu allows you to select the view you want to add to the current layout.



Accordion	adds an Accordion
Button	adds a Button
Canvas	adds a Canvas
CheckBox	adds a CheckBox
ChoiceBox	adds a ChoiceBox
ColorPicker	adds a ColorPicker
ComboBox	adds a ComboBox
CustomView	adds a CustomView if there are any.
DatePicker	adds a DatePicker
HTMLEditor	adds an HTMLEditor
ImageView	adds an ImageView
Label	adds a Label
ListView	adds a ListView
MenuBar	adds a MenuBar
Pagination	adds a Pagination
Pane	adds a Pane
ProgressBar	adds a ProgressBar
ProgressIndicator	adds a ProgressIndicator
RadioButton	adds a RadioButton
ScrollPane	adds a ScrollPane
Slider	adds a Slider
Spinner	adds a Spinner
SplitPane	adds a SplitPane
TableView	adds a TableView
TabPane	adds a TabPane
TextArea	adds a TextArea
TextField	adds a TextField
ToggleButton	adds a ToggleButton
TreeTableView	adds a TreeTableView
TreeView	adds a TreeView
WebView	adds a WebView

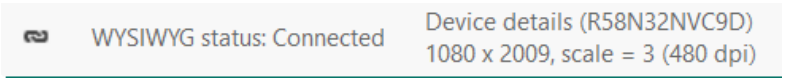
3.2.4 WYSIWYG Designer menu B4A, B4J



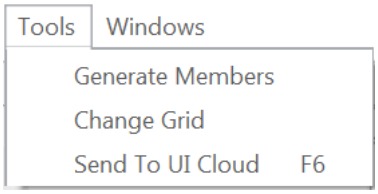
Connects a device or an Emulator to the Visual Designer.
Disconnect from Device / Emulator.

For details on how to connect a real device look at chapter *B4A connecting a real device* in the Getting started B4X Booklet. In B4J it connects to a Form.

The connected device with its prameters is displayed in the lower left corner.

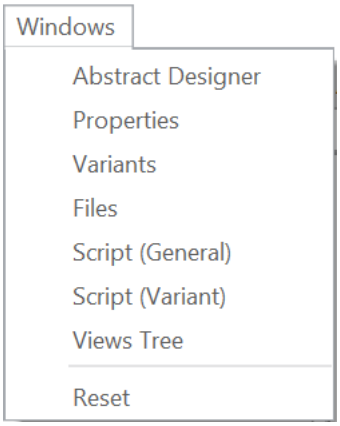


3.2.5 The Tools menu



Generate Members	Members generator
Change Grid	Allows to change the grid size
Send To UI Cloud.	B4A and B4i only.

3.2.6 Windows menu



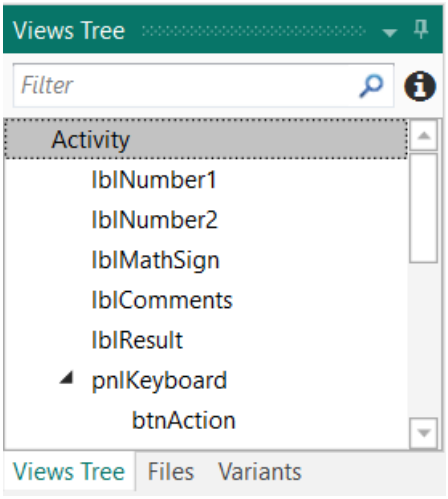
Shows the Abstract Designer window.
Shows the Properties window.
Shows the Variants window.
Shows the Files window.
Shows the Script (General) window.
Shows the Script (Variant) window.
Shows the Views window.

Resets the Visual Designer layout to the default layout.

3.3 Visual Designer Windows

The Visual Designer is composed of different windows.

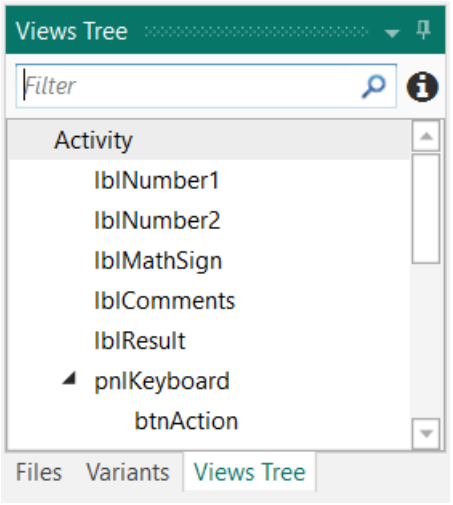
3.3.1 Views windows Views Tree / Files / Variants



In this Window three windows are grouped together:

- Files
- Variants
- Views Tree.

3.3.1.1 Views Tree window



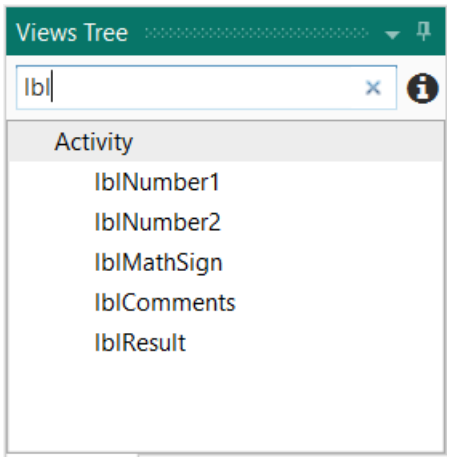
Shows all views of the selected layout in a tree.

When you select a view in the list, all the properties of the selected view are displayed in the Properties window.

You can select several Views at the same time and change common properties.

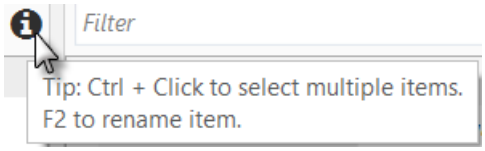
The selected views are highlighted in the Abstract Designer.

On top you can filter the views / nodes.

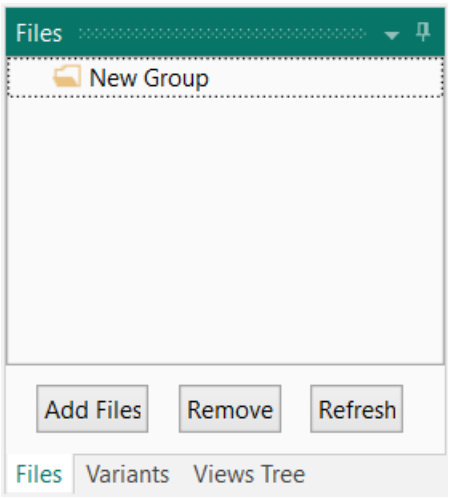


Enter 'lbl' and all the views containing lbl will be displayed in the list.

shows tips:



3.3.1.2 Files Windows

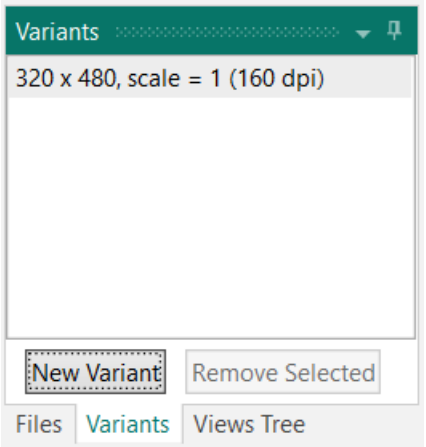


Used to add or remove files to the Visual Designer, mainly image files.

File handling is explained in the Image Files chapter.

These files are copied to the Files folder of the project and can be accessed in the code in the File.DirAssets folder.

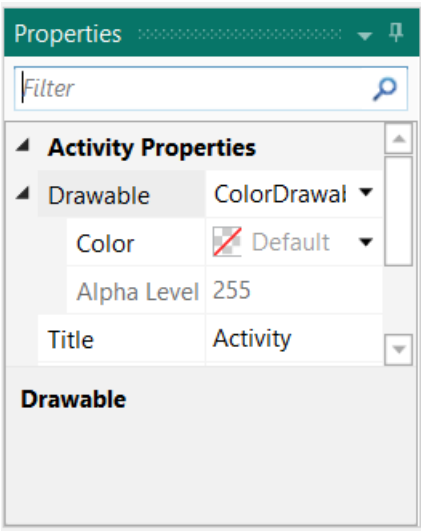
3.3.1.3 Variants window



Used to add and remove layout variants.

Layout variants are explained in the Layout variants chapter.

3.3.2 Properties window



The Properties window shows all properties of the selected View.

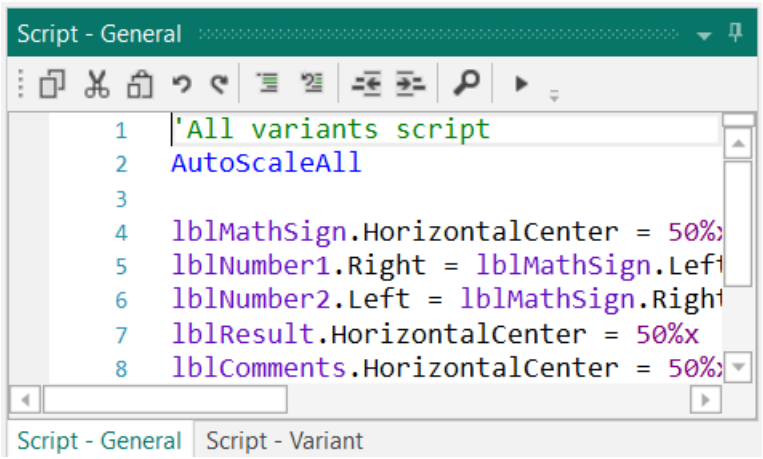
The Properties are explained in the Properties list chapter.

3.3.3 Script (General) / (Variant) windows

In the Scrip windows you can add code to position and resize Views.

Two windows are available:

- **Script - General** Code valid for all layout variants.
- **Script - Variant** Specific code for the selected layout variant.

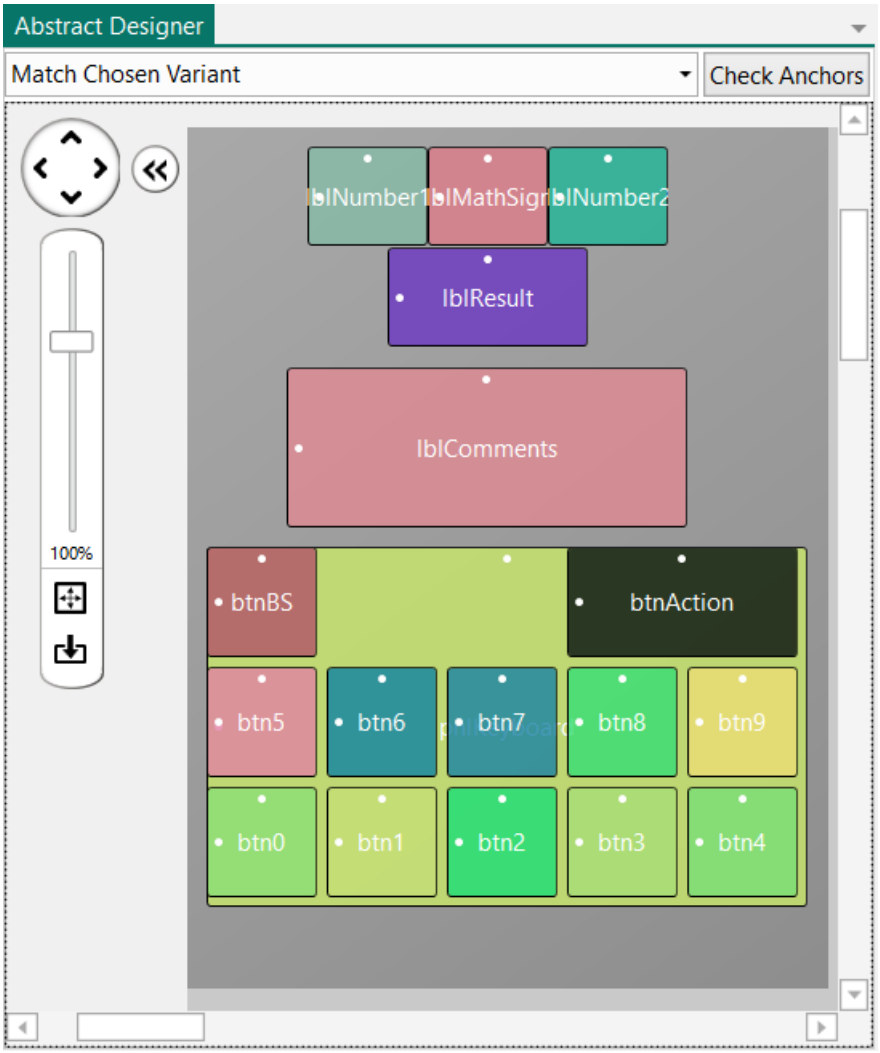


Script code is explained in the Designer Scripts chapter.

3.3.4 Abstract Designer window

The Abstract Designer allows to select, position and resize Views.
It is not a WYSIWYG Designer, for this you need to connect a real device or an Emulator.

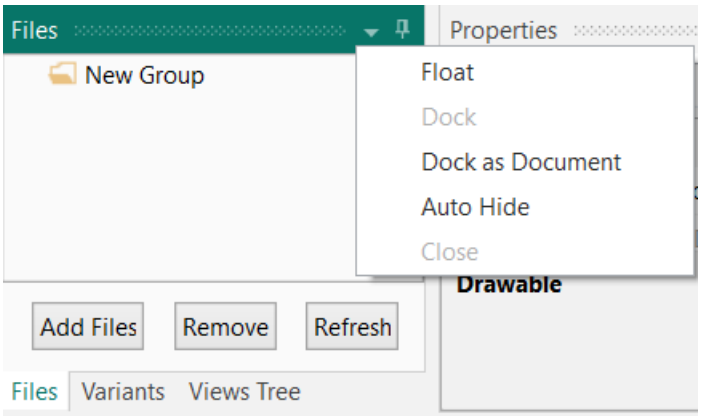
The displayed layout in the picture below is from the SecondProgram project.



3.4 Floating windows

You can define your own Visual Designer layout, rearrange the windows in size and position, docked or floating.

On top of each window two icons allow you to manage the behaviour of this window.



 Options.

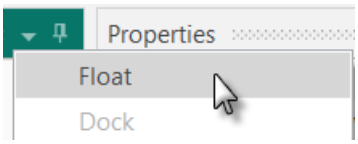
Example with the Files window:

Float sets the window to Float, independent of the Visual Designer window.

Dock as Document

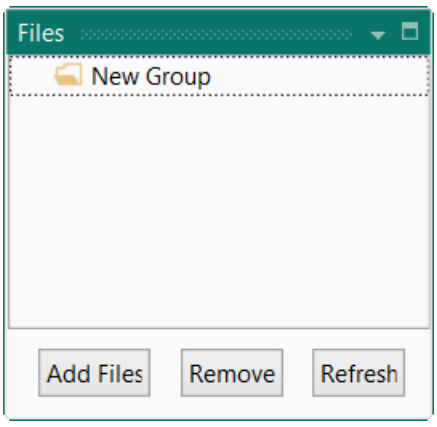
Auto Hide

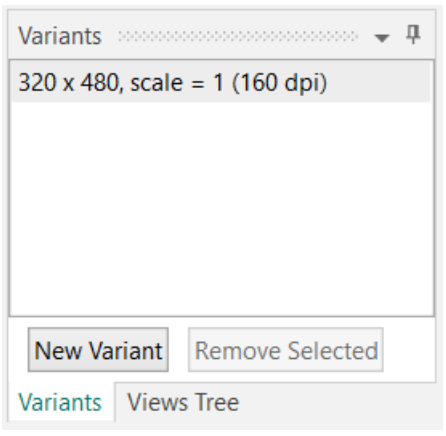
3.4.1 Float

Clock on 

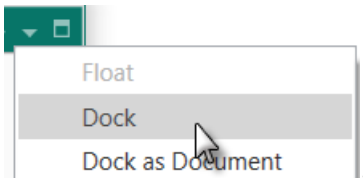
.

The Files windows is independent from the Visual Designer and is removed from the Views window.





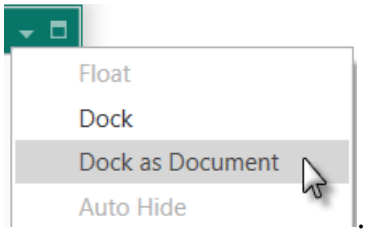
3.4.2 Dock

Click on 

.

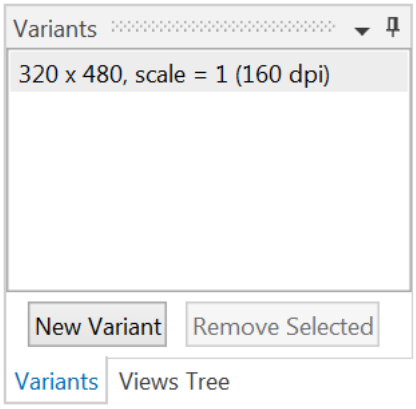
The window is moved back to the Views window.

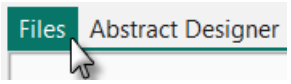
3.4.3 Dock as Document

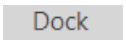
Click on 

.

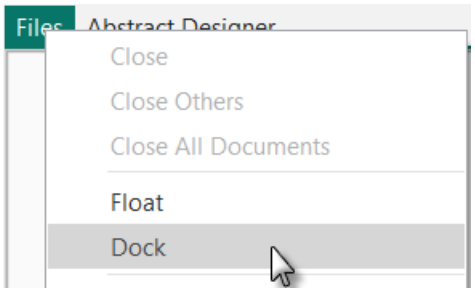
The window is removed from its parent window and added to the Abstract Designer window.



Right click on 

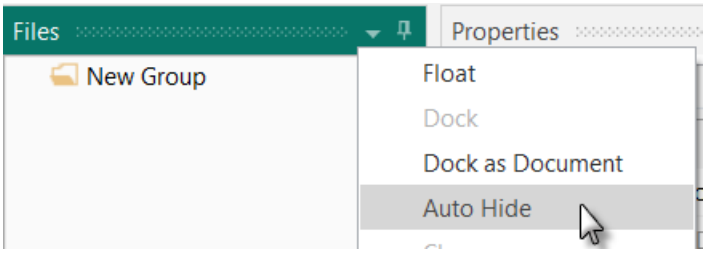
and on 

to move it back to its parent window.

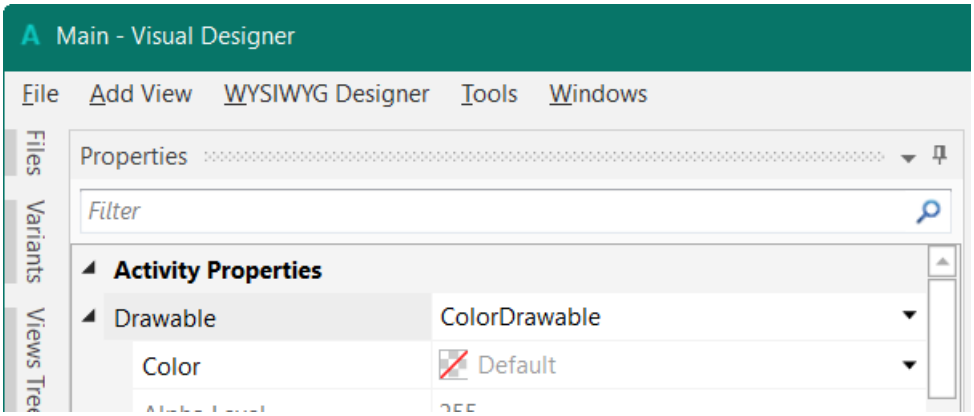


3.4.4 Auto Hide

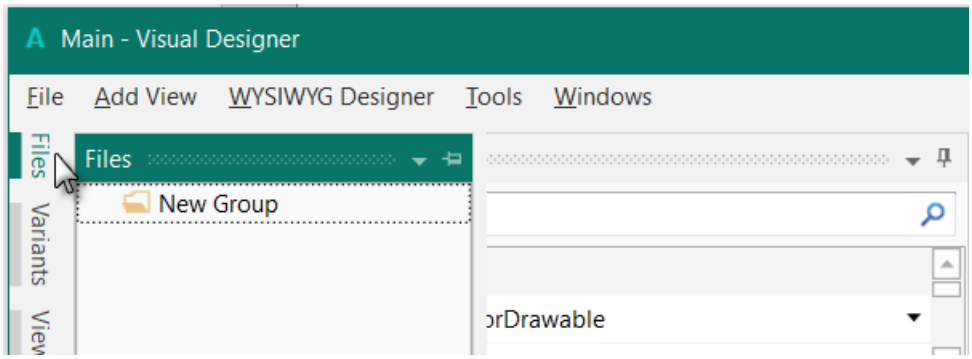
Click either on  or on Auto Hide.



The three windows Files, Variants and Views Tree are moved as Tabs to the left border of the Visual Designer. The Properties window width is increased.

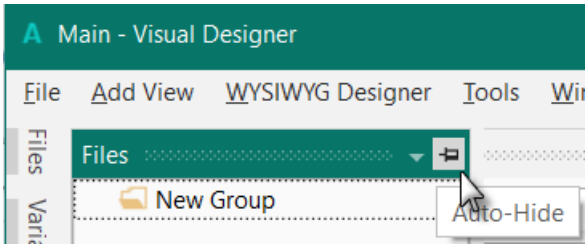


Click on a Tab to show the window.



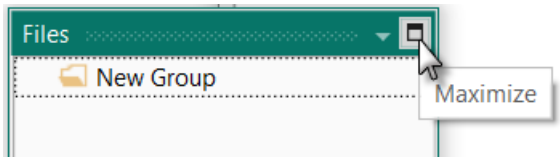
When you click somewhere else, outsides the selected window, hides it automatically.

Click on  in the title to move the windows back to their previous position.

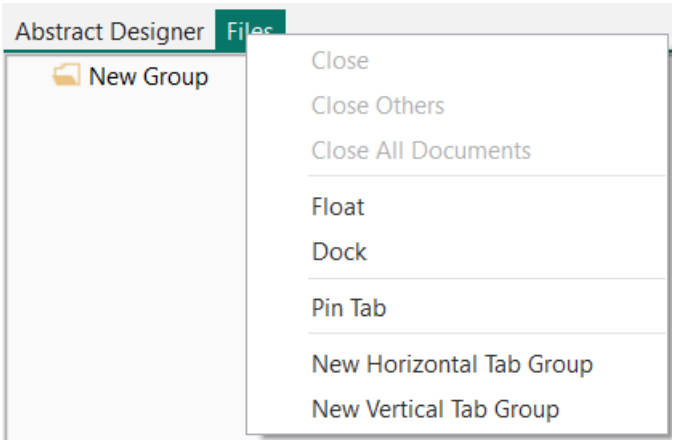


3.4.5 Maximize

Floating windows can be maximized.

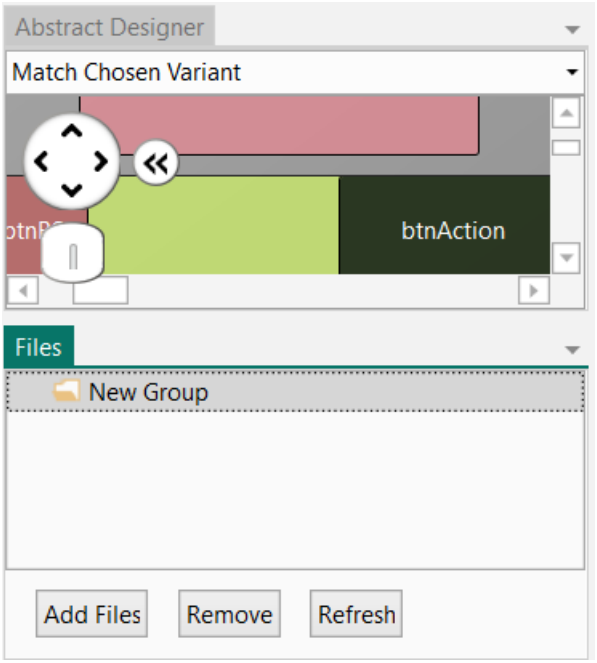


3.4.6 New Horizontal / Vertical Tab Group

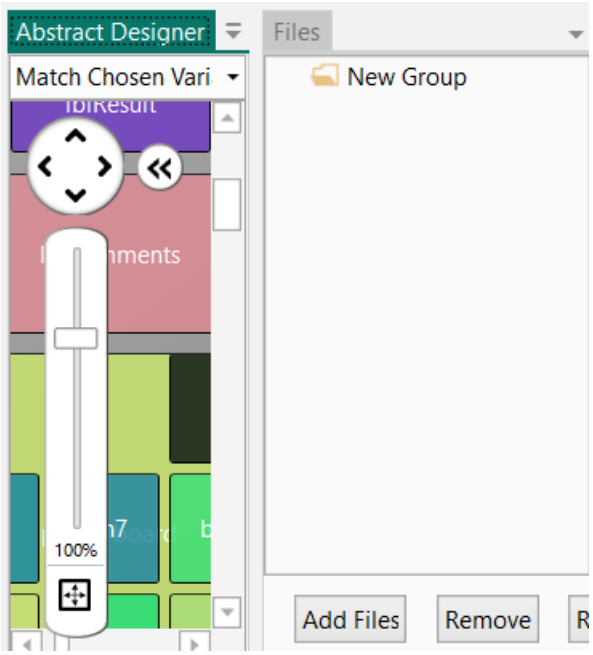


When a window is set as *Dock as Document* two other options are available.

New Horizontal Tab Group
New Vertctal Tab Group

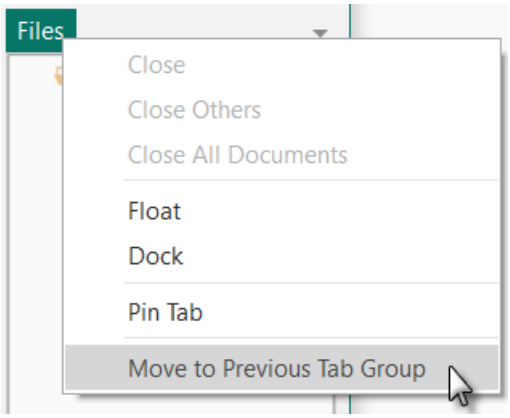


New Horizontal Tab Group



New Vertctal Tab Group

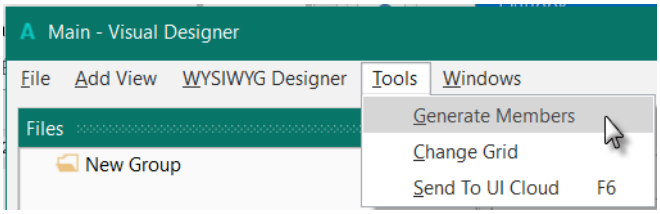
To remove Tab Group right click on Files and click on Move to Previous Tab Group.



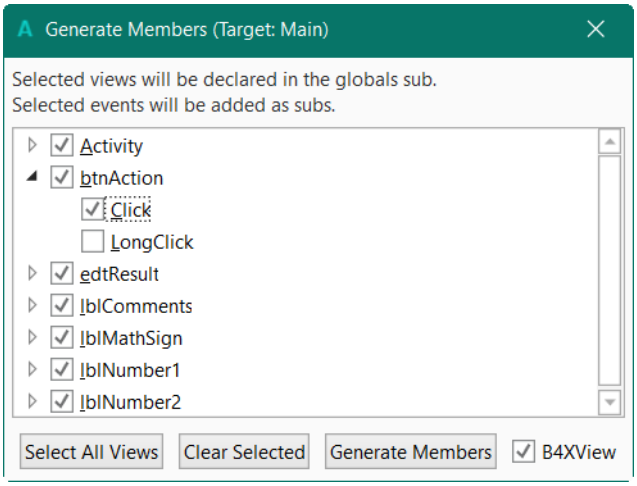
3.5 Tools

3.5.1 Generate Members

Generates declaration statements and subroutines frames. A similar function exists in the Abstract Designer context menu. The example is based on the MyFirstProgram project.



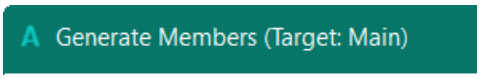
Click on Generate Members to open the generator.



Here we find all the views added to the current layout (MyFirstProgram).
We check all views and check the Click event for the btnAction Button.
Checking a view ▷ ☑ edtResult generates its reference in the Globals Sub routine in the code. This is needed to make the view recognized by the system and allow the autocomplete function.

Views can be declared either as product original views or as B4XViews.
For that check: ☑ B4XView .

Note: The code is added in the active module.
The destination module is displayed in the title



bar .

Variable declarations in Globals

☐ B4XView

```
Sub Globals
   Private btnAction As Button
   Private edtResult As EditText
   Private lblComments As Label
   Private lblMathSign As Label
   Private lblNumber1 As Label
   Private lblNumber2 As Label
```

☑ B4XView

```
Sub Globals
   Private btnAction As B4XView
   Private edtResult As B4XView
   Private lblComments As B4XView
   Private lblMathSign As B4XView
   Private lblNumber1 As B4XView
   Private lblNumber2 As B4XView
```

Clicking on an event of a view ☑ Click generates the Sub frame for this event.

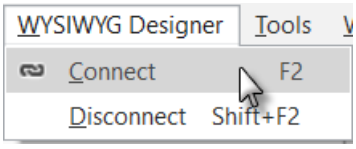
```
Sub btnAction_Click

End Sub
```

Click on Generate Members to generate the references and Sub frames.

Click on Select All Views to select all vies in the list,

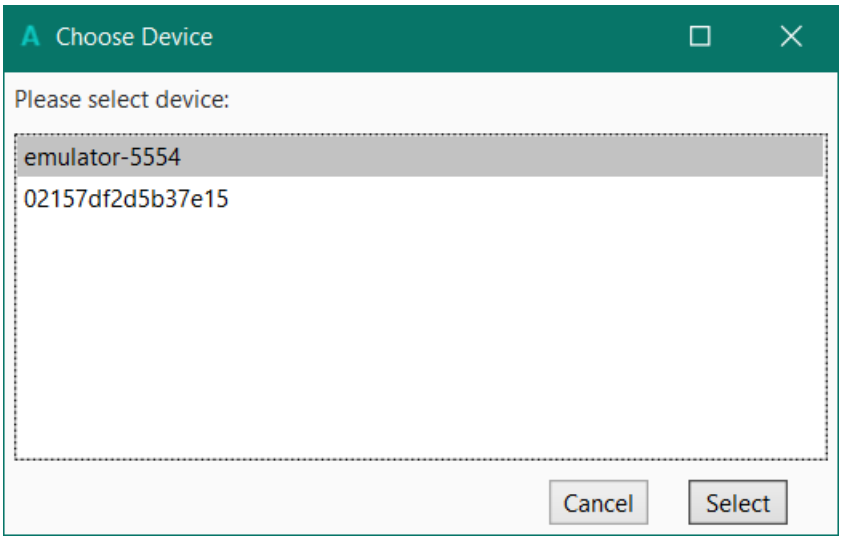
Click on Clear Selected to clear the current selections.

3.5.2 Connect device or emulator



To connect a device or an emulator click Connect in the WYSIWYG Designer menu or press F2.

If different devices or Emulators are connected, you will be asked which device or Emulator you want to connect to.



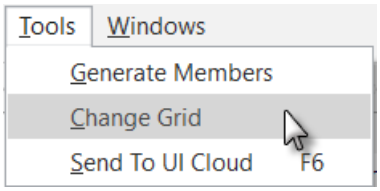
Select an emulator or a device in the list.

Click on Select to confirm.

To disconnect it click on Disconnect Shift+F2 in the WYSIWYG Designer menu or press SHIFT + F2.

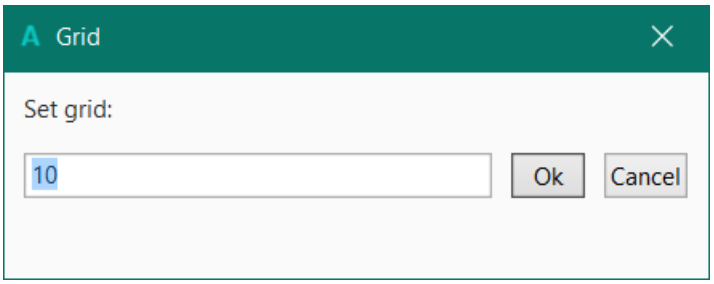
3.5.3 Change grid

The grid is an invisible grid with a given size. The default grid size is 10 pixels. That means that all positions and dimensions of a view will be set to values in steps corresponding to the grid size. Moving a view will be done in steps equal to the grid size.



In the Tools menu click on Change Grid.

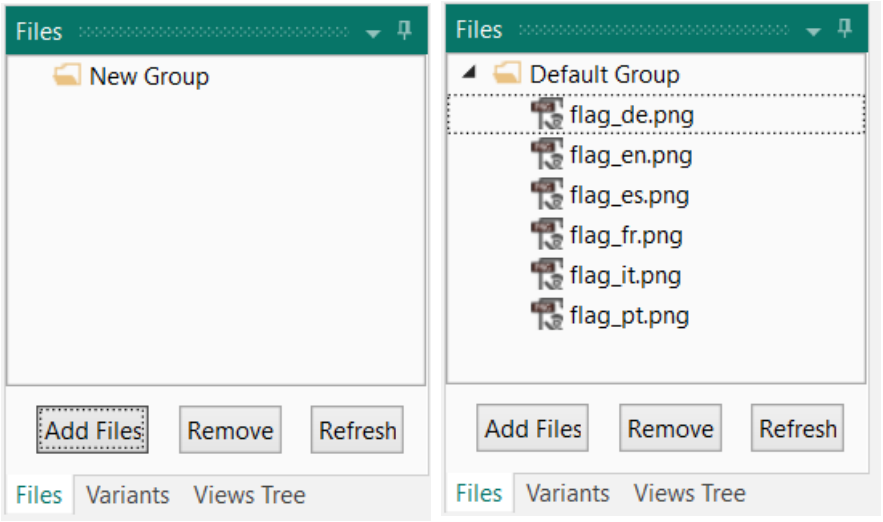
You can change the grid size to the value you want.



The value is saved in the layout file, you will get the same value when you reload this layout.

The default value when you start a new project is 10.

3.6 Image files



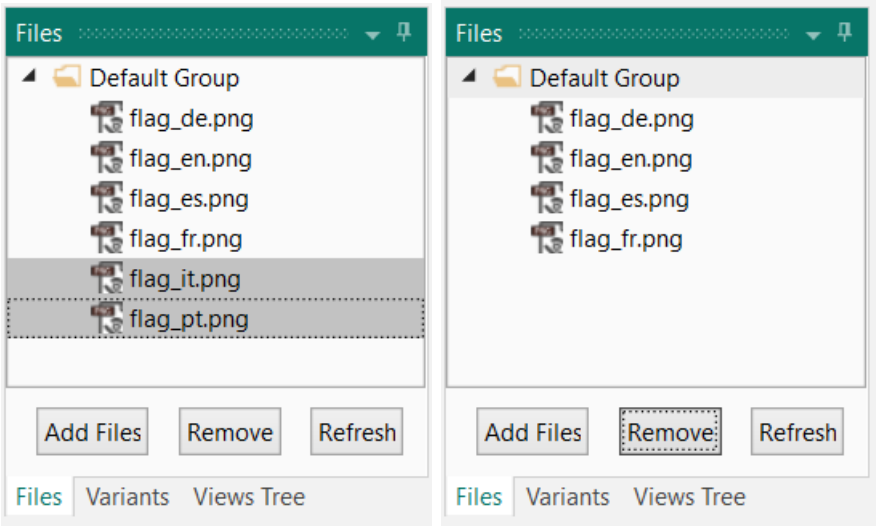
You can add image files to the layout.

Click on Add Files to select the files(s) to add.

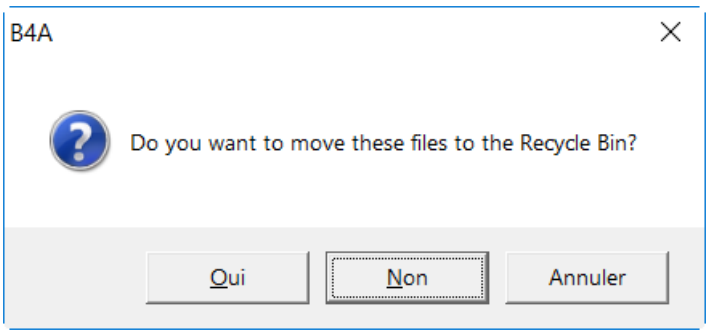
These files will be listed in the Image Files list.

These files are saved to the Files folder of the project and can be accessed in the code in the Files.DirAssets folder.

To remove files, check the files to remove and click on Remove.

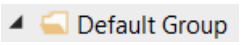


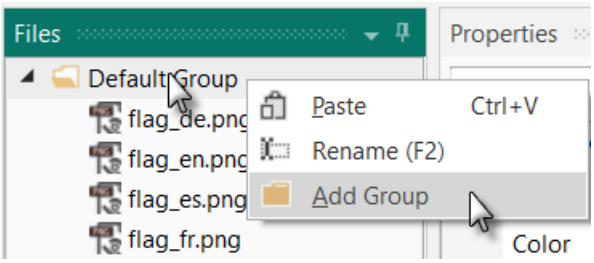
You are asked if you want to move these files to the Recycle Bin.

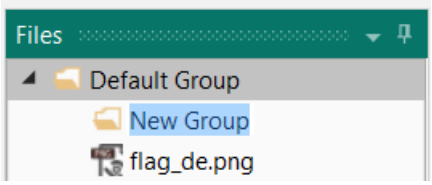


You can define different groups for the files.

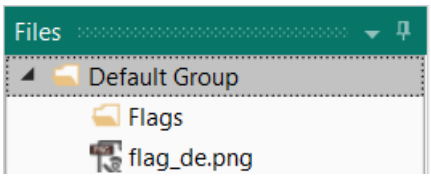
Add a new goup.

Right click on  and click on  .

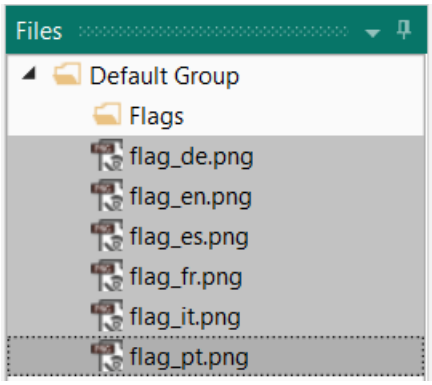




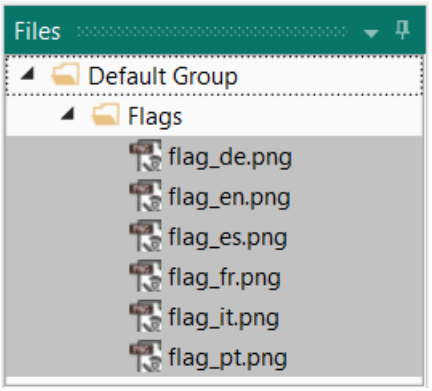
Enter the name.



For example ‘Flags’



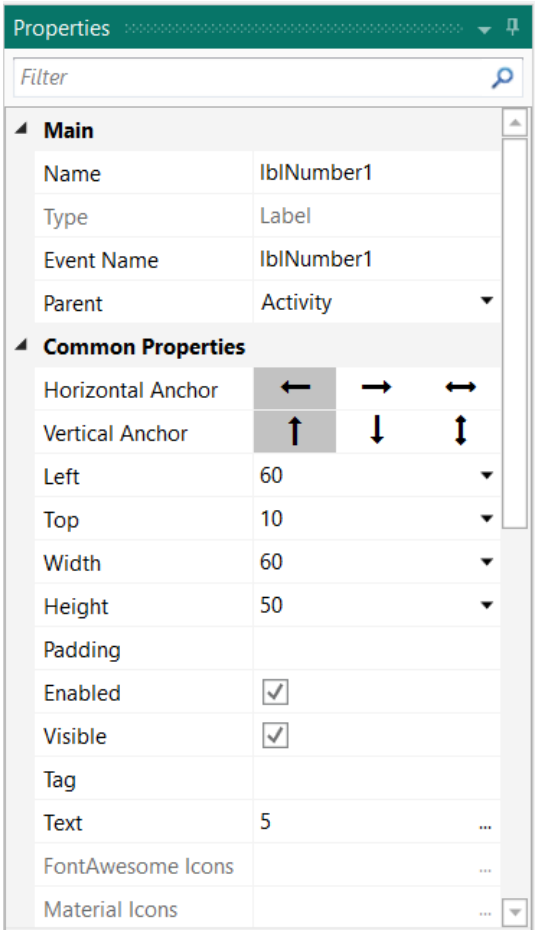
Select the ‘flag’ files and move them into the Flags group.

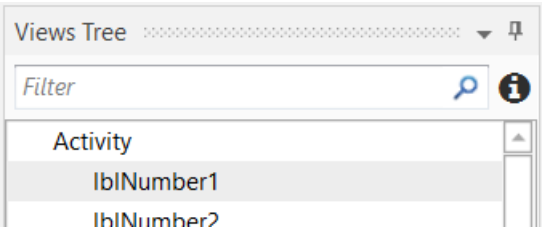


The flag files are now in the Flags group.

Different files can be grouped in deiiferent groups but they remain in the Files folder of the project.

3.7 Properties list



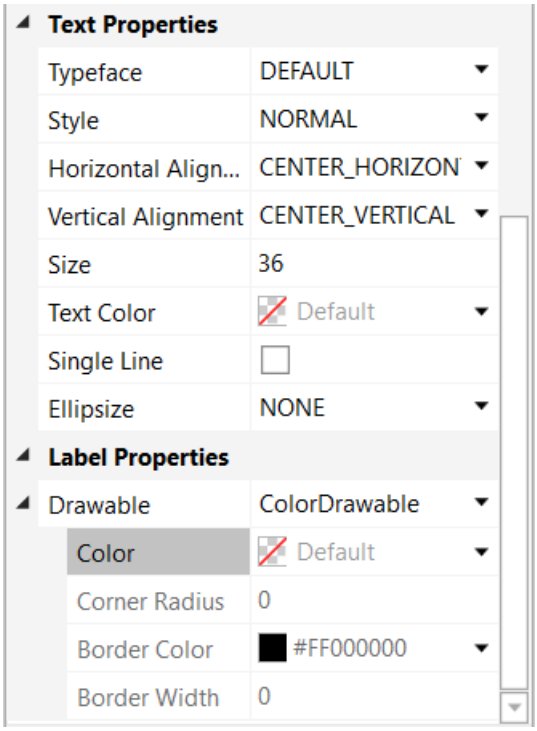


Select for example lblNumber1 in the list.

All the properties of lblNumber1 are displayed. These are organized in groups.

All properties can be modified directly in the list.

All properties in the Main group and some of the properties in the other groups are common to all view types.



3.7.1 Main properties

Name Name of the view (B4A, B4i) or node (B4J). It is good practice to give meaningful names. Common usage is to give a 3-character prefix and add the purpose of the view. In the example, the view is of type Label and its purpose is to enter a result. So, we give it the name "lblResult", "lbl" for
Label and "Result" for the purpose. This does not take much time during the design of the layout but saves a lot time during coding and maintenance of the program.

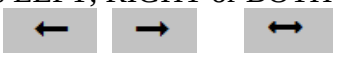
Type Type of the view (B4A, B4i) or node (B4J), not editable. It is not possible to change the type of a view. If you need to, you must remove the view and add a new one.

Event Name Generic name for the subroutines that manages the view's events. By default, the Event Name is the same as the view's name like in the example. The Events of several views can be redirected to a same subroutine. In that case you must enter the name of that routine.
Look at the SecondProgram example for the Click event management for the buttons of the keyboard, the *btnEvent_Click* routine.

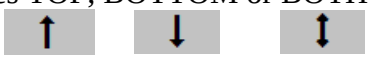
Parent Name of the parent view (B4A, B4i) or node (B4J). Activity, in the example. The parent view can be changed in selecting the new one in the list.

3.7.2 Common properties

HorizontalAnchor Horizontal Anchor function. Possible values LEFT, RIGHT or BOTH



VerticalAnchor Vertical Anchor function. Possible values TOP, BOTTOM or BOTH



Left X coordinate of the left edge of the View from the left edge of its parent View, in pixels (the pixels are in reality dips, density independent pixels).

Top Y coordinate of the upper edge of the View from the upper edge of its parent View, in pixels (the pixels are in reality dips, density independent pixels).

Width Width of the View in pixels (the pixels are in reality dips, density independent pixels).

Height Height of the View in pixels (the pixels are in reality dips, density independent pixels).

Enabled Enables or disables the use of the View Ex: Enabled = True **B4A, B4J**

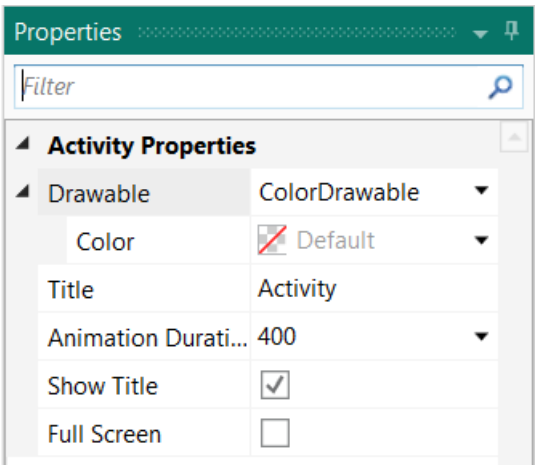
Visible Determines if the View is visible to the user or not.

Tag This is a place holder which can used to store additional data. Tag can simply be text but can also be any other kind of object.
Tag is used in the SecondProgram example for the numeric buttons click events management in the btnEvent_Click routine.

Text The text which will be displayed in the View, this property is only available for views having a Text property.

3.7.3 Activity / Main properties

3.7.3.1 B4A Activity properties



Drawable Sets the Activity background Drawable, the default property is ColorDrawable.

Title Sets the activity title text.

Animation Duration Sets the animation duration in milliseconds.
When you launch the program, the Activity is not shown directly but grows with the given duration. If you set this value to '0' the Activity will be shown instantly.

Show Title Changes the Abstract Designer height.
This setting does not change the Activity property, only the Abstract Designer height.

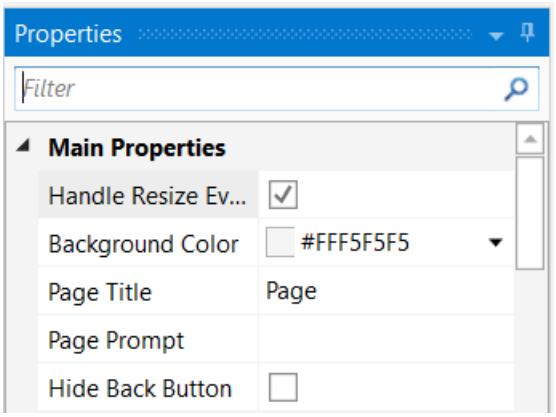
Full Screen Changes the Abstract Designer height.
This setting does not change the Activity property, only the Abstract Designer height.

To not show the titles or set full screen, you need to set these two properties in the Module code in the `Activity Attributes` or `Module Attributes` Regions:

```
#Region  Activity Attributes
   #FullScreen: False
   #IncludeTitle: True
#End Region
```

Checking or unchecking the last two properties only changes the visible screen size in the Abstract Designer.

3.7.3.2 B4i Main properties



Handle Resize Event Handles the resize event.

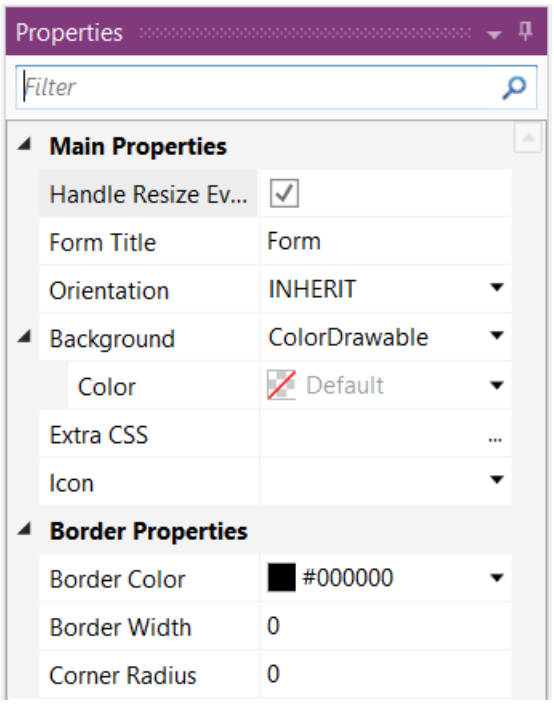
Background Color Sets the Title text of the Page.

Page Title Sets the Title text of the Page.

Page Prompt Sets the Prompt text of the Page.

Hide Back Button Shows or hides the Back button.

3.7.3.3 B4J Main properties



Handle Resize Event Handles the resize event.

Form Title Sets the Title text of the Form.

Orientation Sets the Form orientation.

Background Sets Form Activity background Drawable, the default property is ColorDrawable.

Extra CSS Extra layout properties defined with CSS strings.

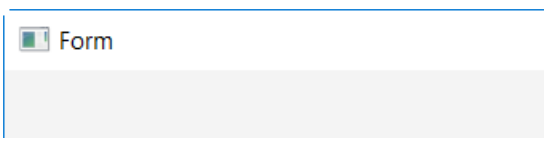
Icon Icon selector.

Border Properties Sets the border properties of the RootPane.

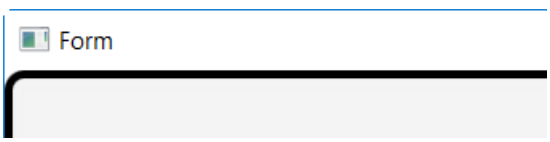
Border Color Sets the border color, default Black.

Border Width Sets the border width, default 0.

Border Radius Sets the border radius, default 0.



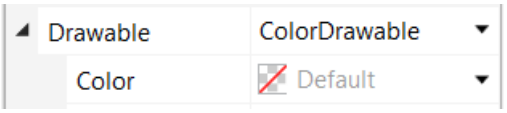
Default



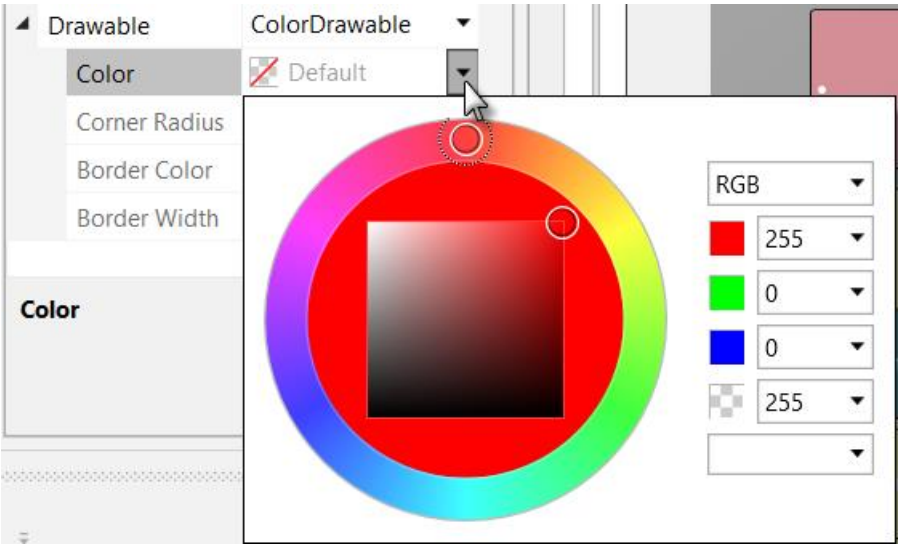
Color: Black, Width: 5, Radius: 10

3.7.4 Color properties

For some properties, like ColorDrawable color, TextColor, you can select a color.

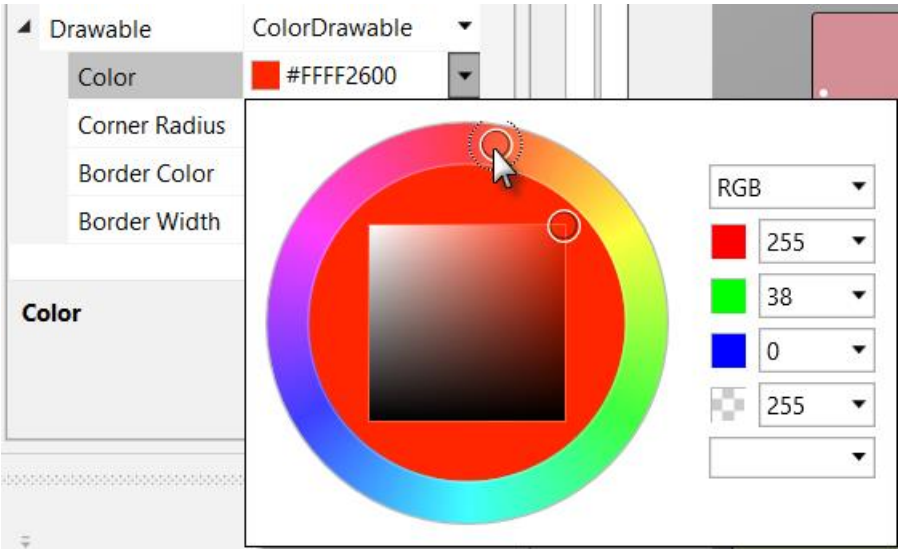


By default, the Default color is selected.



Click on  to select another color.

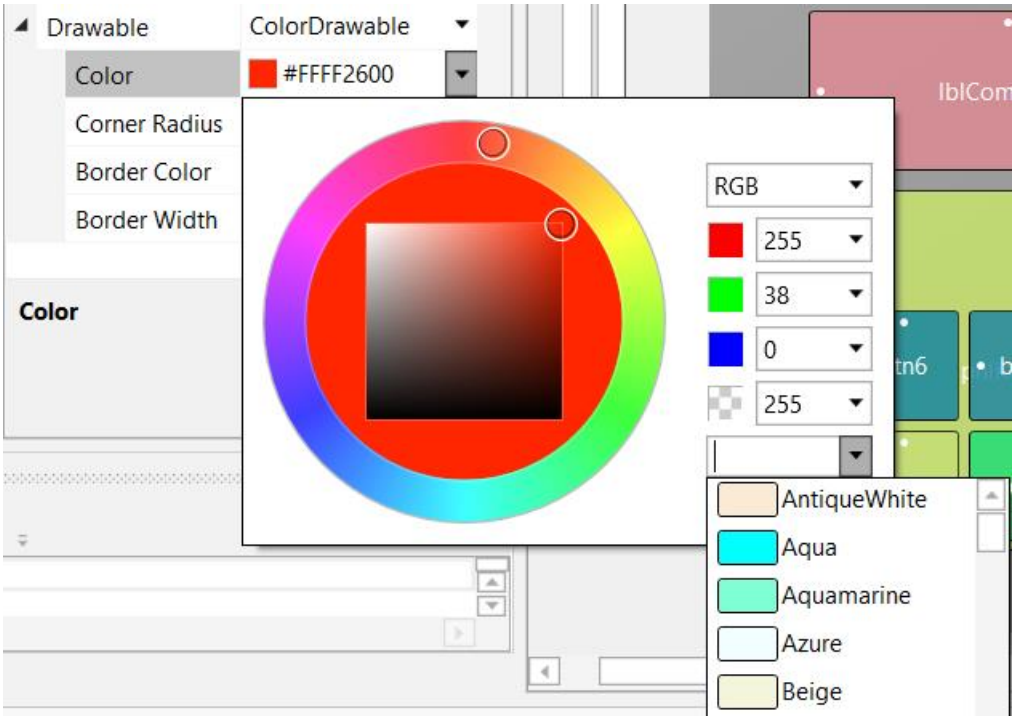
The color picker is displayed.



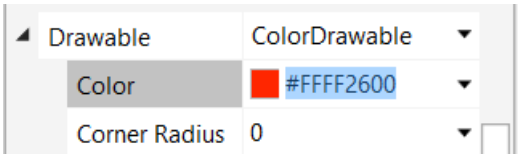
You can either:

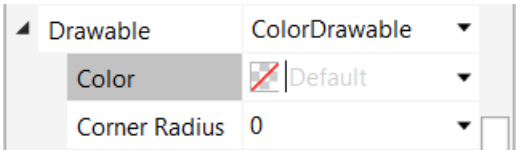
- Move the cursor in the outer circle to select a color.
- Move the cursor in the square to select the 'darkness'.
- Enter RGB or HSB values.
- Select a predefined color.

Select a predefined color.

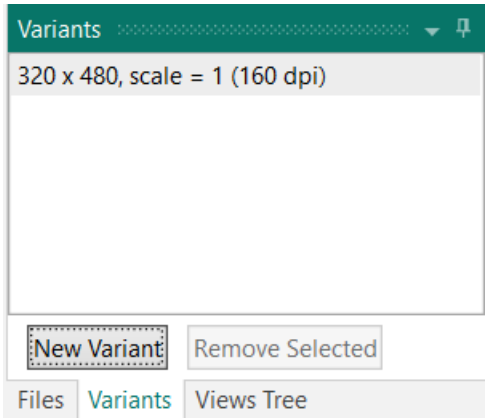


To reset the default color remove the hex color code.





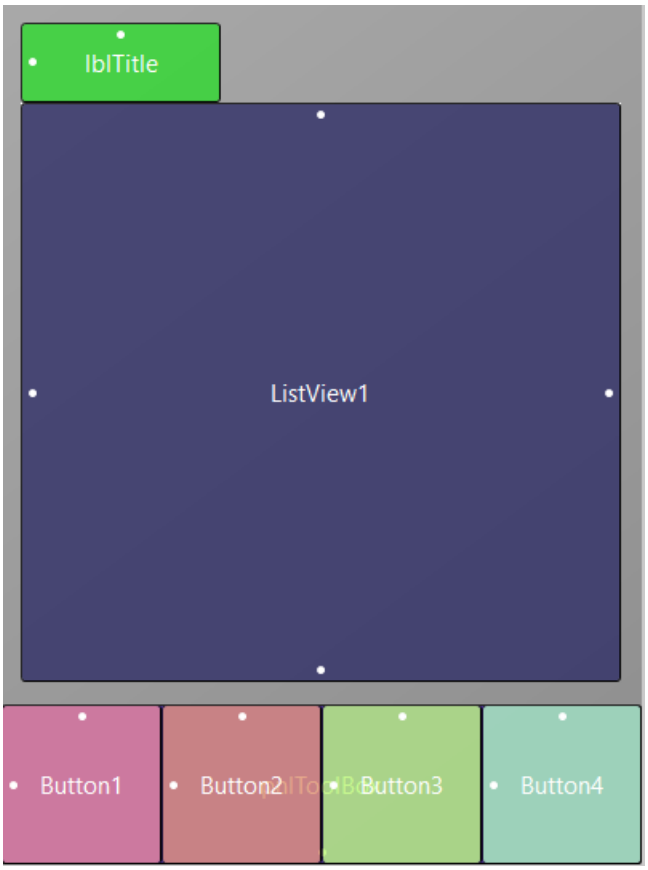
3.8 Layout variants

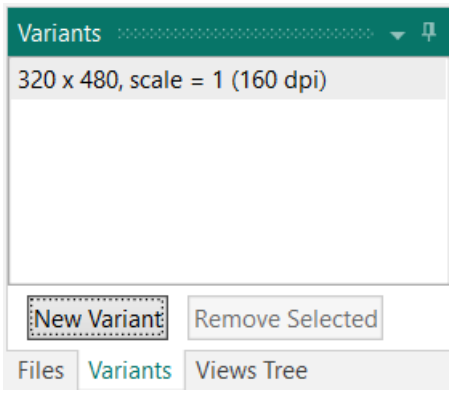


Different layout variants can be managed in a same layout file.

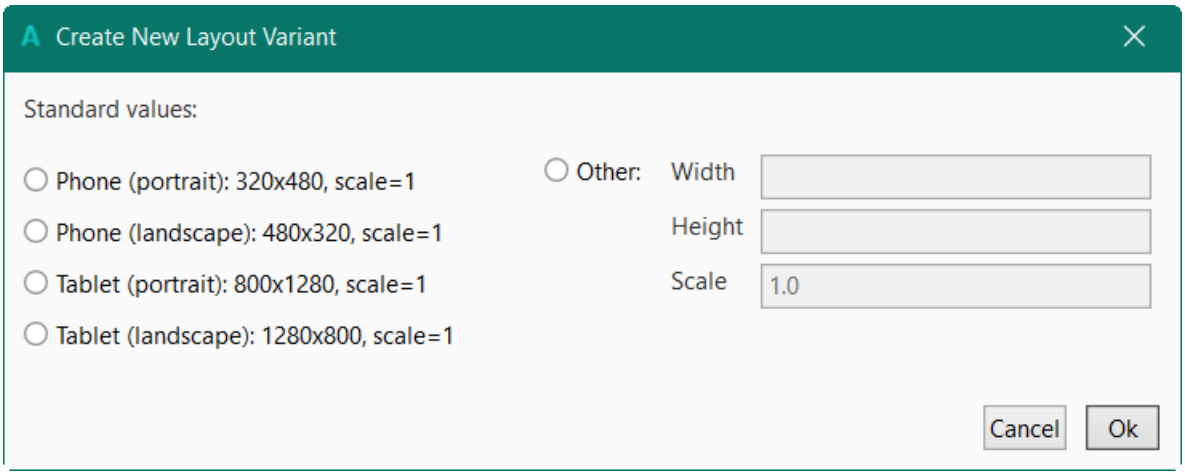
Let us make an example based on the TestLayoutsAnchors project
(which can be found under the GettingStarted\SourceCode\TestLayoutsAnchors directory):
- Create a new folder and name it TestLayoutVariants.
- Copy the whole contents of the TestLayoutsAnchors folder.
- Rename the TestLayoutAnchors.b4a file to TestLayoutVariants.b4a.
- Rename the TestLayoutAnchors.b4a.meta file to TestLayoutVariants.b4a.meta.
- Run the IDE.
- Run the Visual Designer.

The layout in the Abstract Designer should look like this.



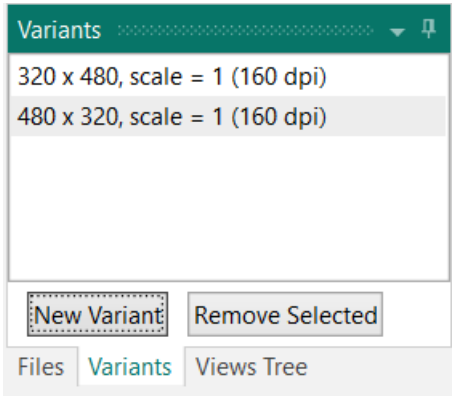


In the Designer, click on New Variant.



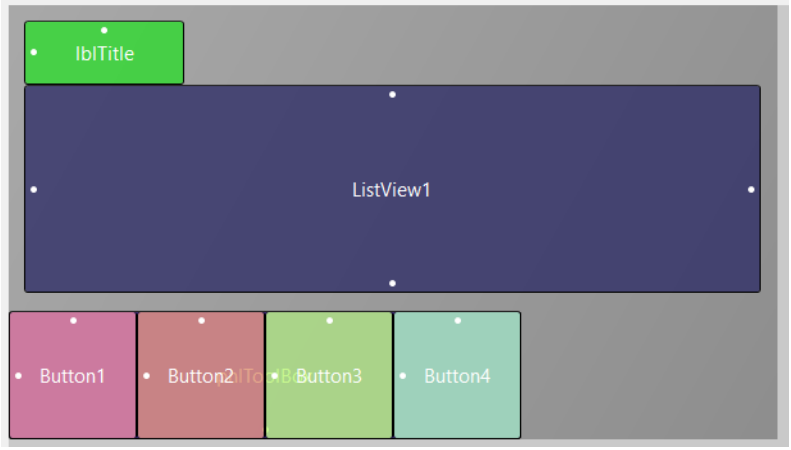
Select: Phone (landscape):480 x 320, scale = 1

Click on Ok.



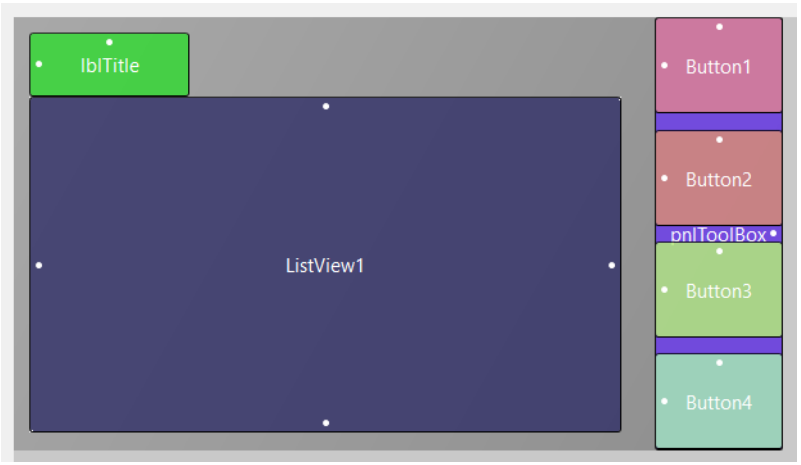
The new variant is added.

In the Abstract Designer you'll see something like this.



We see that the anchors work well.
pnlToolBox is still at the bottom of the screen and ListView1 is stretched the fill almost the whole screen width.

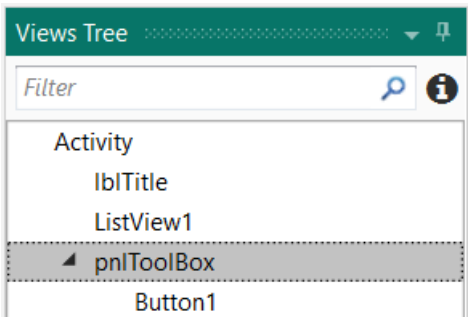
But for the landscape variant we want the ToolBox at the right side of the screen.



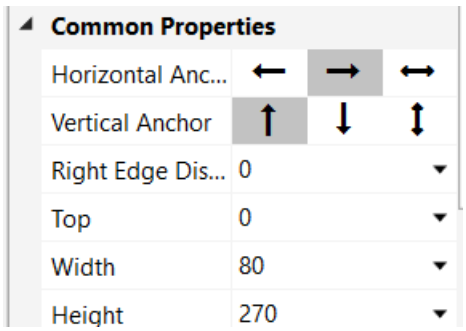
We:

- Reduce the width of ListView1 to get space for the ToolBox.

- Move pnlToolBox to the right side of the screen, change the Button heights and rearrange them vertically like in the picture.



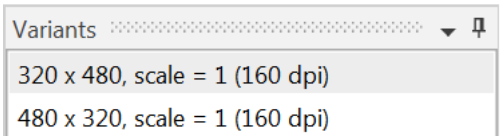
- Select pnlToolBox



- Set the pnlToolBox Horizontal Anchor to →.
  Set the Right Edge Distance to 0
  Set the pnlToolBox vertical anchor to TOP.
  Adjust the button Height (60) and Top properties accordingly.

To always show pnlToolBox in the middle of the screen we add following code in the Script – Variant window.

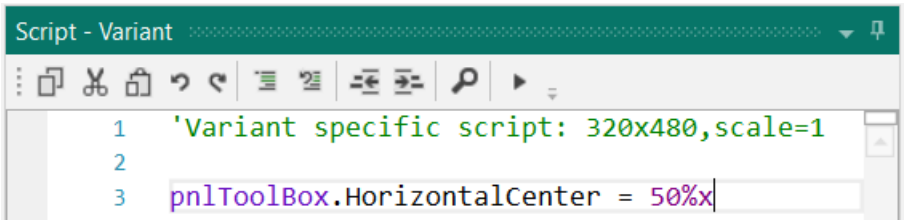
For portrait:



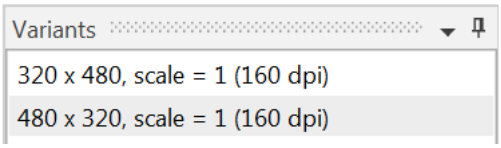
Select the portrait variant.

And add this code

```
pnlToolBox.HorizontalCenter = 50%x.
```



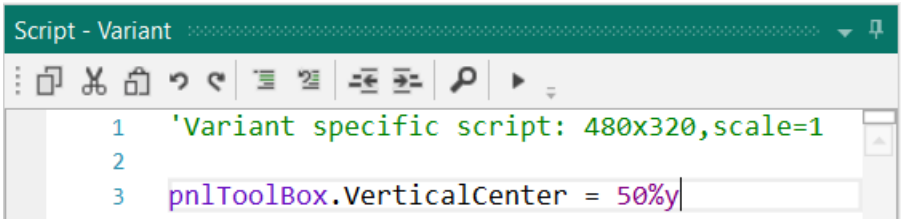
For landscape :

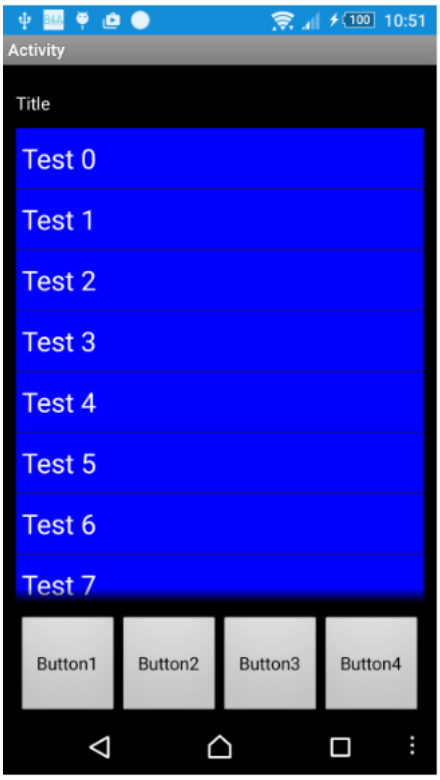


Select the landscape variant.

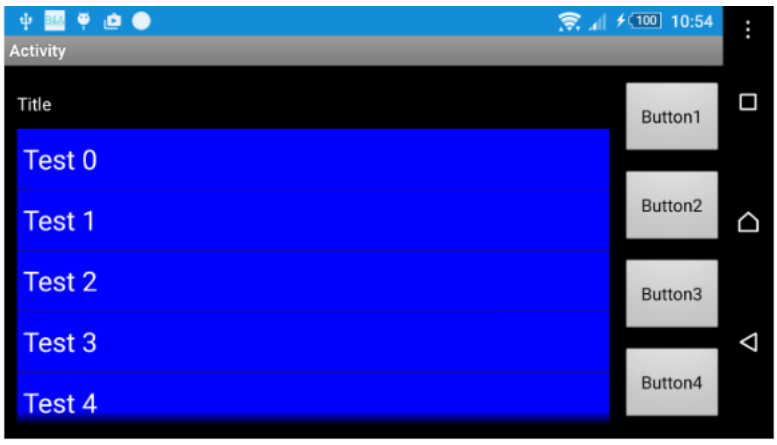
And add this code

```
pnlToolBox.VerticalCenter = 50%y.
```

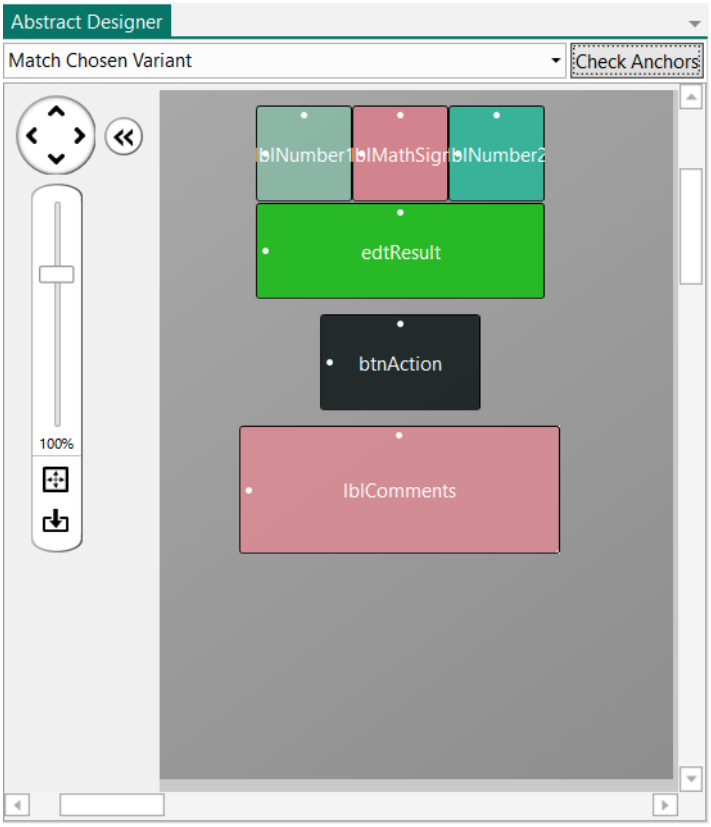




And the result on a device.



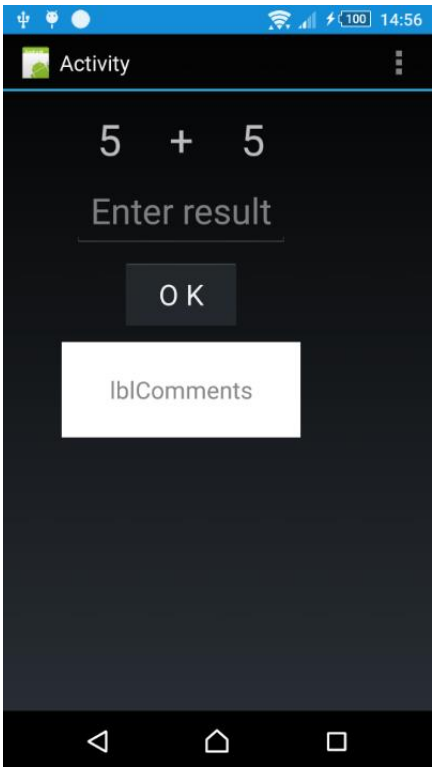
3.9 Abstract Designer



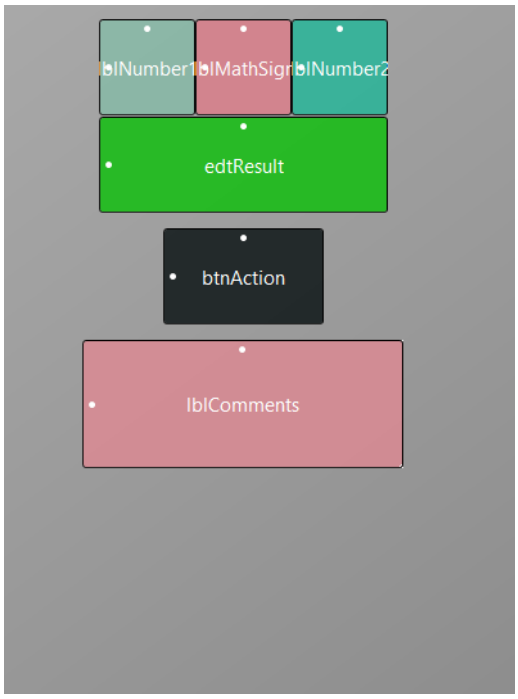
The Abstract Designer is a tool that shows the layout.
Its main purpose is to create different layout variants.

The different views are not shown with their exact shape but only as coloured rectangles.
Clicking on a view shows its properties in the Designer.

The images are made with the project MyFirstProgram.



Device

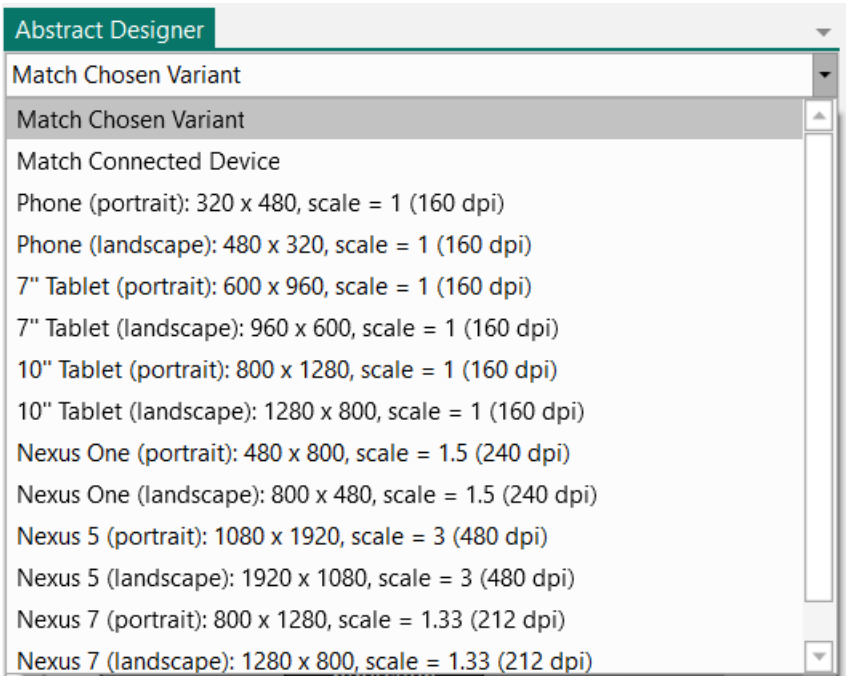


Abstract Designer

3.9.1 Selection of a screen size

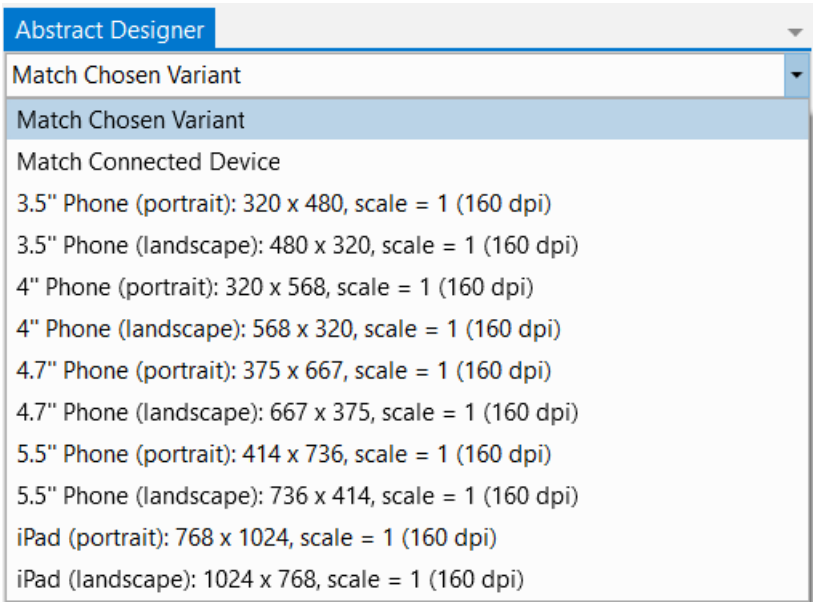
On top you can select different screen sizes:

3.9.1.1 B4A Selection of a screen size



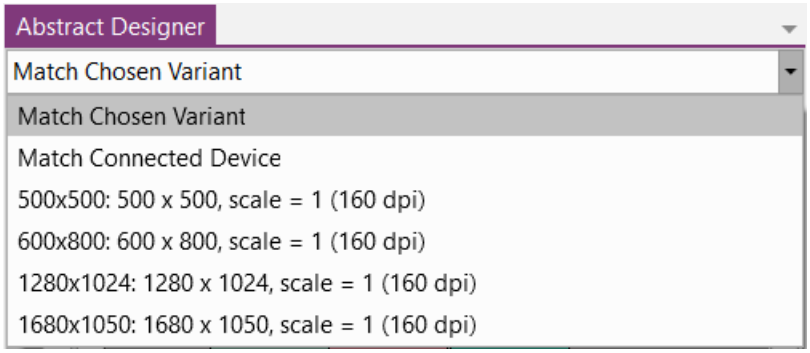
- Match chosen Variant.
Matches the variant selected in the Variant window.
- Match Connected Device.
Matches the size of the connected device or emulator.
- Different ‘standard’ sizes.
This allows see how a layout looks on s different screen.

3.9.1.2 B4i Selection of a screen size



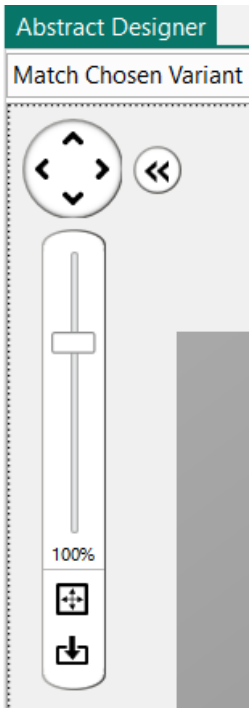
- Match chosen Variant.
Matches the variant selected in the Variant window.
- Match Connected Device.
Matches the size of the connected device or emulator.
- Different ‘standard’ sizes.
This allows see how a layout looks on s different screen.

3.9.1.3 B4J Selection of a screen size



- Match chosen Variant.
Matches the variant selected in the Variant window.
- Match Connected Device.
Matches the size of the connected device or emulator.
- Different ‘standard’ sizes.
This allows see how a layout looks on s different screen.

3.9.2 Zoom



With  you can move the virtual screen in the four directions.

With  you can hide the zoom cursor and show it again with  .

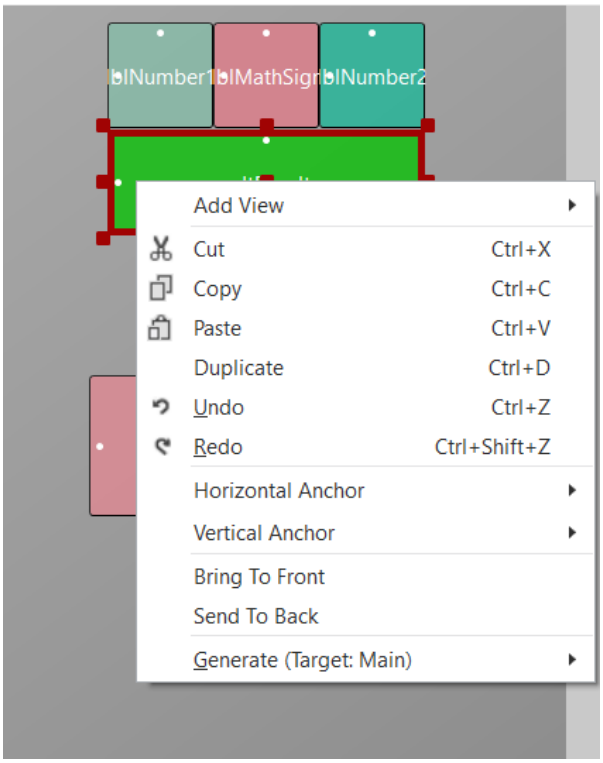
With the cursor you can set the zoom level you want.

With  you can zoom to fit the selected screen size.
With  you can reset the zoom back to 100%.

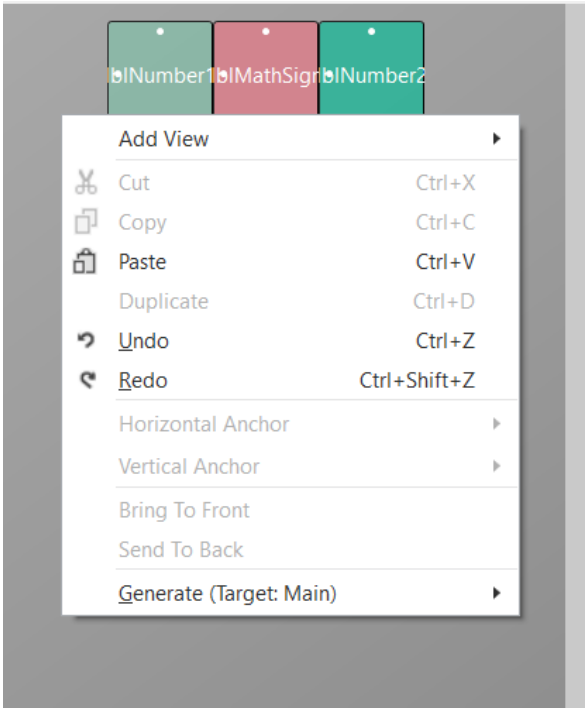
With the bottom and side cursors you can move the layout vertically or horizontally.

3.9.3 Context menus

Right clicking on a view shows a context menu.



Add View
Cut
Copy
Paste
Duplicate
Undo
Redo
Horizontal Anchor
Vertical Anchor
Bring To Front
Send To Back
Generate (Target: Main)
Target: reminds you the current module active in the IDE, the Main module in this case.



Right clicking somewhere on Main area shows the context menu with some functions disabled which are not relevant for an Activity.

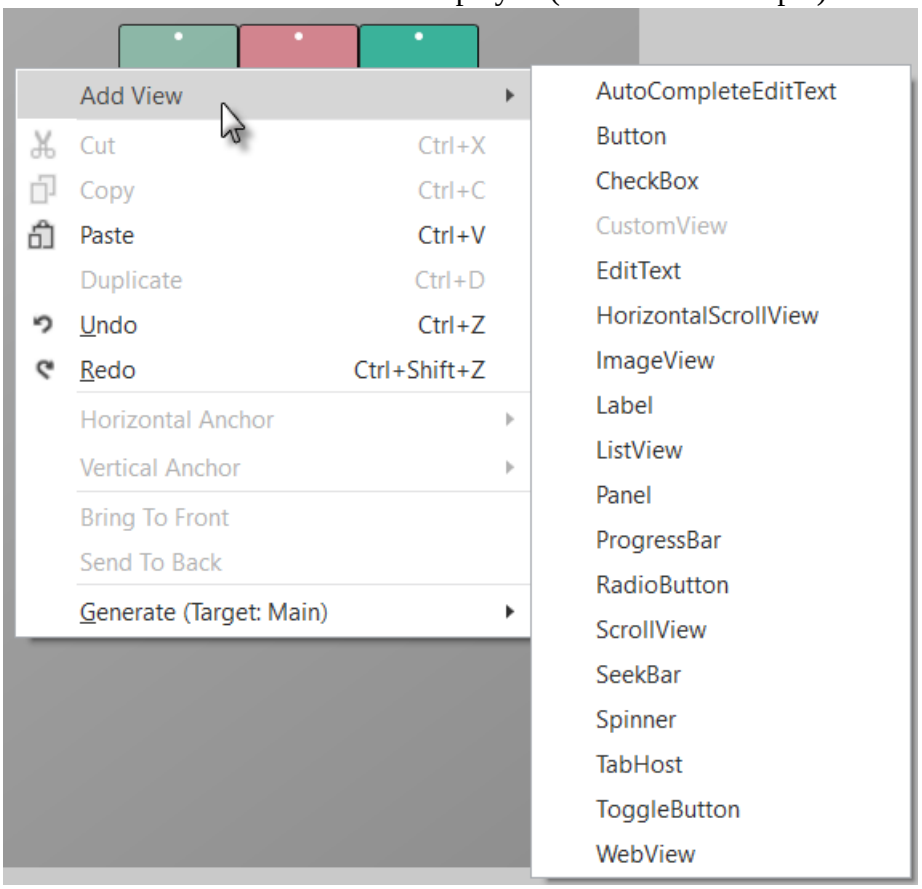
Only Add View, Paste, Undo, Redo and Generate are available.

3.9.3.1 Add View

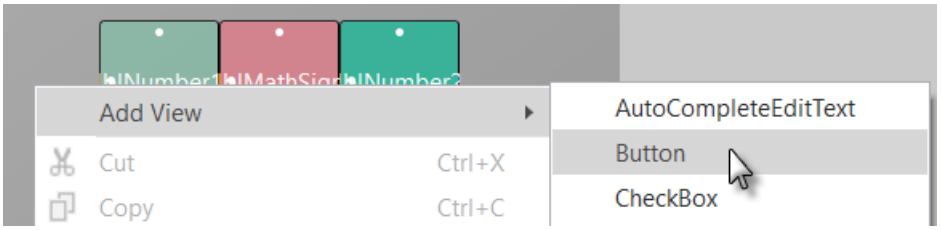
Right click somewhere and move the cursor onto Add View.

This function is the same as the Add View function in the Visual Designer menu.

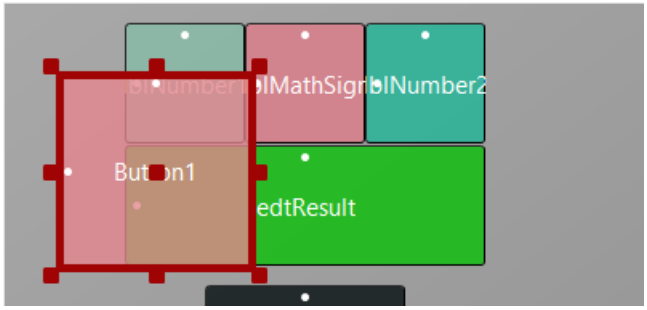
The list of all available views is displayed (B4A in the example).



Click on the desired view to add it.



Example for a Button.



The Button is added to the layout.

3.9.3.2 Cut

removes the selected view from the layout.
If you selected a Panel, it will be removed with all its child views!
If you cut it by accident, click on  or press Ctrl+Z to recover it.

3.9.3.3 Copy

copies the selected view into the clipboard.
If you selected a Panel, it will be copied with all its child views!

3.9.3.4 Paste

copies the content of the clipboard.
If you selected a Panel, it will be pasted with all its child views!

Before you paste a view, you must select where you want to paste it. This can be either onto the Activity or onto a Panel.

3.9.3.5 Duplicate

Duplicates the selected view, it is added over itself.
Duplicate is a shortcut of Copy and Paste.

If you selected a Panel, it will be duplicated with all its child views!

3.9.3.6 Undo / Redo

These two functions allow you to undo or redo the last operations.

3.9.3.7 Horizontal Anchor

You can set the horizontal anchor in the context menu instead of changing it in the Properties window.

The current anchor is checked.

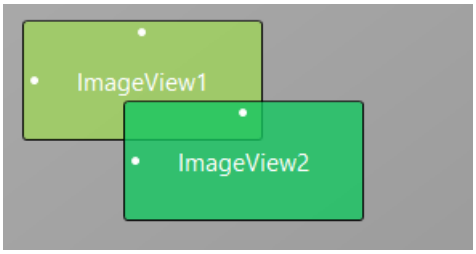
3.9.3.8 Vertical Anchor

You can set the vertical anchor in the context menu instead of changing it in the Properties window.

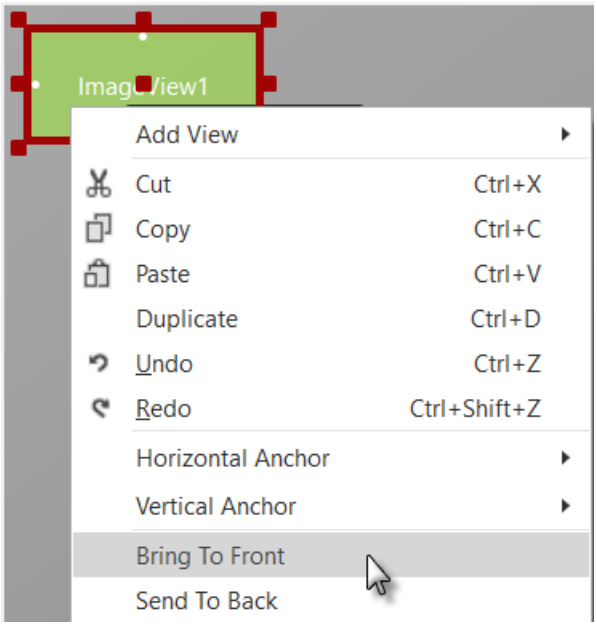
The current anchor is checked.

3.9.3.9 Bring To Front

Bring To Front Moves the selected View on top of the layout.

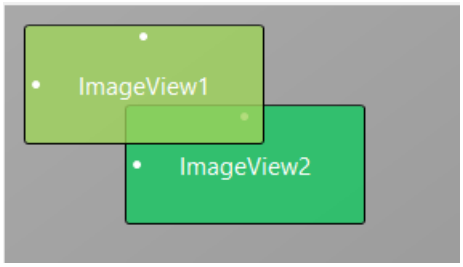


In the picture ImageView2 is over ImageView1.
You see it with the border color.

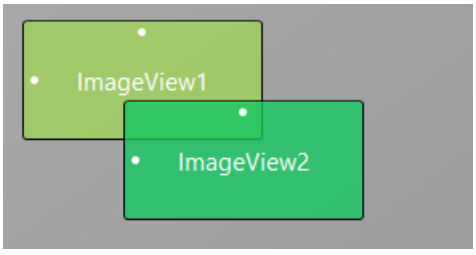


Right click on ImageView1 and click on Bring To Front to move ImageView1 to front of all other views.

And the result:



3.9.3.10 Send To Back



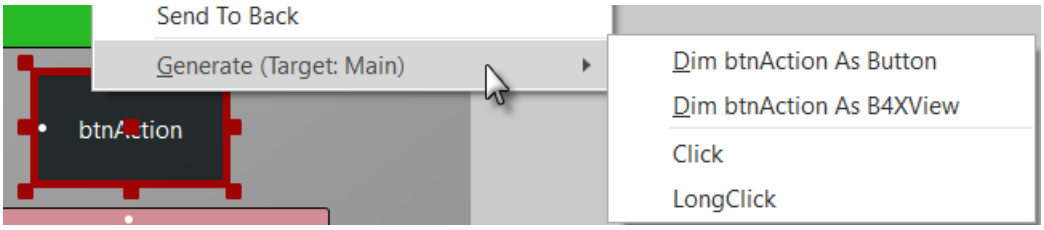
Send To Back is the inverse function of Bring To Front function above.

3.9.3.11 Generate

Generates the declaration statement or an event routine for the selected View. It is a shortcut of the Generate Members function in the VisualDesigner Tools menu but only for the selected view.

A popup menu allows you to select what code you want to generate, the possibilities depend on the type of the selected view.

Example with a Button (B4A):



Dim btnAction As Button
Generates the declaration statement in the Globals routine.

```
Private btnAction As Button
```

Dim btnAction As B4XView
Generates the declaration statement in the Globals routine.

```
Private btnAction As B4XView
```

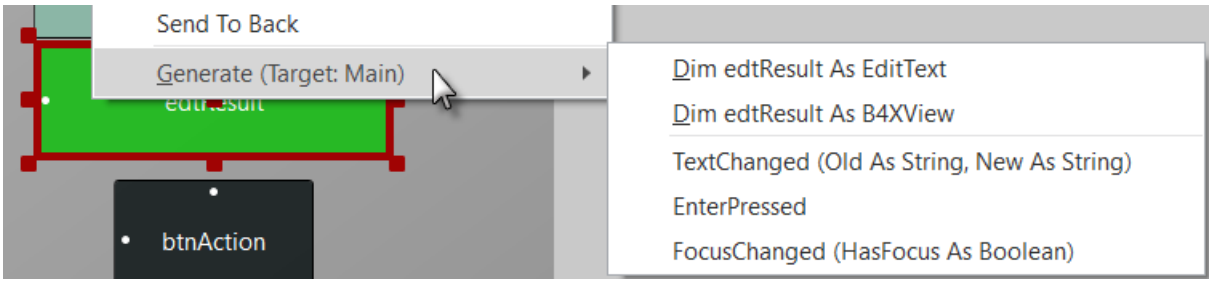
Click
Generates the Click event routine frame.

```
Sub btnAction_Click

End Sub
```

LongClick
Generates the LongClick event routine frame.

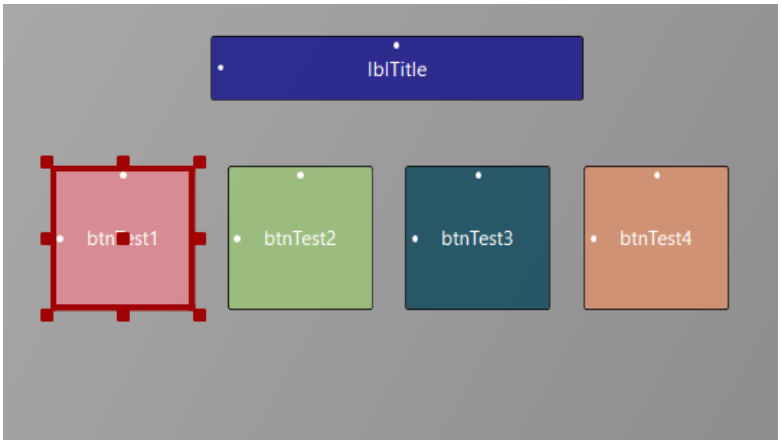
```
Sub btnTest1_LongClick

End Sub
```

Example with an EditText (B4A):

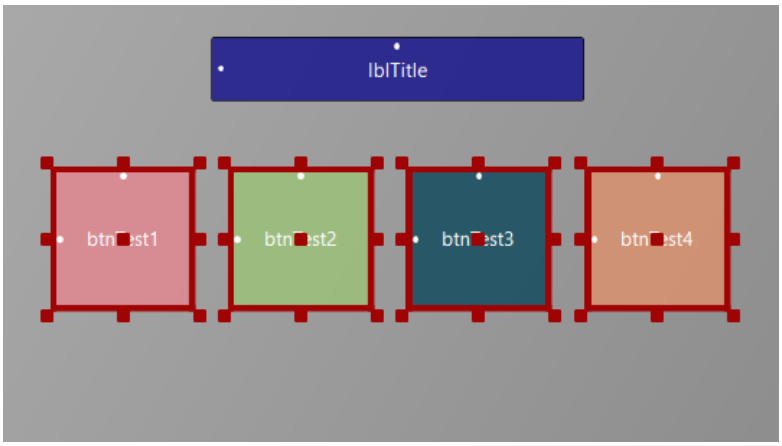


The context menu depends on the type of the product (B4A, B4i, B4J) and the type of the view.

3.9.4 Select views

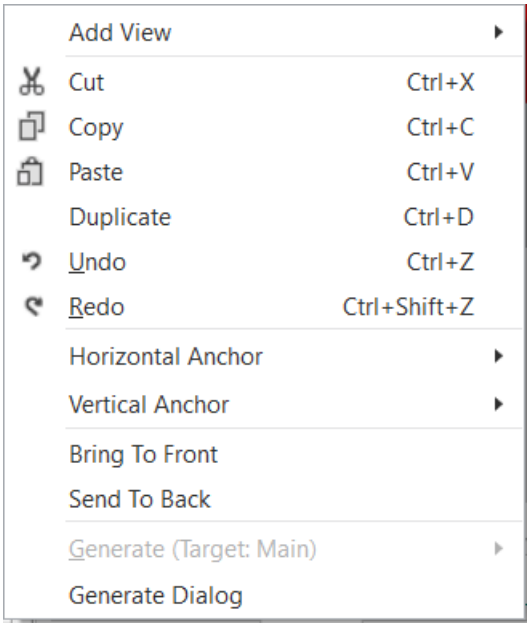


Select a single view:
Click on the view



Select several views:
Click on the first view.
Press the Ctrl key,
Click the following views.

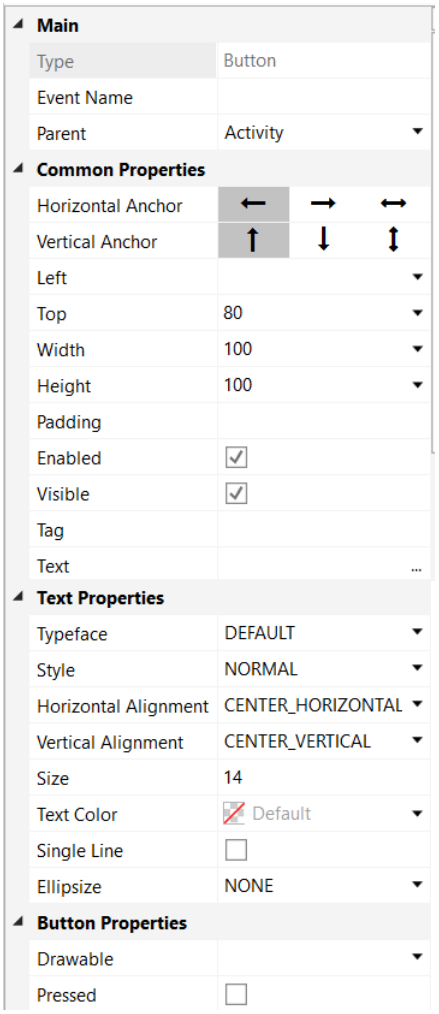
The selected views are highlighted.



After the selection you can:

- Move the selected views with the arrow keys of the keyboard in the four directions.
- Right click on one of the selected view to show the context menu.

The functions are the same as for a single view, but a new function, **GenerateDialog**, is available to Generate Members.
This is the same function as in the Visual Designer Tools menu but only for the selected views.

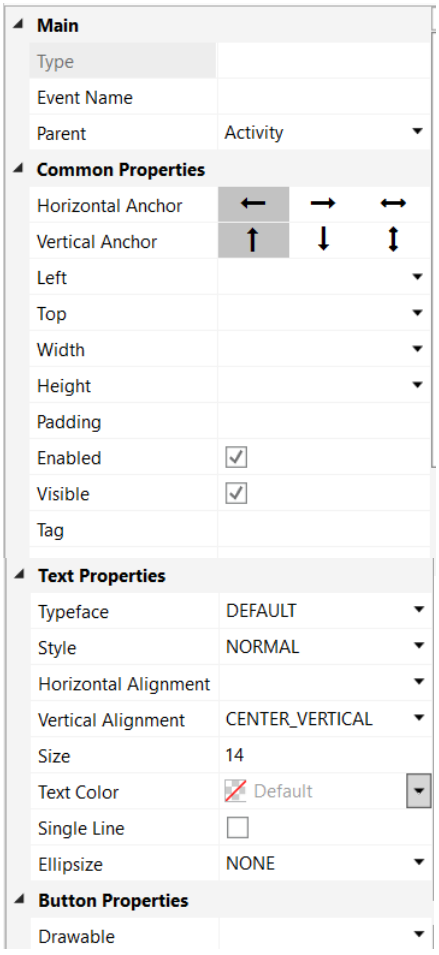


•In the Properties window you can change all properties common to the selected views.

You can change the parent view.

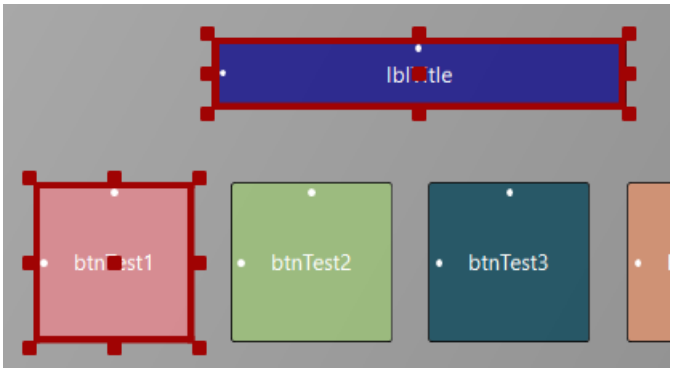
You can change all these properties because they are the same for the four views selected in the example.

Changing, for example, the Height property will change it for all the selected views.



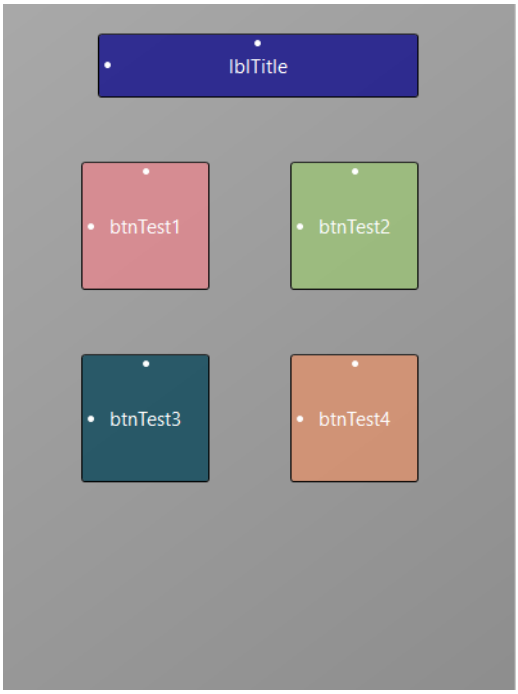
If you select views of different types, only the properties common to the selected views can be changed.

Example with a Label and a Button.



3.9.5 Example

Let us take a simple example with a layout in portrait mode, like the image below.
This example project is in the *Getting Started\SourceCode* folder in the *AbstractDesigner* subfolder.
The example is for B4A, but the principle is the same for B4i and B4J.

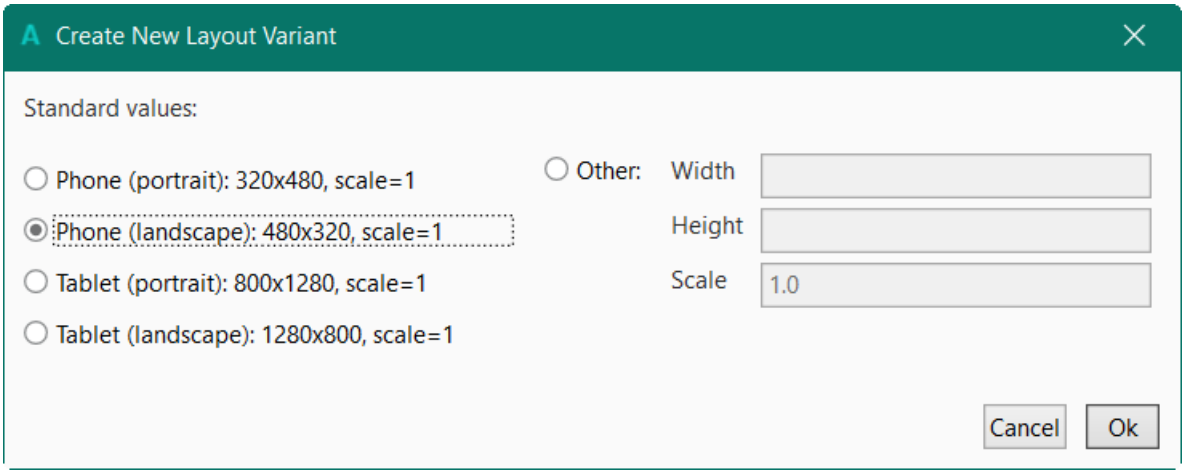


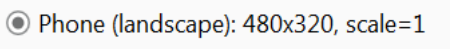
Now we would like to make a landscape variant.

In the Variant window click on 

.

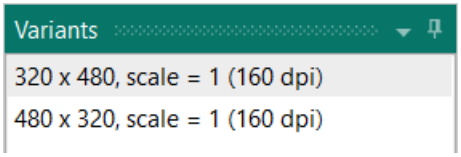
A selection window is displayed.



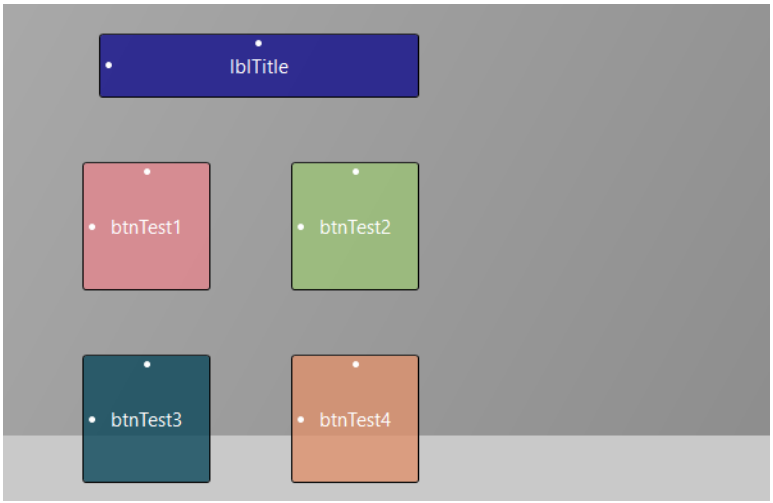
Select 

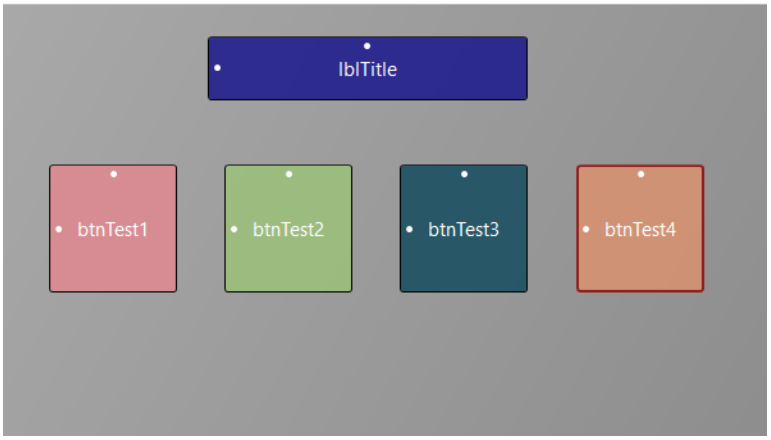
.

In the Variant window the new variant is displayed.

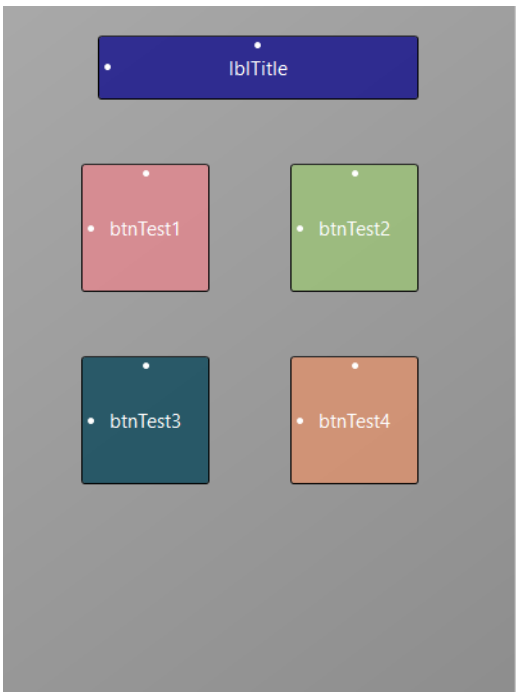


The Abstract Designer looks now like this:

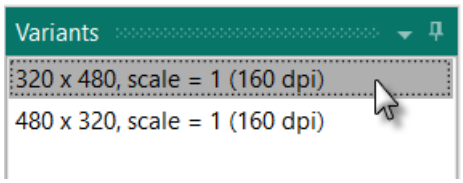




Now we rearrange the views to fit the new orientation.



If you select in the Variant window the previous variant



you will see that layout.

3.10 Copy layouts cross platform between B4A, B4i and B4J

You can copy layouts between the three products B4A, B4i and B4J.
Simply, select the views to copy and paste them where ever you want.

This allows you to build the layout on one platform and later copy all of it or parts of it to other platforms.

Each platform has unique controls. These cannot be copied. The same is true for unique properties that are not available in other platforms.
Custom views, and especially custom views that are built over XUI and are intended to be cross platform (example: XUI Views), can be shared with all there properties.

It is also possible to declare views as B4XViews directly from the designer. Note that custom views do not need to be declared as B4XViews. They should be declared with the custom type, which is cross platform.
Tip: you can always cast B4XViews to the explicit type and vice versa.

Instead of showing images here, it's more efficient to watch the demo video in the forum.

A concrete example is explained in the B4XPages Cross-platform projects booklet.

3.11 Adding views by code

It is also possible to add views by code instead of using the Designer with a device or the Abstract Designer.

- Advantage: You have full control of the view.

- Disadvantage: You must define almost everything and Anchors and AutoScale don't work.

Note that you should avoid adding views in code but use the Designer with Anchors and Designer Scripts.

3.11.1 B4A Adding views by code

The source code is in the source code directory: B4A\AddViewsByCode

For the positions and dimensions of the views on the screen two special options are available:

- dip **d**ensity **i**ndependent **p**ixels.
  100dip = DipToCurrent(100) `DipToCurrent` is a Keyword `dip` is the Shortcut
  100dip = 100 / 160 * device density
  The default density is 160 dpi **d**ots **p**er **i**nch (pixels per inch)
  Densities in Android:
  - 120 scale 0.75
  - 160 scale 1 default
  - 240 scale 1.5
  - 320 scale 2

- %x and %y represent distances proportional to the active screen width and height.
  20%x = 0.2 * Activity.Width
  90%y = 0.9 * Activity.Height
  20%x = PerXToCurrent(20) `PerXToCurrent` is a Keyword `%x` is the Shortcut
  90%y = PerYToCurrent(90)

Example:
Let us put a Label on top of the screen and a Panel below it with a Label and a Button on it:

```
Sub Globals
   Private lblTitle, lblPanelTitle As Label
   Private pnlTest As Panel
   Private btnTest As Button
End Sub
```

```
Sub Activity_Create(FirstTime As Boolean)
   lblTitle.Initialize("")
   lblTitle.Color = Colors.Red
   lblTitle.TextSize = 20
   lblTitle.TextColor = Colors.Blue
   lblTitle.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL
   lblTitle.Text = "Title"
   Activity.AddView(lblTitle, 20%x, 10dip, 60%x, 30dip)

   pnlTest.Initialize("")
   pnlTest.Color = Colors.Blue

   btnTest.Initialize("btnTest")
   btnTest.Text = "Test"

   lblPanelTitle.Initialize("")
   lblPanelTitle.Color = Colors.Red
   lblPanelTitle.TextSize = 16
   lblPanelTitle.TextColor = Colors.Blue
   lblPanelTitle.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL
   lblPanelTitle.Text = "Panel test"

   Activity.AddView(pnlTest, 0, lblTitle.Top+lblTitle.Height+10dip, 100%x, 50%y)
   pnlTest.AddView(lblPanelTitle, 20dip, 10dip, 100dip, 30dip)
   pnlTest.AddView(btnTest, 50dip, 50dip, 100dip, 60dip)
End Sub
```

Declaring the views.

```
Private lblTitle, lblPanelTitle As Label
Private pnlTest As Panel
Private btnTest As Button
```

Initializing the title label:

`lblTitle.Initialize("")`	Initializes the Label, no EventName required.
`lblTitle.Color = Colors.Red`	Sets the Background color to red.
`lblTitle.TextSize = 20`	Sets the text size to 20.
`lblTitle.TextColor = Colors.Blue`	Sets the text color to blue.
`lblTitle.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL`	Sets the label gravity.
`lblTitle.Text = "Title"`	Sets the label text to 'Title'.
`Activity.AddView(lblTitle, 20%x, 10dip, 60%x, 30dip)`	Adds the view to the activity.

If the Label had been added in the Designer, all the above code wouldn't have been necessary because the properties would already have been defined in the Designer.
In the `Activity.AddView` line we see that:

- the Left property is set to 20%x, 20% of Activity.Width.
- the Top property is set to 10dip, 10 density independent pixels.
- the Width property is set to 60%x, 60% of Activity.Width
- the Height property is set to 30dip, 30 density independent pixels.

`pnlTest.Initialize("")`	Initializes the Panel, no EventName required.
`pnlTest.Color = Colors.Blue`	Sets the Background color to blue.
`btnTest.Initialize("btnTest")`	Initializes the Button, EventName = btnTest.
`btnTest.Text = "Test"`	Sets the button text to "Test"
`lblPanelTitle.Initialize("")`	

```
lblPanelTitle.Color = Colors.Red
lblPanelTitle.TextSize = 16
lblPanelTitle.TextColor = Colors.Blue
lblPanelTitle.Gravity = Gravity.CENTER_HORIZONTAL + Gravity.CENTER_VERTICAL
lblPanelTitle.Text = "Panel test"
```

Similar to the title Label.

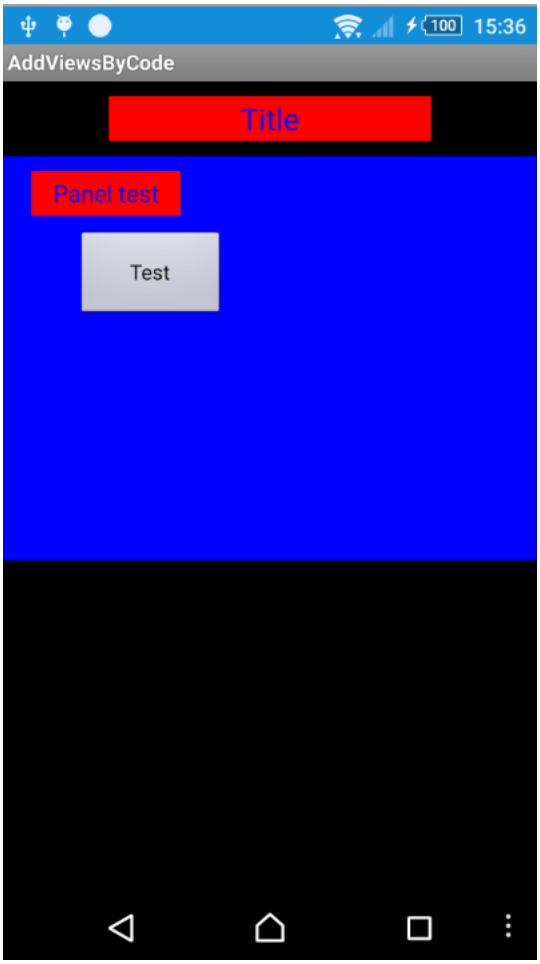
```
Activity.AddView(pnlTest,0,lblTitle.Top + lblTitle.Height + 10dip, 100%x, 50%y)
```
Adds the Panel pnlTest to the Activity.

- the Left property is set to 0
- the Top property is set to 10dips below the title Label
- the Width property is set to 100%x, the total Activity.Width
- the Height property is set to 50%y, half the Activity.Height

```
pnlTest.AddView(lblPanelTitle, 20dip, 10dip, 100dip, 30dip)
```
Adds the Label lblPanelTitle to the Panel pnlTest at the given position and with the given dimensions in dips.

```
pnlTest.AddView(btnTest, 50dip, 50dip, 100dip, 60dip)
```
Adds the Button btnTest to the Panel pnlTest at the given position and with the given dimensions in dips.

And the result on the device:



3.11.2 B4i Adding views by code

The source code is in the source code directory: B4i\AddViewsByCode

For the positions and dimensions of the views on the screen two options are available:

- Points, scale independent.
  The default density is 160 dpi **d**ots **p**er **i**nch (pixels per inch).
  All coordinates refer to this density, iOS adapts the values internally to the scale.
  No *dip* values like in Android. More details in Coordinates.

- %x and %y represent distances proportional to the current screen width and height.
  20%x = 0.2 * Page1.RootPanel.Width
  90%y = 0.9 * Page1.RootPanel.Height
  20%x = PerXToCurrent(20) `PerXToCurrent` is a Keyword `%x` is the Shortcut
  90%y = PerYToCurrent(90)

Example:

Let us put a Label on top of the screen and a Panel below it with a Label and a Button on it:

The whole code.

```
Sub Process_Globals
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page

   Private lblTitle, lblPanelTitle As Label
   Private pnlTest As Panel
   Private btnTest As Button
End Sub

Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   NavControl.ShowPage(Page1)

   lblTitle.Initialize("")
   lblTitle.Color = Colors.Red
   lblTitle.Font = Font.CreateNew(20)
   lblTitle.TextColor = Colors.Blue
   lblTitle.TextAlignment =lblTitle.ALIGNMENT_CENTER
   lblTitle.Text = "Title"

   pnlTest.Initialize("")
   pnlTest.Color = Colors.LightGray

   btnTest.InitializeCustom("btnTest", Colors.Black, Colors.Blue)
   btnTest.SetBorder(1, Colors.Black, 5)
   btnTest.Text = "Test"

   lblPanelTitle.Initialize("")
   lblPanelTitle.Color = Colors.Red
   lblPanelTitle.Font = Font.CreateNew(16)
   lblPanelTitle.TextColor = Colors.Blue
   lblPanelTitle.TextAlignment = lblPanelTitle.ALIGNMENT_CENTER
   lblPanelTitle.Text = "Panel test"
End Sub

Private Sub Page1_Resize(Width As Int, Height As Int)
   Page1.RootPanel.AddView(lblTitle, 20%x, 10, 60%x, 30)
   Page1.RootPanel.AddView(pnlTest, 10%x, lblTitle.Top + lblTitle.Height + 10, 80%x,
30%y)
   pnlTest.AddView(lblPanelTitle, 20, 10, 100, 30)
   pnlTest.AddView(btnTest, 50, 50, 100, 60)
End Sub
```

Code explanations:

Declaring the views in Process_Globals.

```
Private lblTitle, lblPanelTitle As Label
Private pnlTest As Panel
Private btnTest As Button
```

Initializing the different views in Application_Start:

`lblTitle.Initialize("")`	Initializes the Label, no EventName required.
`lblTitle.Color = Colors.Red`	Sets the Background color to red.
`lblTitle.Font = Font.CreateNew(20)`	Sets the text size to 20.
`lblTitle.TextColor = Colors.Blue`	Sets the text color to blue.
`lblPanelTitle.TextAlignment = lblPanelTitle.ALIGNMENT_CENTER`	Sets the label text alignment to ‘CENTER’.
`lblTitle.Text = "Title"`	Sets the label text to 'Title'.

If the Label had been added in the Designer, all the above code wouldn't have been necessary because the properties would already have been defined in the Designer.

`pnlTest.Initialize("")`	Initializes the Panel, no EventName required.
`pnlTest.Color = Colors.LightGray`	Sets the Background color to light gray.

`btnTest.InitializeCustom("btnTest", Colors.Black, Colors.Blue)`
Initializes the Button, EventName = btnTest, TextColor, PressedTextColor.

`btnTest.SetBorder(1, Colors.Black, 5)`	Sets a Border with the given Width, Color and CornerRadius.
`btnTest.Text = "Test"`	Sets the button text to "Test".

```
lblPanelTitle.Initialize("")
lblPanelTitle.Color = Colors.Red
lblPanelTitle.Font = Font.CreateNew(16)
lblPanelTitle.TextColor = Colors.Blue
lblPanelTitle.TextAlignment = lblPanelTitle.ALIGNMENT_CENTER
lblPanelTitle.Text = "Panel test"
```

Similar to the title Label.

Note that we add the views to their parent views in Page1_Resize and not in Application_Start because the real size of Page1.RootPanel is not known before!

```
Private Sub Page1_Resize(Width As Int, Height As Int)
  Page1.RootPanel.AddView(lblTitle, 20%x, 10, 60%x, 30)
```

Adds the view to the Page1.RootPanel.
In the `Page1.RootPanel.AddView` line we set:

- the Left property to 20%x, 20% of Page1.RootPanel.Width,
- the Top property to 10, 10 points independent of the device scale,
- the Width property to 60%x, 60% of Page1.RootPanel.Width,
- the Height property to 30, 30 points independent of the device scale.

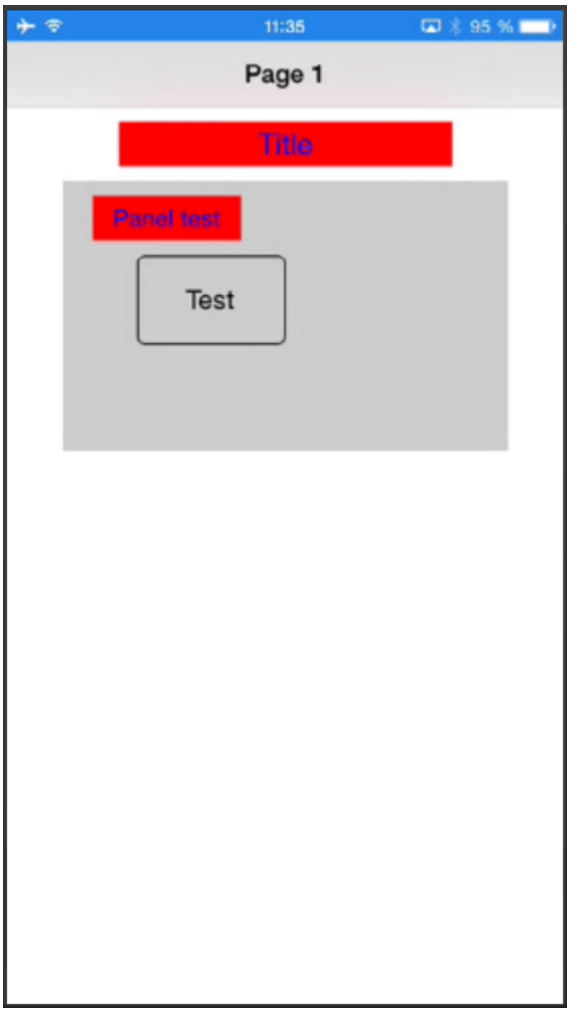
```
Page1.RootPanel.AddView(pnlTest, 10%x, lblTitle.Top + lblTitle.Height + 10, 80%x, 30%y)
```
Adds the Panel pnlTest to the Page1.RootPanel.

- the Left property is set to 0
- the Top property is set to 10 points below the title Label
- the Width property is set to 100%x, the total Page1.RootPanel.Width
- the Height property is set to 30%y, 30% of the Page1.RootPanel.Height

```
pnlTest.AddView(lblPanelTitle, 20, 10, 100, 30)
```
Adds Label lblPanelTitle to Panel pnlTest at the given position and with the given dimensions in points.

```
pnlTest.AddView(btnTest, 50, 50, 100, 60)
```
Adds Button btnTest to Panel pnlTest at the given position and with the given dimensions in points.

And the result:



3.11.3 B4J Adding views by code

The source code is in the source code directory: B4J\AddViewsByCode

In B4J there are no dip, %x nor %y values, only pixel values.

If you want use %x and %y values, you must do it in the MainForm_Resize event routine.
The MainForm_Resize event is also raised when the program starts, just after AppStart.

Example:

Let us put a Label on top of the form, a Pane below it with a Label on it and a Pane at the bottom of the form with three Buttons.

Code explanations:

Declaring the nodes in Process_Globals.

```
Sub Process_Globals
   Private fx As JFX
   Private MainForm As Form

   Private lblTitle, lblPanelTitle As Label
   Private pnlTest, pnlTools As Pane
   Private btnCut, btnCopy, btnPaste As Button
End Sub
```

Initializing and adding the different nodes in AppStart:

`MainForm.Title = "Add nodes by code"` Sets the Form title
`MainForm.Resizable = True` Sets the Form being resizable.
`MainForm.SetWindowSizeLimits(310, 300, 1200, 800)` Sets the min and max form sizes.

`lblTitle.Initialize("")` Initializes the Label, no EventName required.
`CSSUtils.SetBackgroundColor(lblTitle, fx.Colors.Red)` Sets the Background color to red.
`lblTitle.TextSize = 20` Sets the text size to 20.
`lblTitle.TextColor = fx.Colors.White` Sets the text color to white.
`lblTitle.Alignment = "CENTER"` Sets the label text alignment to ‘CENTER’.
`lblTitle.Text = "Title"` Sets the label text to 'Title'.

If the Label had been added in the Designer, all the above code wouldn't have been necessary because the properties would already have been defined in the Designer.

`pnlTest.Initialize("")` Initializes the Panel, no EventName required.
`CSSUtils.SetBackgroundColor(pnlTest, fx.Colors. RGB(240, 255, 240))`
Sets the Background color to white.
`CSSUtils.SetBorder(pnlTest, 1, fx.Colors.Black, 0)`
Sets the Border color to black and the corner radius to 0.

The rest of the initialization code is like the code above.

We add the nodes. In B4J Views are called Nodes, therefore AddNode instead of AddView.
We add `lblTitle`, `pnlTest` and `pnlTools` to the `MainForm.RootPane`.

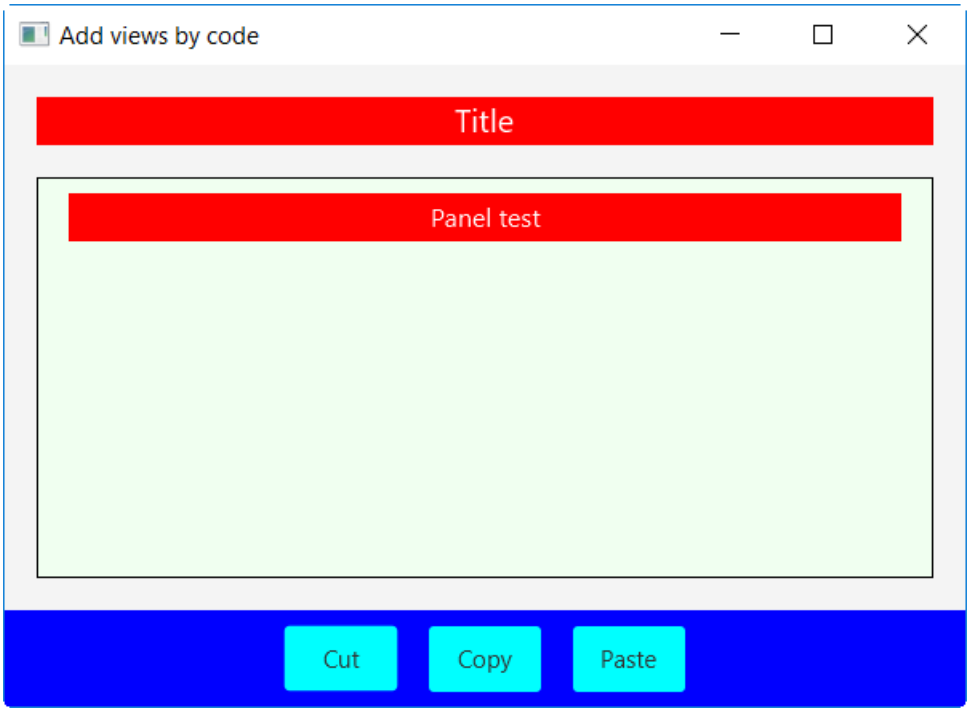
```
MainForm.RootPane.AddNode(lblTitle, 0.1 * 600, 10, 0.8 * 400, 30)
MainForm.RootPane.AddNode(pnlTest, 0.1 * 600, lblTitle.Top + lblTitle.Height + 10, 0.8
* 600, 0.8 * 400)
```

And we add `lblPanelTitle` to `pnlTest and the buttons to pnlTools`.

```
MainForm.RootPane.AddNode(lblTitle, 20, 20, 400, 30)
MainForm.RootPane.AddNode(pnlTest, 20, 20, 50, 50)
MainForm.RootPane.AddNode(pnlTools, 10, 10, 50, 60)
pnlTest.AddNode(lblPanelTitle, 20, 10, 100, 30)
pnlTools.AddNode(btnCut, 50, 10, 70, 40)
pnlTools.AddNode(btnCopy, 50, 10, 70, 40)
pnlTools.AddNode(btnPaste, 50, 10, 70, 40)
```

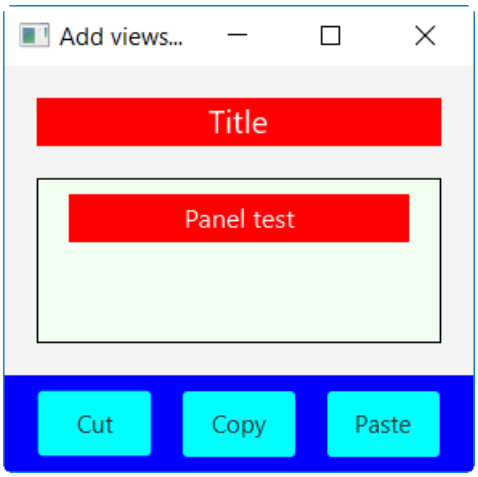
In the `Private Sub MainForm_Resize` routine we resize the nodes.

```
Private Sub MainForm_Resize (Width As Double, Height As Double)
   lblTitle.Left = 20
   lblTitle.SetSize(Width - 40, 30)

   pnlTools.Left = 0
   pnlTools.SetSize(Width, 60)
   pnlTools.Top = Height - pnlTools.Height

   pnlTest.Left = lblTitle.Left
   pnlTest.Top = lblTitle.Top + lblTitle.Height + 20
   pnlTest.SetSize(Width - 40, pnlTools.Top - lblTitle.Height - 60)

   lblPanelTitle.Left = 20
   lblPanelTitle.SetSize(Width - 80, 30)

   btnCopy.Left = (Width - btnCopy.Width) / 2
   btnCut.Left = btnCopy.Left - btnCut.Width - 20
   btnPaste.Left = btnCopy.Left + btnCut.Width + 20
End Sub
```



And the result:

The positions and sizes are adjusted to the form size.



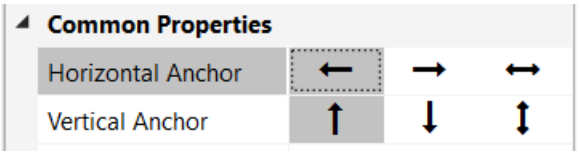
Or resized to the minimum size.

The same can be better done in the Designer with anchors and Designer Scripts !
See next chapters.

3.12 Anchors

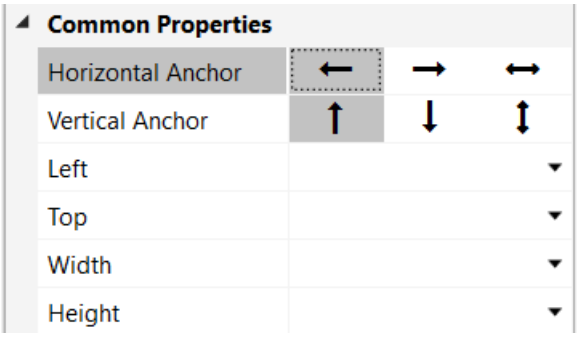
The Horizontal Anchor and Vertical Anchor properties are very powerful to adapt to different screen sizes.

3.12.1 Horizontal Anchor



The horizontal anchor property can take three values:

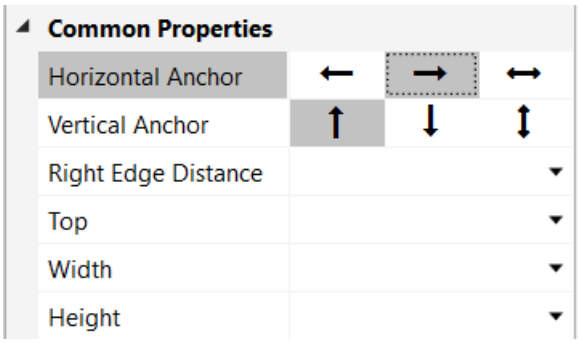
-  LEFT



This is the default value.
The left edge is anchored to the left edge of the parent view with the distance given in the Left property.

Left and Top anchors are shown.

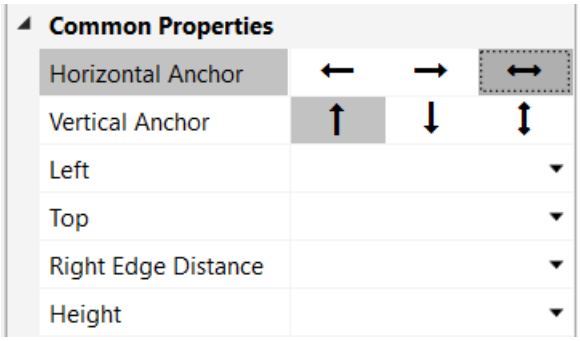
-  RIGHT



The right edge is anchored to the right edge of the parent view with the distance given in the Right Edge Distance property.
The Left property is no longer available because it is defined by the width and the right anchor !

The dot on top and on the right edge show the anchors.

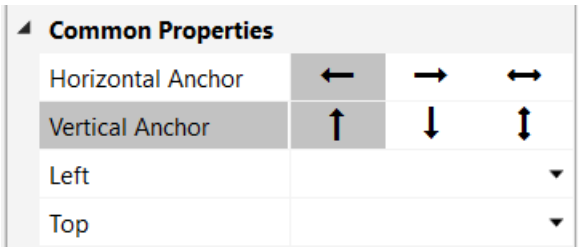
-  BOTH



Both edges are anchored.
The Width property is no longer available because it is defined by the anchors !
Setting the Horizontal Anchor property to BOTH is like using the SetLeftAndRight function in the Designer Scripts.

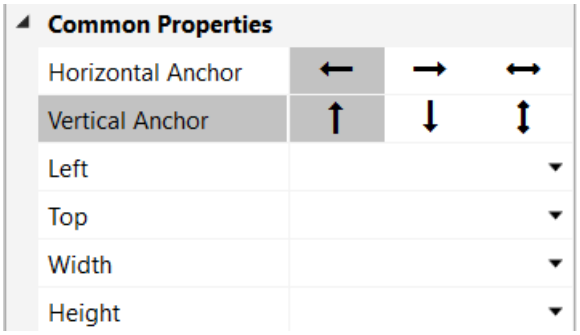
The dots on top and on the two edges show the anchors.

3.12.2 Vertical Anchor



The vertical anchor property can take three values:

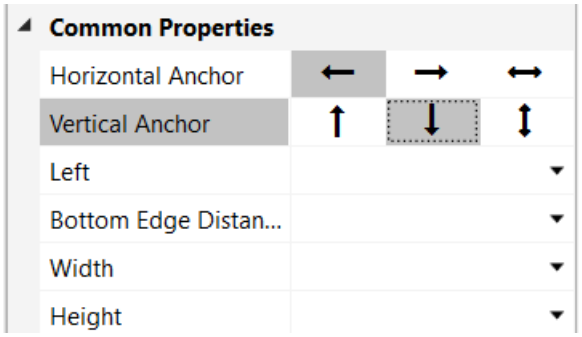
-  TOP



This is the default value.
The top edge is anchored to the top edge of the parent view with the distance given in the Top property.

Left and Top anchors are shown.

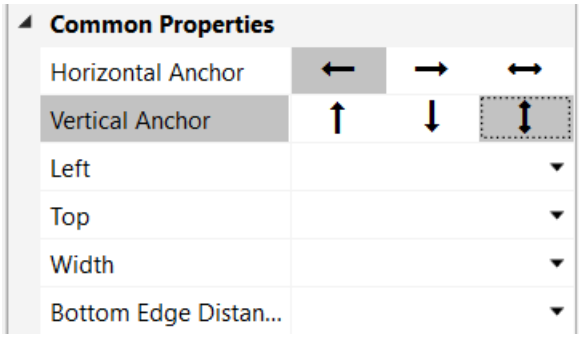
-  BOTTOM



The bottom edge is anchored to the bottom edge of the parent view with the distance given in the Bottom Edge Distance property.
The Top property is no longer available because it is defined by the Height and the bottom anchor !

The dot on the left and on the bottom edge show the anchors.

-  BOTH



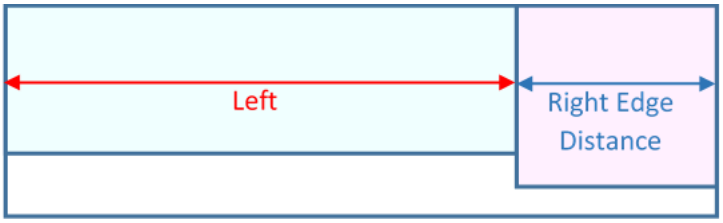
Both edges are anchored.
The Height property is no longer available because it is defined by the anchors !

Setting the Vertical Anchor property to BOTH is like using the SetTopAndBottom function in the Designer Scripts.

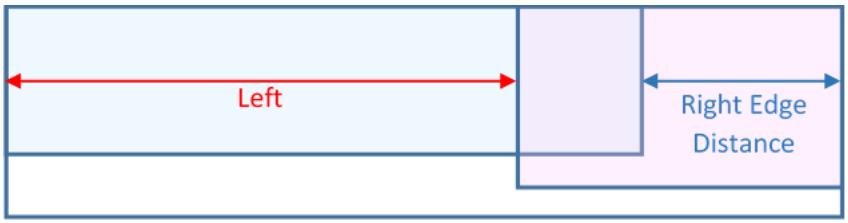
The dots on the left and the two edges show the anchors.

What happens when we set the horizontal anchor of the two views below to BOTH and change the parent view width?

The left view's right edge is anchored to the right edge of the parent view with the Right Edge Distance.
The right view's left edge is anchored to the left edge of the parent view with the Left distance.



If we increase the width of the parent view, we get the layout below.



The left view's right edge is still at the Right Edge Distance from the parent view's right edge.
The right view's left edge is still at the Left distance from the parent view's left edge.
The result is an overlapping of both views.

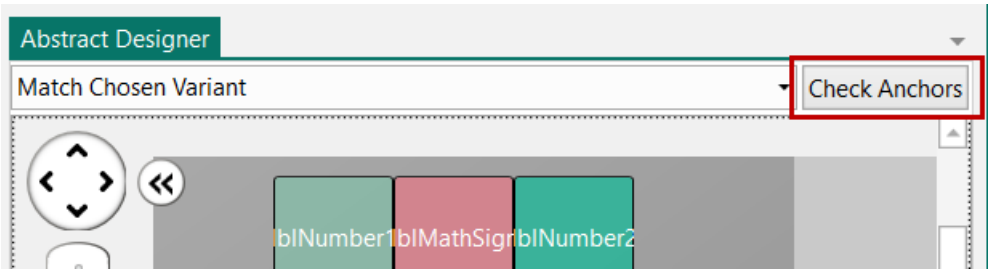
In this case you must adjust the views in the Designer Scripts with the `SetLeftAndRight` method!

For example:

```
LeftView.SetLeftAndRight(0, 67%x)
RightView.SetLeftAndRight(33%x, 100%x)
```

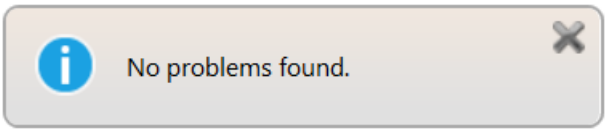
3.12.3 AnchorChecker

In the top right corner of the Abstract Designer you find the Check Anchors button to check if the anchors are coherent.



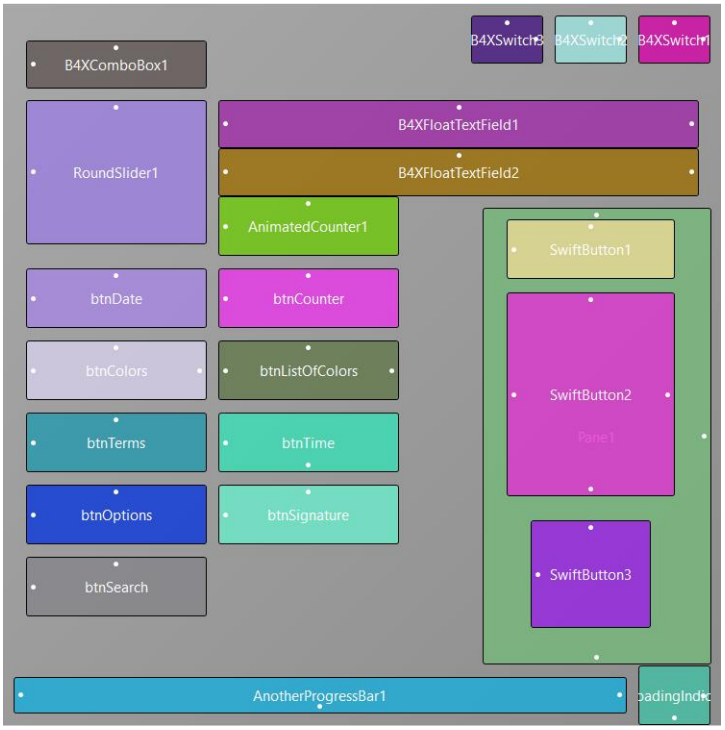
When you click on Check Anchors all views with none coherent anchors are displayed in red.

Otherwise, you'll get this message.

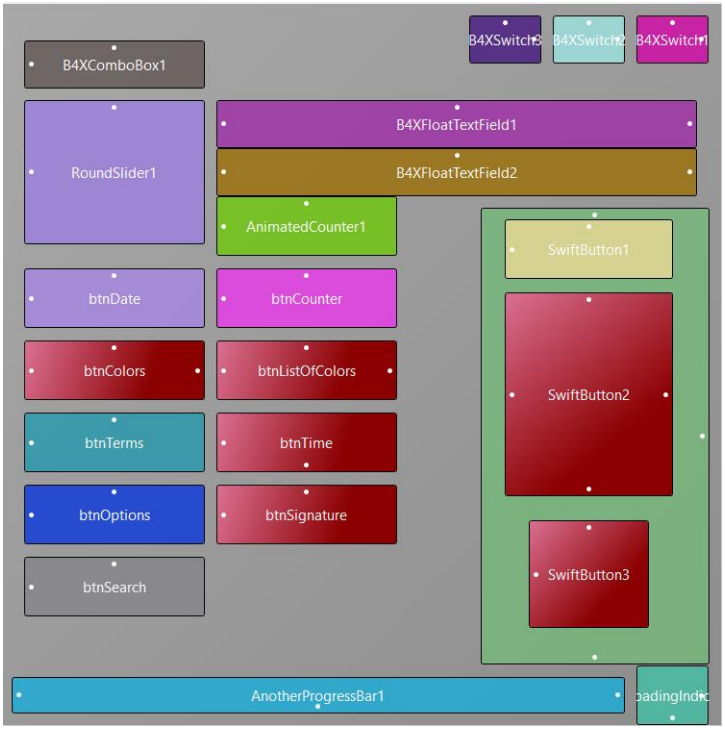


Example with ‘wrong’ anchors.
The B4J source code is in the *Getting Started\SourceCode\CheckAnchors* folder.

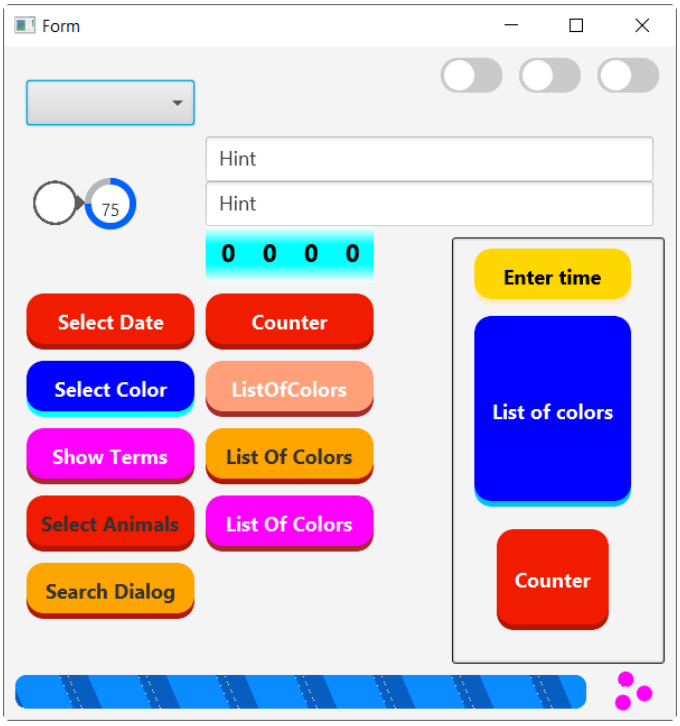
The layout:



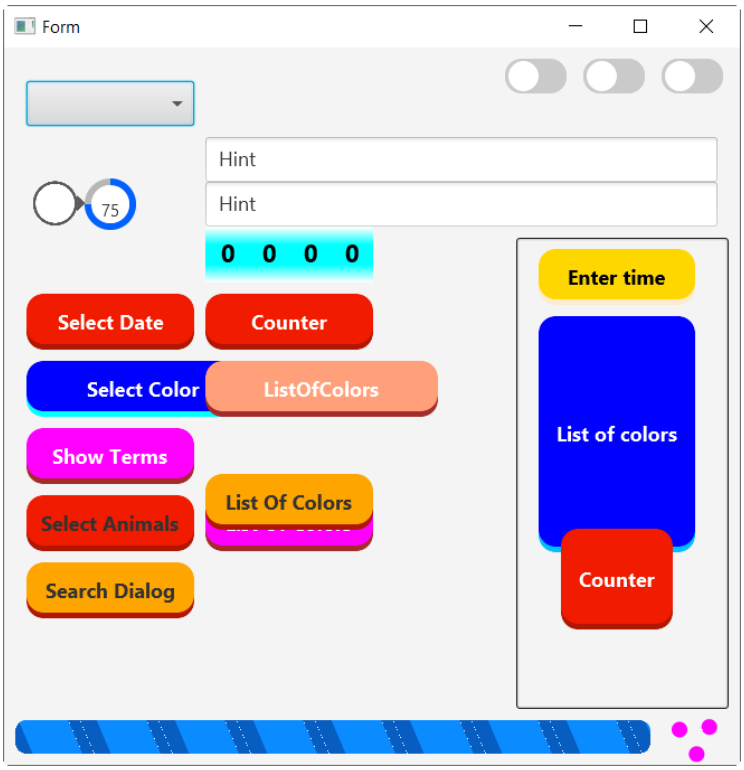
After the click on Check Anchors:



Screenshot of the result.



And after stretching the window horizontally and vertically:



You may have a look at this thread in the forum, it shows an animation:
https://www.b4x.com/android/forum/threads/new-feature-anchors-checker.108805/

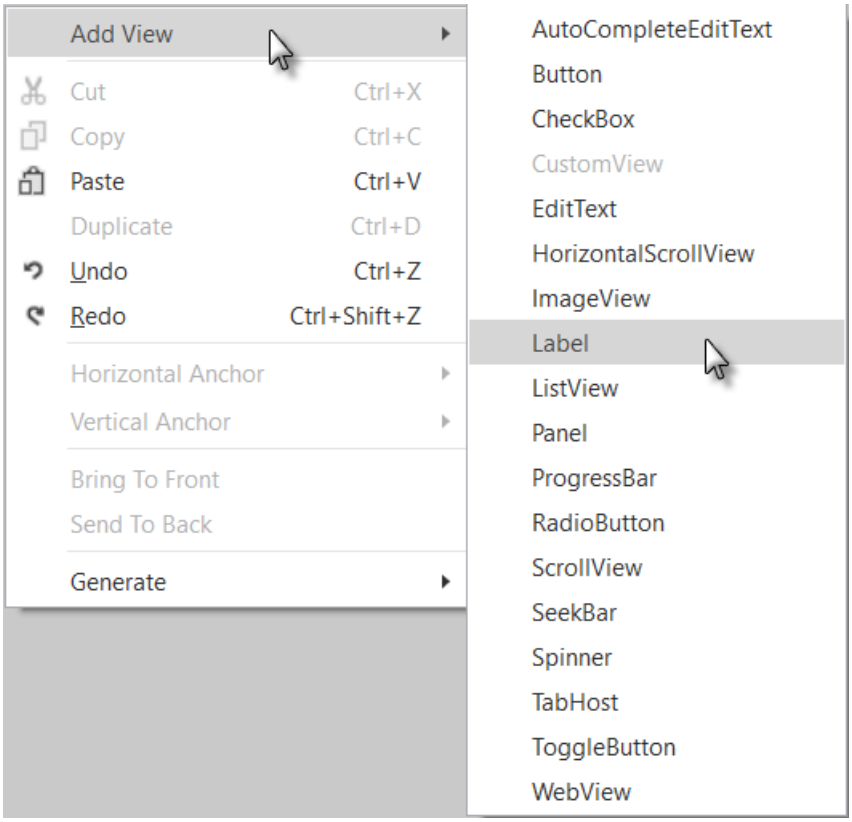
3.12.4 Example project

The examples shown in this chapter are based on the *DesignerAnchor* project.
The source code is in the *Getting Started\SourceCode\DesignerAnchor* folder.

The example is made with B4A, but the principle is exactly the same with B4i and B4J

First, we add a label on top of the screen which should cover the whole width and stay on top.

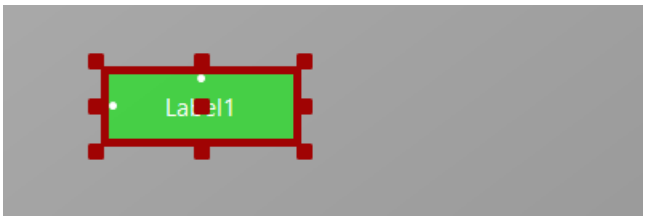
In the AbstractDesigner right-click somewhere on the screen, the menu below will be displayed:



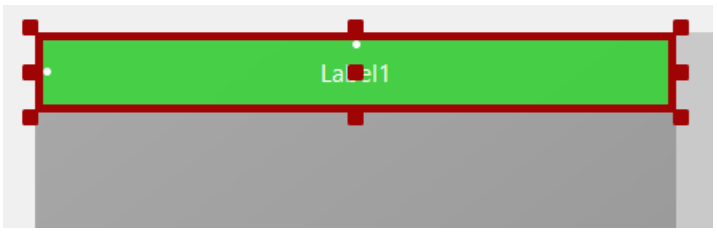
Click on Add View.

On the left side appears the list of the views you can add to the layout.

Click on Label.



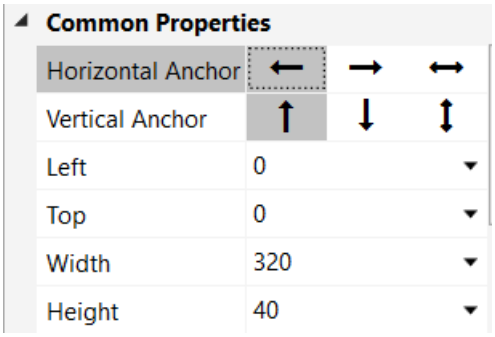
The Label is added.



Move the label's upper left corner to the upper left corner of the screen and stretch it to fill the whole width of the screen.

Click somewhere else on the screen to remove the red anchors.

The left and top anchors are displayed.



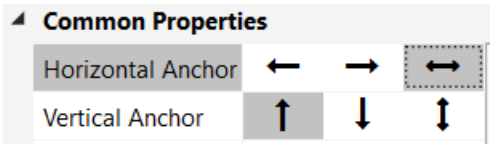
Click again on the Label and we see these properties:

Left = 0

Top = 0

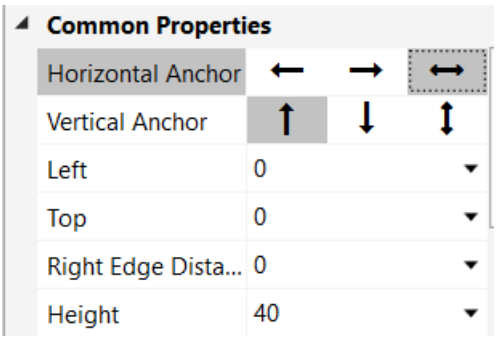
Width = 320 full layout width

Height = 40

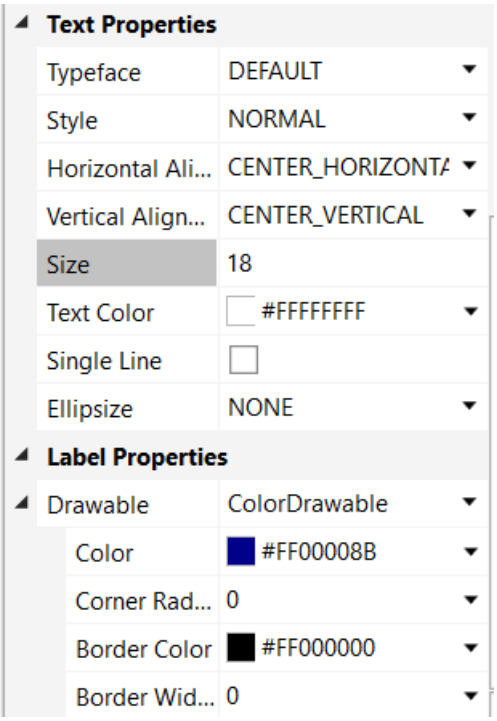


Now we change the 'Horizontal Anchor' property:

Click on  BOTH.

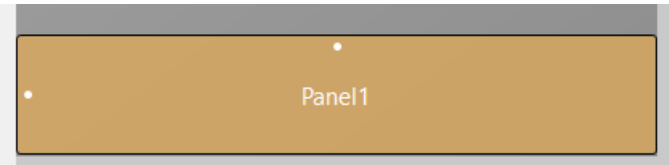


We see that the properties changed:
Left, Top and Height are still the same.
But Width has disappeared and is replaced by
Right Edge Distance = 0
Its value = 0 because the right edge is on the right edge of the screen.



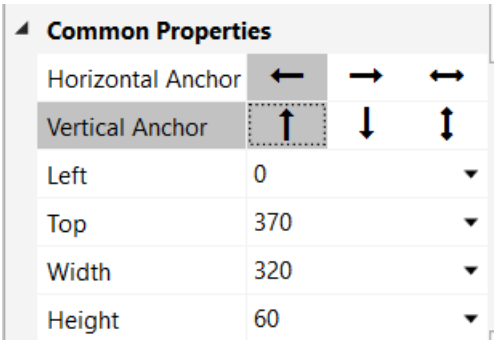
Set the other properties like in the picture.

Now we see the top anchor and the anchors on the left and the right edge.

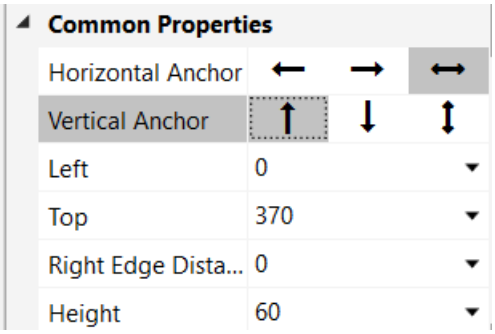


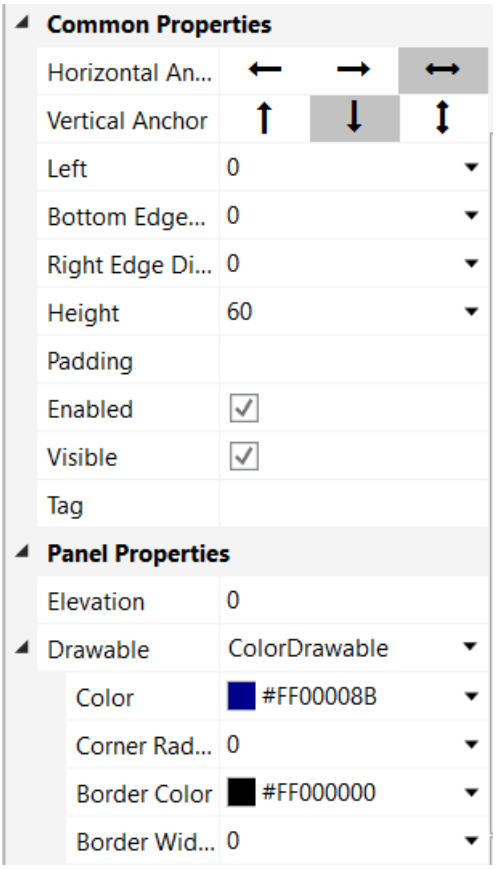
Now, let us add a Panel at the bottom of the screen covering also the whole screen width.

The properties look like in the picture.



We set the Horizontal Anchor to BOTH.
Same as for Label1.

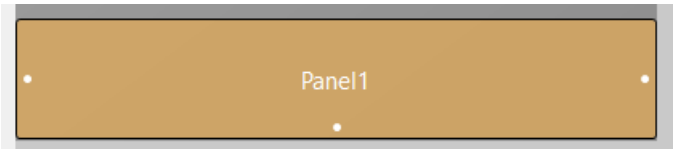




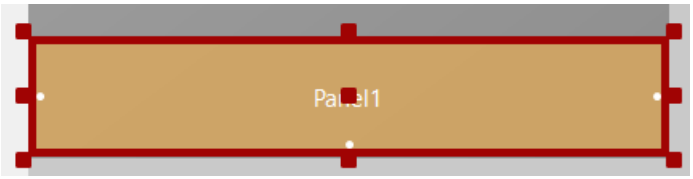
We set the Vertical Anchor to  BOTTOM.

The Top property is replaced by the:
Bottom Edge Distance = 0 property.
Its value = 0 because we anchor the bottom edge of Panel1 to the screens bottom edge.

We see the three anchors.

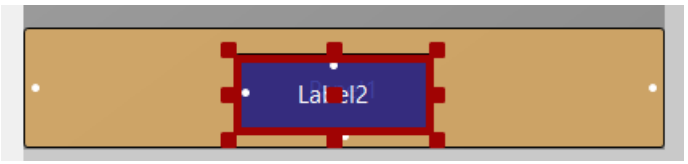


And set the other properties like this.

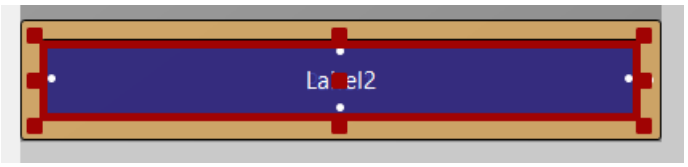


Now we add a second label onto Panel1.

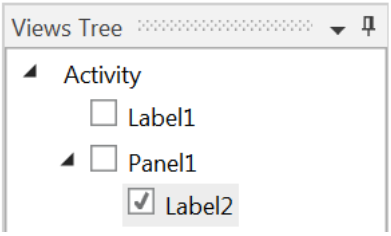
Click on Panel1 to select it.



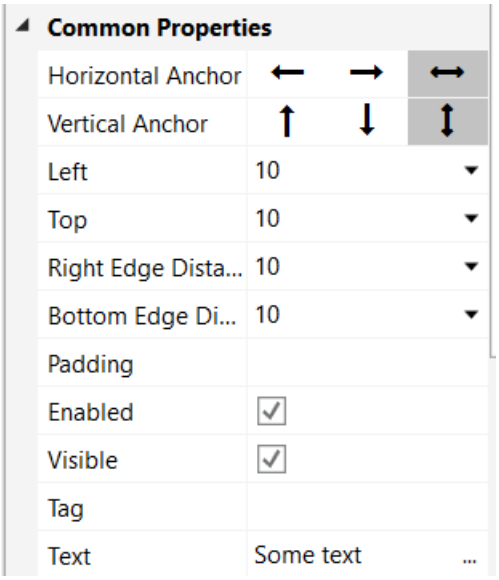
Add the label.



Move and size the label like in the picture with the Left, Top, Width and Height properties like in the list below.



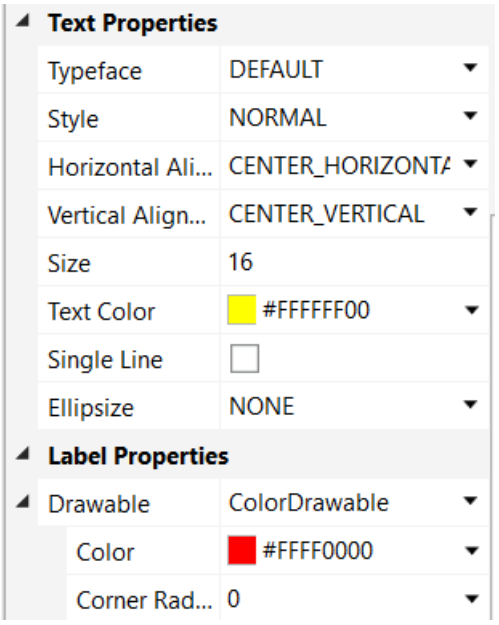
In the Views Tree window, we see that Label2 is shifted to the right because its parent view is Panel1 and not the Activity like for Label1 and Panel1!



Set the Horizontal and Vertical Anchors to ↔ ↕.

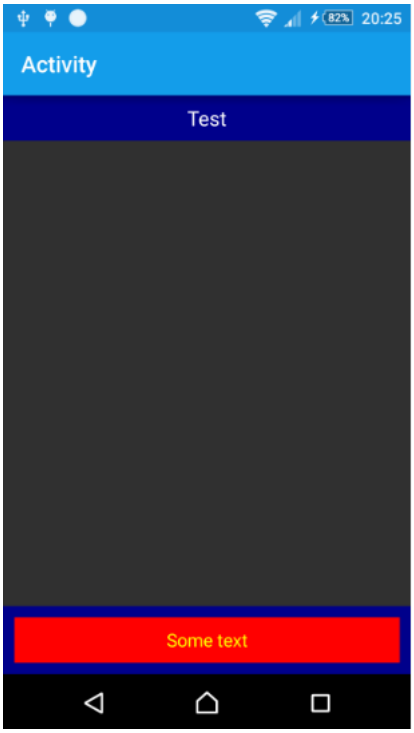
The properties
Left = 10 and
Top = 10 remain the same.

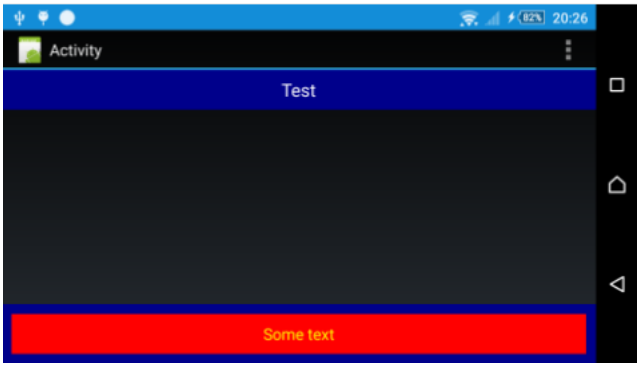
Right Edge Distance = 10 and
Bottom Edge Distance = 10
The two values are equal to 10 because we want a 'frame' around Label2.



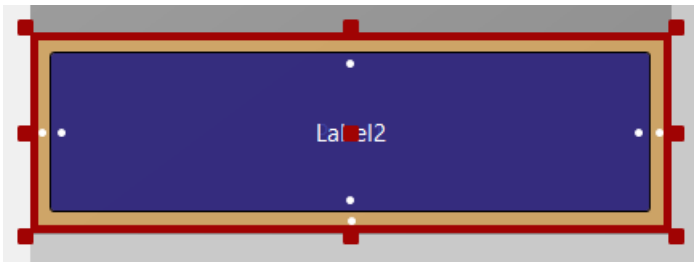
Set the other properties like in the picture.

And the result looks like the pictures below in portrait and landscape screen orientations.

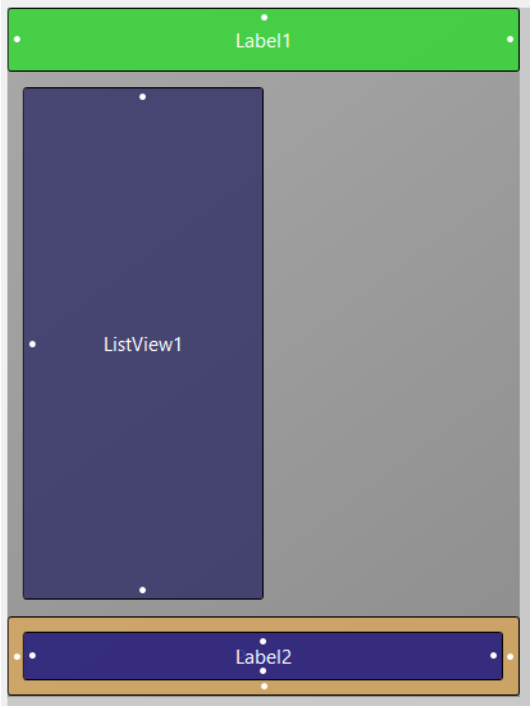


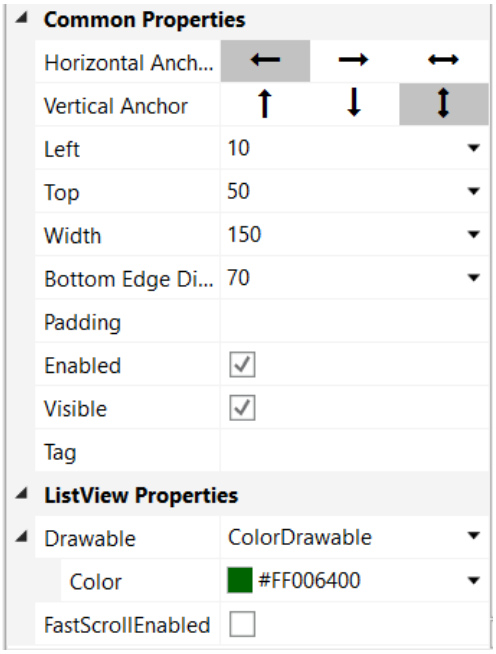


To demonstrate the anchor feature we move, in the Abstract Designer, the top edge of Panel1 upwards.



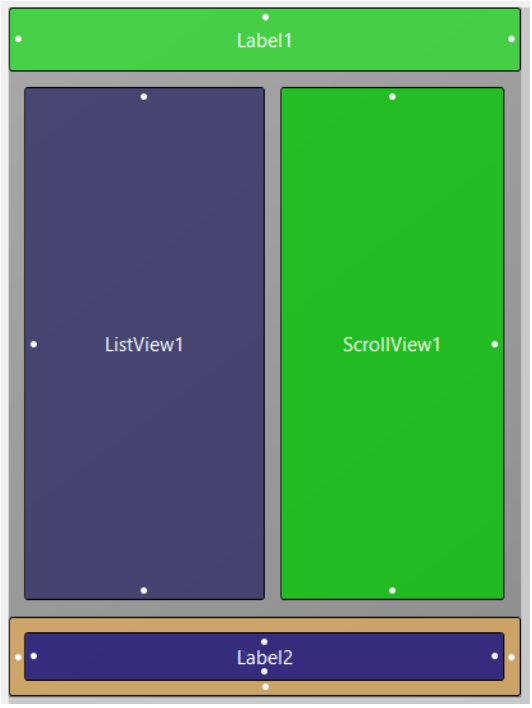
We see that the bottom edge of Label2 remains at its place !

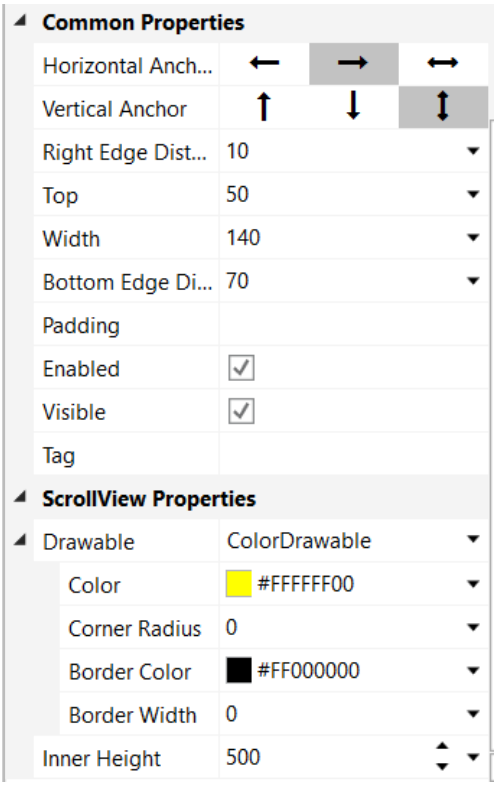




Now, we add a ListView onto the left half of the screen and vertically positioned between Label1 and Panel1 leaving a small space.

We set the vertical anchor to  BOTH.
And set the other properties like in the picture.





Now, we add a ScrollView on the right half of the screen also positioned between Label1 and Panel1 leaving a small space.

We set the horizontal anchor to  RIGHT.
We set the vertical anchor to  BOTH.
And set the other properties like in the picture.

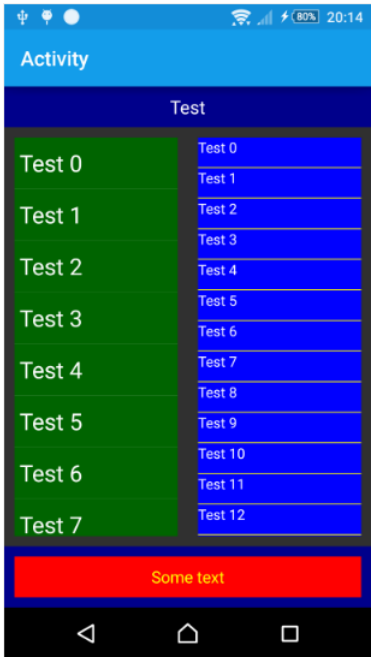
In the code we:
- Load the layout.
- Fill the ListView and the ScrollView.

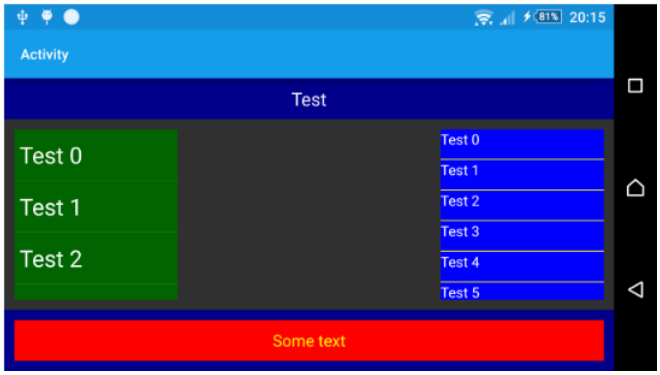
```
Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("Main")
   FillListView
   FillScrollView
End Sub
```

The two filling routines.

```
Sub FillListView
   Private i As Int

   For i = 0 To 20
      ListView1.AddSingleLine("Test " & i)
   Next
End Sub

Sub FillScrollView
   Private i As Int
   Private lblHeight = 30dip As Int

   For i = 0 To 20
      Private lbl As Label
      lbl.Initialize("lbl")
      ScrollView1.Panel.AddView(lbl, 0, i*lblHeight, 100%x-20dip, lblHeight-1dip)
      lbl.Color = Colors.Blue
      lbl.TextColor = Colors.White
      lbl.Text = "Test " & i
      lbl.Tag = i
   Next
   ScrollView1.Panel.Height = i * lblHeight
End Sub
```

And the result:
In portrait and landscape screen orientations.





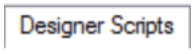
We see that the anchors work fine.
But, we see that there is a big gap between the ListView and the ScrollView.

Why do we have this gap ?

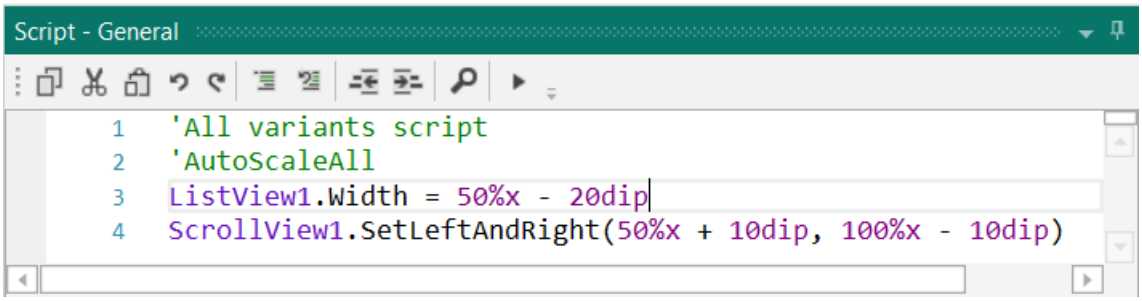
Because we set the Horizontal Anchor of the ListView to LEFT
and the Horizontal Anchor of the ScrollView to RIGHT.

But the Width property remains the same and that's why we get the gap between the two views when the screen width is wider than the layout screen width.

To adjust the width, we add two lines in the DesignerScripts.

Click on 

to show the DesignerScripts window.

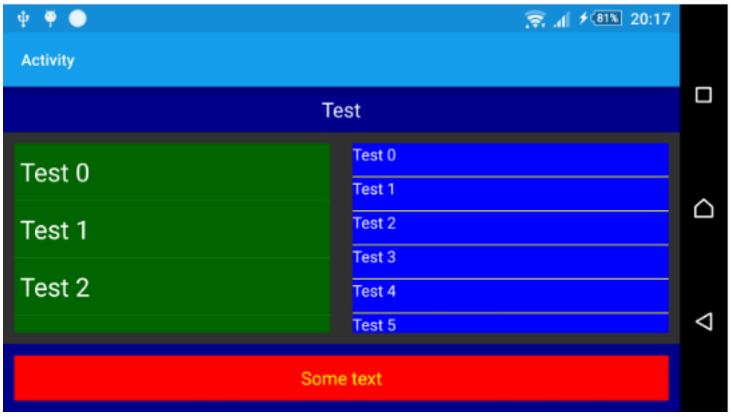


Here we comment AutoScaleAll and add the following two lines:

```
'AutoScaleAll
ListView1.Width = 50%x - 20dip
ScrollView1.SetLeftAndRight(50%x + 10dip, 100%x - 10dip)
```

The anchors are valid in the AbstractDesigner but not in Designer Scripts.

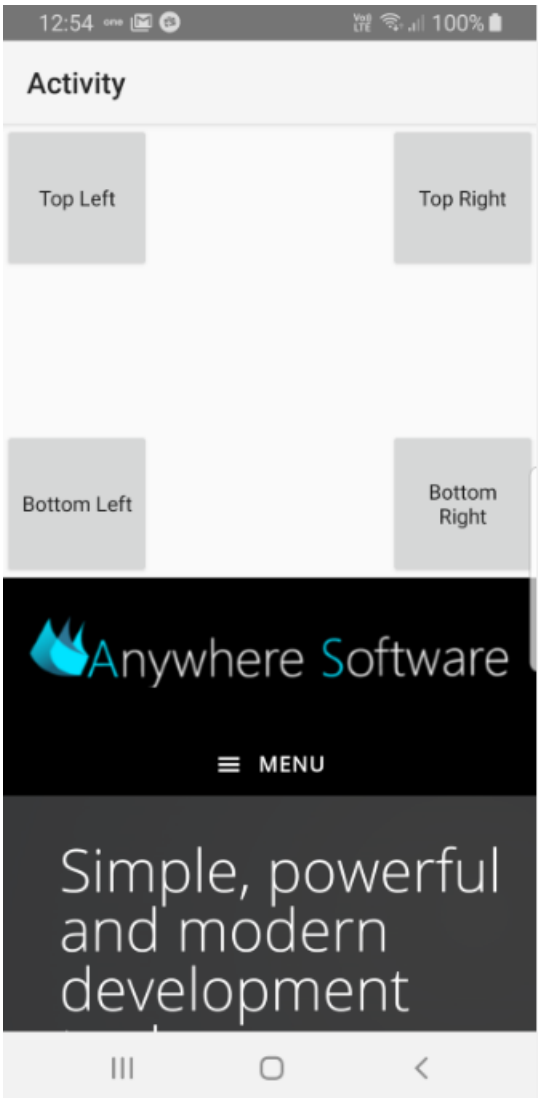
For ListView1 it's enough to set its Width property.
But for ScrollView1 we need to define both properties Left and Right which is done with `SetLeftAndRight` because the RIGHT anchor is lost.



And the new result in landscape orientation.

3.12.5 Nested layouts

Lets say that we want to build a layout such as below where the screen is split into two halves.



If we try to build this layout with a single layout file we will quickly meet a problem.
We will add this designer script to split the screen:

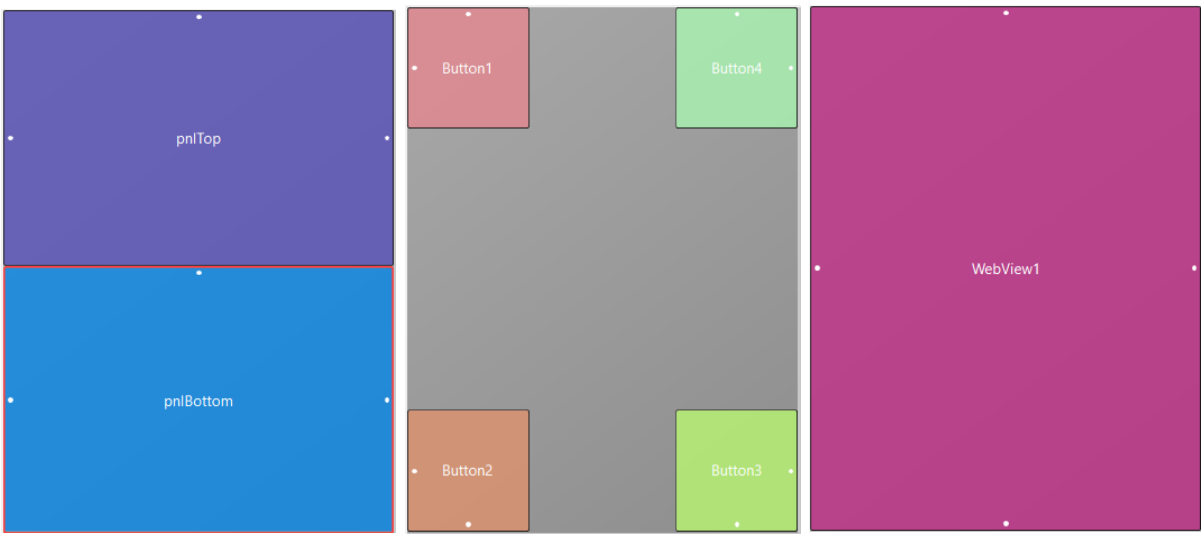
```
pnlTop.SetTopAndBottom(0, 50%y)
pnlBottom.SetTopAndBottom(50%y, 100%y)
```

The problem is that anchors are applied right before the designer script is executed, so the anchors will not be updated when the panels are resized.
This means that we will not be able to use anchors and will need to set the size of all the views in the designer script.

However there is a better solution, we can split the layout into three nice and clean layout files and load them with:

```
Sub Globals
   Private WebView1 As WebView
   Private pnlTop As B4XView
   Private pnlBottom As B4XView
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("Main")
   pnlTop.LoadLayout("Top")
   pnlBottom.LoadLayout("Bottom")
   WebView1.LoadUrl("https://www.b4x.com")
End Sub
```

The only designer script needed is for the two panels in the main layout.
Note that the variant size doesn't matter. The nested layouts will be resized based on the parent panels sizes. Look also at the anchors.



Main lyout Top layout Bottom layout

3.13 Designer Scripts

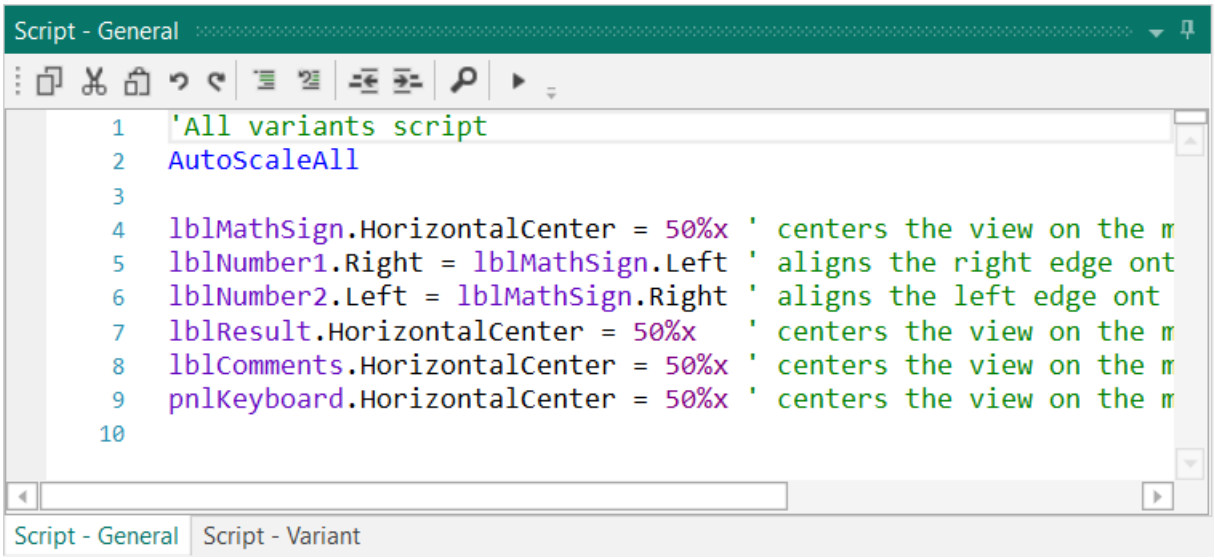
One of the most common issues that Android and iOS developers face is the need to adapt the user interface to devices with different screen sizes, it is much less important in B4J.
As described in the visual designer tutorial, you can create multiple layout variants to match different screens.
However it is not feasible nor recommended to create many layout variants.

The Designer Scripts will help you fine tune your layout and easily adjust it to different screens and resolutions.

The idea is to combine the usefulness of the visual designer with the flexibility and power of programming code.

You can write a simple script to adjust the layout based on the dimensions of the current device and immediately see the results. No need to compile and install the full program each time.

You can also immediately see the results in the Abstract Designer. This allows you to test your layout on many different screen sizes.



Picture from the SecondProgram project.

3.13.1 General

Every layout file can include script code. The script is written inside the Visual Designer in the Script window:

There are two types of scripts:
- Script – General, the general script that will be applied to all variants.
- Script – Variant, specific code can be written for each variant.

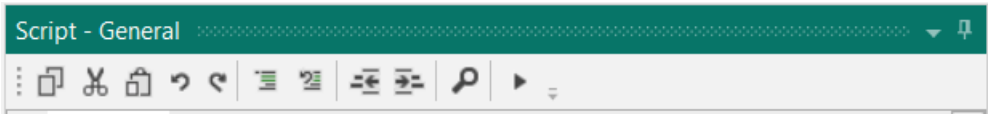
Once you press on the Run Script button (or F5), the script is executed, and the connected device / emulator and abstract designer will show the updated layout.

The same thing happens when you run your compiled program. The (now compiled) script is executed after the layout is loaded.

The general script is first executed followed by the variant specific script.

The script language is very simple and is optimized for managing the layout.

3.13.2 The menu



	Ctrl + C	Copy
	Ctrl + X	Cut
	Ctrl + V	Paste
	Ctrl + Z	Undo
	Ctrl + Shift + Z	Redo
	Ctrl + Q	Block Comment
	Ctrl + W	Block Uncomment
		Outdent
		Indent
	F3	Find / Replace
	F5	Run

3.13.3 Supported Properties

The following properties are supported:
- **Left / Right / Top / Bottom / HorizontalCenter / VerticalCenter** –
Gets or sets the view's position. The view's width or height will not be changed.

- **Width / Height** - Gets or Sets the view's width or height.

- **TextSize** - Gets or sets the text size.
You should not use 'dip' units with this value as it is already measured in physical units.

- **Text** - Gets or sets the view's text. TextSize and Text properties are only available to views that show text.

- **Image** - Sets the image file (write-only). Only supported by ImageView.

- **Visible** - Gets or sets the view's visible property.

3.13.4 Supported Methods

- **SetLeftAndRight** (Left, Right) - Sets the view's left and right properties. This method changes the width of the view based on the two values.

- **SetTopAndBottom** (Top, Bottom) - Sets the view's top and bottom properties. This method changes the height of the view based on the two values.

3.13.5 Supported Keywords

- **And / Or** - Same as the standard And / Or keywords.

- **False / True** - Same as the standard False / True keywords.

- **Min / Max** - Same as the standard Min / Max keywords.

- **Landscape / Portrait** - Detects if the layout is in landscape or portrait.
Can be used with If / Then.

- **AutoScale** - Autoscales a view based on the device physical size. Example: AutoScale(Button1)

- **AutoScaleAll** - Autoscales all layout views.

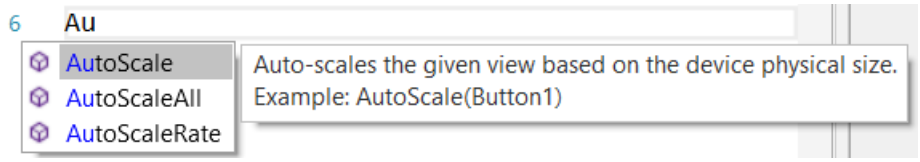
- **AutoScaleRate** - Sets the scaling rate, a value between 0 and 1. The default value is 0.3
Example : `AutoScaleRate(0.5)`

- **ActivitySize** - Returns the approximate activity size measured in inches.

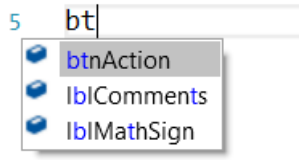
- **If . Else If . Else . Then** condition blocks - Both single line and multiline statements are supported. The syntax is the same as the regular If blocks.

3.13.6 Autocomplete

When you begin typing, the AutoComplete function shows all possible keywords or view names containing the written text with the help of the selected keyword.
Example: Au, shows all AutoScale methods.

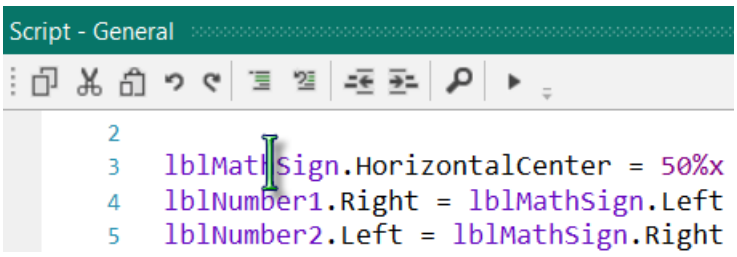


Example: Writing bt, shows all views containing the characters ‘b’ and ‘t’.

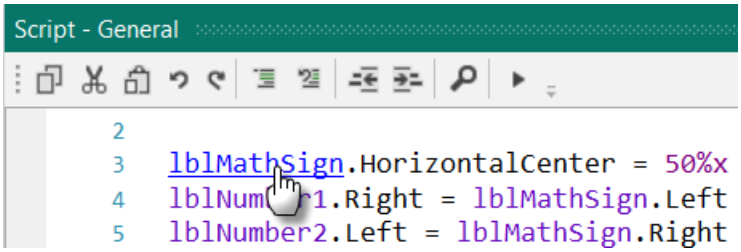


3.13.7 Select a view in the DesignerScript with Ctrl + Click

You can select a view directly in the DesignerScript with Ctrl + Click.
Examples with the SecondProgram.

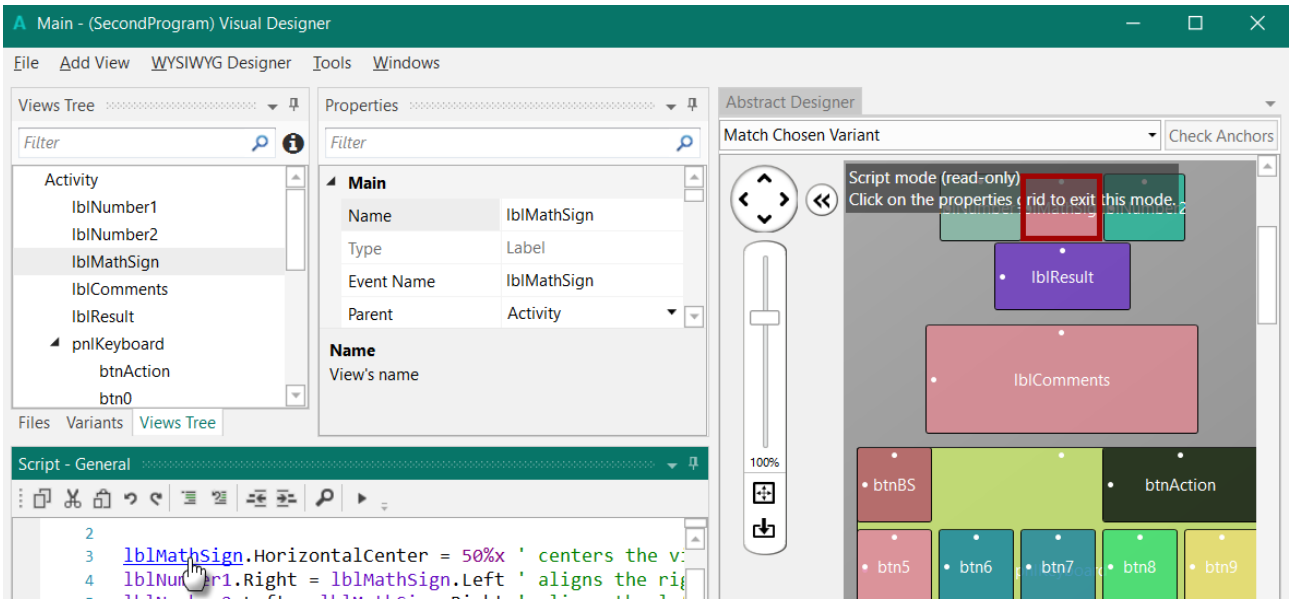


When the cursor in on a view name.



And you press Ctrl,
The viewname is highlightes in blue and the cursor becomes a hand.

If you click now, the view is selected.



You see the view name in the Properties window and the views border is colored in red in the Abstract Designer

3.13.8 Notes and tips

- %x and %y values are relative to the view that loads the layout.
Usually it will be the activity or main page. However, if you use Panel.LoadLayout then it will be relative to this panel.

- **(B4A only) Use 'dip' units for all specified sizes (except of TextSize).** By using 'dip' units the values will be scaled correctly on devices with higher or lower resolution.

- In most cases it is not recommended to create variants with scales other than 1.0. When you add such a variant you will be given an option to add a normalized variant instead with a scale of 1.0.

- Variables - You can use variables in the script. You do not need to declare the variables before using them (there is no Private, Public nor Dim keyword in the script).

- (B4A only) Activity.RerunDesignerScript (LayoutFile As String, Width As Int, Height As Int) - In some cases it is desirable to run the script code again during the program. For example, you may want to update the layout when the soft keyboard becomes visible. Activity.RerunDesignerScript method allows you to run the script again and specify the width and height that will represent 100%x and 100%y. For this method to work all the views referenced in the script must be declared in Sub Globals.
Note that this method should **not** be used to handle screen orientation changes. In that case the activity will be recreated, and the script will run during the Activity.LoadLayout call.

3.14 AutoScale

AutoScale includes three functions:

- AutoScaleRate(rate)
- AutoScale
- AutoScaleAll

Larger devices offer a lot more available space. The result is that even if the physical size of a view is the same, it just "feels" smaller.
Some developers use %x and %y to specify the views size. However, the result is far from being perfect. The layout will just be stretched.
The solution is to combine the "dock and fill" strategy with a smart algorithm that increases the views size and text size based on the running device physical size.

The AutoScale function is based on the standard variant (320 x 480, scale = 1.0).
Since B4A version 3.2 AutoScale considers the dimensions of the variant defined in the layout.
For other screen sizes and resolutions AutoScale calculates a scaling factor based on the equations below.

```
delta = ((100%x + 100%y) / (320dip + 430dip) - 1)
rate = 0.3 'value between 0 to 1.
scale = 1 + rate * delta
```

AutoScale multiplies the Left / Top / Width and Height properties by the scale value.
If the view has a Text property this one is also multiplied by the scale value.

You can play with the 'rate' value. The rate determines the change amount in relation to the device physical size.
Value of 0 means no change at all. Value of 1 is almost like using %x and %y: If the physical size is twice the size of the standard phone then the size will be twice the original size.
Values between 0.2 and 0.5 seem to give good results. The default value is 0.3.
Be careful when you ‘downsize’ a layout defined for a big screen to a small screen. The views may become very small.
Note: The size of the CheckBox and RadioButton images is the same for all screen sizes.

The abstract designer is useful to quickly test the effect of this value.

Functions:

- **AutoScaleRate(rate)** Sets the rate value for above equations.
  Example: `AutoScaleRate(0.5)` Sets the rate value to 0.5.

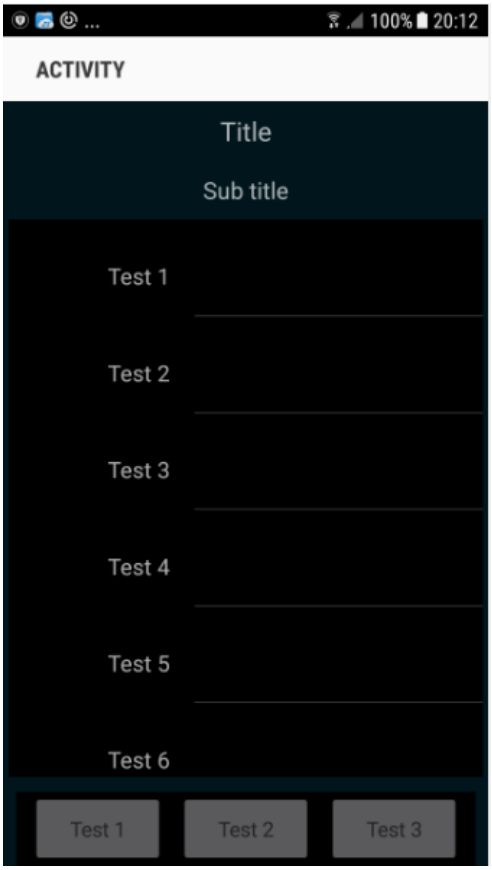
- **AutoScale(View)** Scales the given view.
  Example: `AutoScale(btnTest1)`
  This is equivalent to :

```
btnTest1.Left = btnTest1.Left * scale
btnTest1.Top = btnTest1.Top * scale
btnTest1.Width = btnTest1.Width * scale
btnTest1.Height = btnTest1.Height * scale
btnTest1.TextSize = btnTest1.TextSize * scale
```

- **AutoScaleAll** Scales all the views in the selected layout

3.14.1 Simple AutoScale example with only one layout variant

We will AutoScale a simple example with the layout below, source code AutoScaleExample:



The example is written with B4A, but the principle is similar for B4i.
Source code in the *Getting Started\SourceCode* folder.

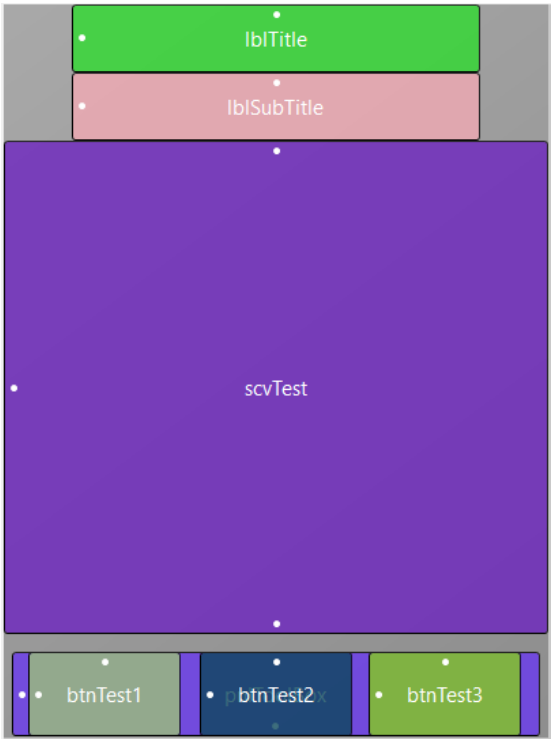
We have:

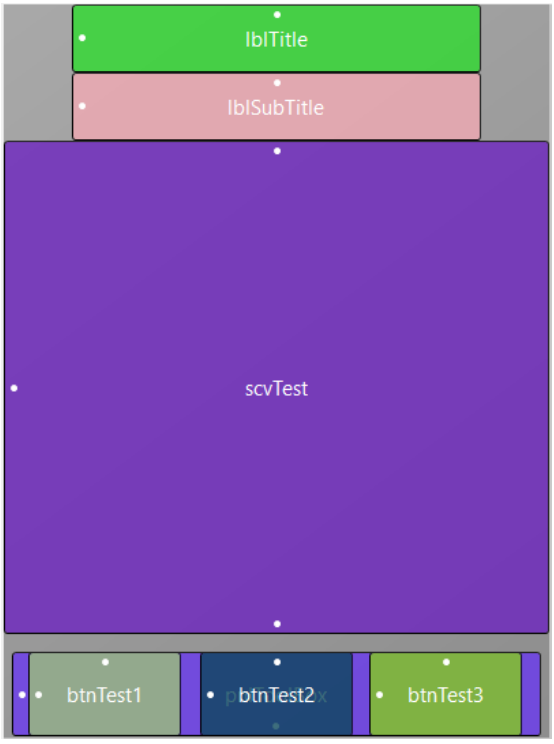
- 2 Labels on the top of the screen:
  - lblTitle
  - lblSubTitle

- 1 ScrollView in the middle of the screen:
  - scvTest containing
    - one Panel pnlSetup with
    - 10 Labels lblTest1 to lblTest10
    - 10 EditTexts edtTest1 to edtTest10

- 1 Panel at the bottom of the screen:
  - pnlToolBox
  - Containing 3 Buttons
    - btnTest1
    - btnTest2
    - btnTest3

We have two layout files *Main* for the main screen and *Panel* for the ScrollView content with only one layout variant 320 x 480 scale = 1 (160dip) for each.





Main layout file:

We want to have the:

- Two Labels on the top of the screen and centered horizontally on the screen.
- ToolBox Panel on the bottom of the screen and centered horizontally.
- ScrollView filling the space between the SubTitle Label and the ToolBox Panel.

Note: Look at the anchors especially for the ToolBox and the ScrollView.

First, we set the AutoScaleRate to 0.5 with:

```
AutoScaleRate(0.5)
```

and AutoScale all views with:

```
AutoScaleAll
```

The two Labels are already on top so there is no need to change the Top property for different screen sizes.

But we need to center them on the screen with:

```
lblTitle.HorizontalCenter = 50%x
lblSubTitle.HorizontalCenter = 50%x
```

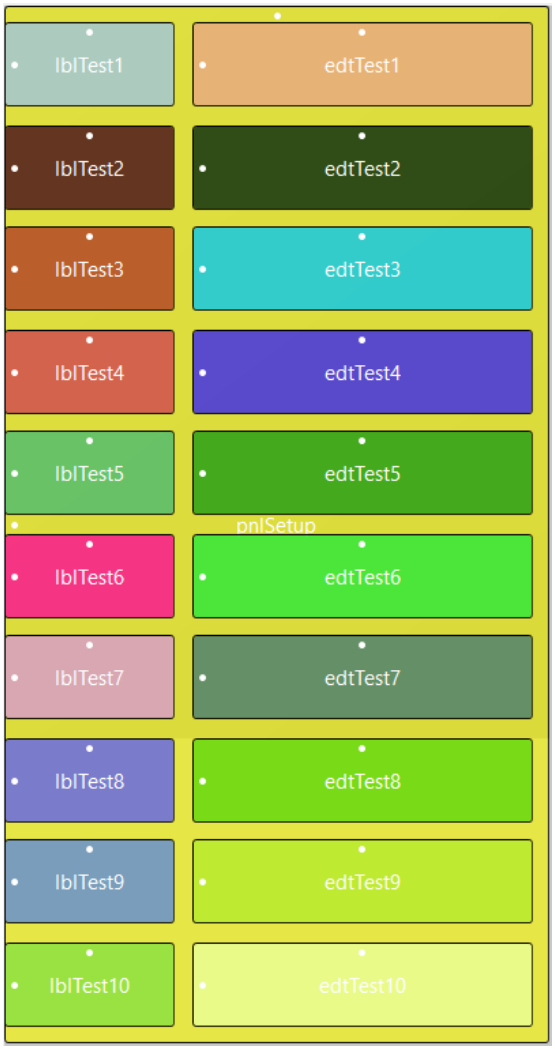
Then we center the ToolBox with:

```
pnlToolBox.HorizontalCenter = 50%x
```

And we set the Vertical Anchor property of the ToolBox to BOTTOM to 'anchor' it to the bottom of the screen.
This is needed because not all screens have the same width / height ratio and in landscape orientation it would even not be visible.

Then we set the Vertical Anchor property of the ScrollView to BOTH because we want it to fill the space between lblSubTitle and pnlToolBox.
We set the Bottom Edge Distance property to 60 to leave a small space of 10dip between the ScrollView and the ToolBox.
Code in the Designer Scripts of the Main layout in the area for All variants script:

```
'All variants script

'Set the rate value to 0.5
AutoScaleRate(1)

'Scale all the views in the layout
AutoScaleAll

'Center the Labels horizontally to the middle of the screen
lblTitle.HorizontalCenter = 50%x
lblSubTitle.HorizontalCenter = 50%x

'Center the ToolBox Panel horizontally to the middle of the screen
pnlToolBox.HorizontalCenter = 50%x

'Center the ScrollView horizontally to the middle of the screen
scvTest.HorizontalCenter = 50%x
```



Panel layout file:

All the Label and EditText views are on a Panel.
This is needed because they occupy more space than the screen size.
This layout file is loaded into the ScrollView.Panel.

For this layout file we set also the AutoScaleRate value to 0.5 with:
`AutoScaleRate(0.5)`
and AutoScale all views with:
`AutoScaleAll`

There is no need to modify any view after auto scaling.

Code in the Designer Scripts of the Panel layout in the area for 'All variants script':
The whole code is very simply:

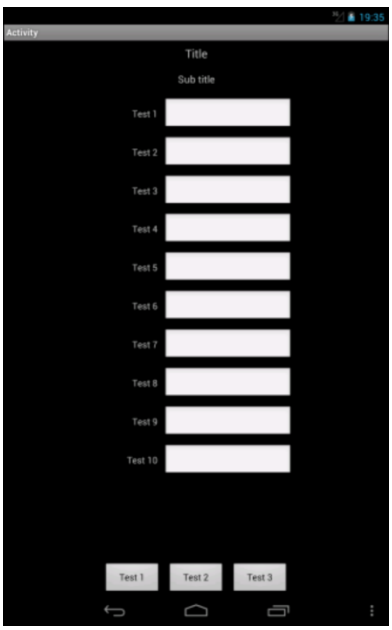
```
'All variants script
AutoScaleRate(0.5)
AutoScaleAll
```

In the program the code is the following:

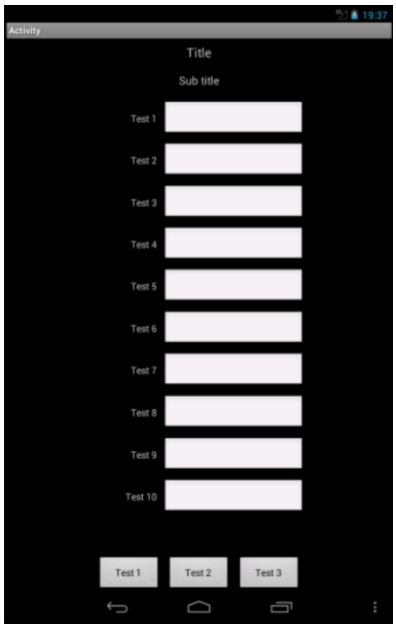
```
Sub Activity_Create(FirstTime As Boolean)
   ' load the Main layout file
   Activity.LoadLayout("Main")

   ' load the ScrollView.Panel layout file
   scvTest.Panel.LoadLayout("Panel")

   ' set the ScrollView.Panel.Height to the pnlSetup Panel height
   scvTest.Panel.Height = pnlSetup.Height
End Sub
```

We load the Main layout file into the Activity with `Activity.LoadLayout("Main")`.
We load the Panel layout file into the ScrollView with `scvTest.Panel.LoadLayout("Panel")`.
We set the ScrollView.Panel.Height to the height of the Panel in the layout file with:
`scvTest.Panel.Height = pnlSetup.Height`

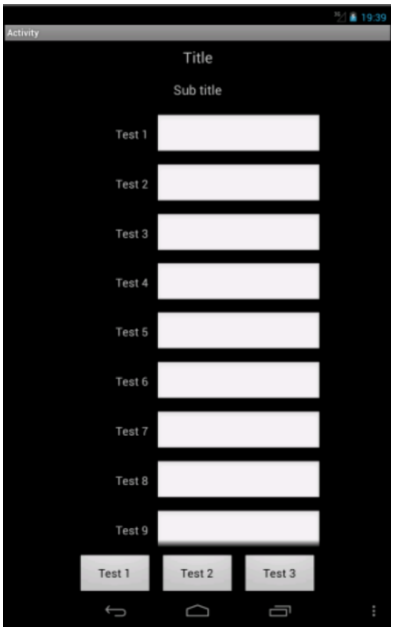
Screenshots of an 800/1280 10" screen Emulator with different Rate values:
All the images have been downsized.



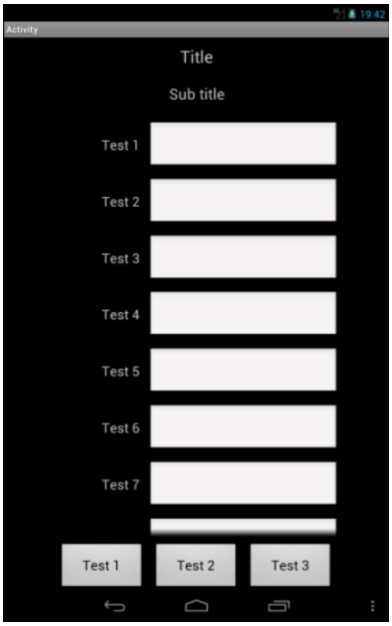
Rate = 0



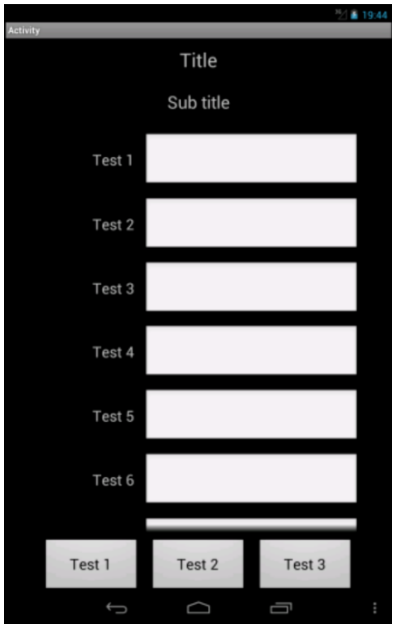
Rate = 0.1



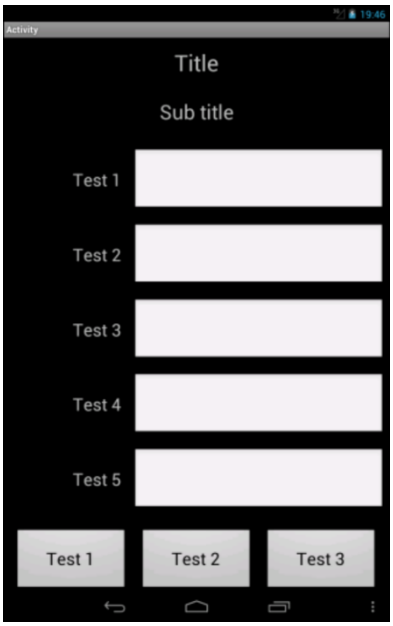
Rate = 0.3



Rate = 0.5

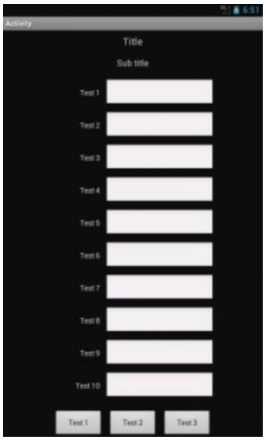


Rate = 0.7

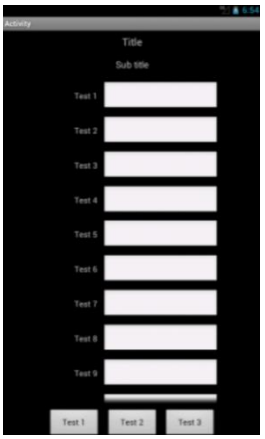


Rate = 1.0

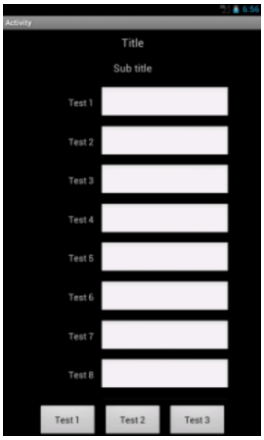
Screenshots of an 480/800 7" screen Emulator with different Rate values:



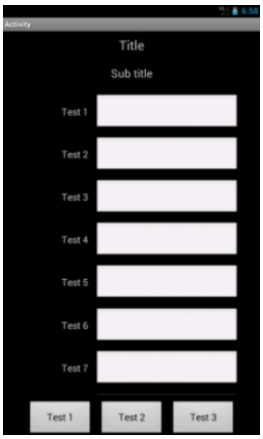
Rate = 0



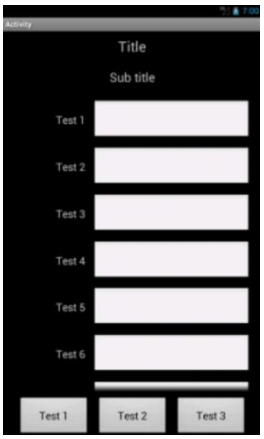
Rate = 0.1



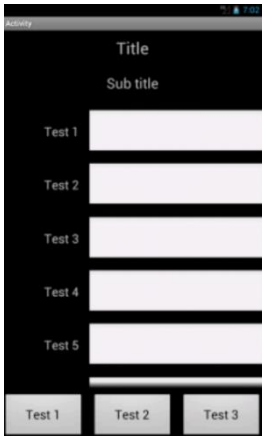
Rate = 0.3



Rate = 0.5



Rate = 0.7

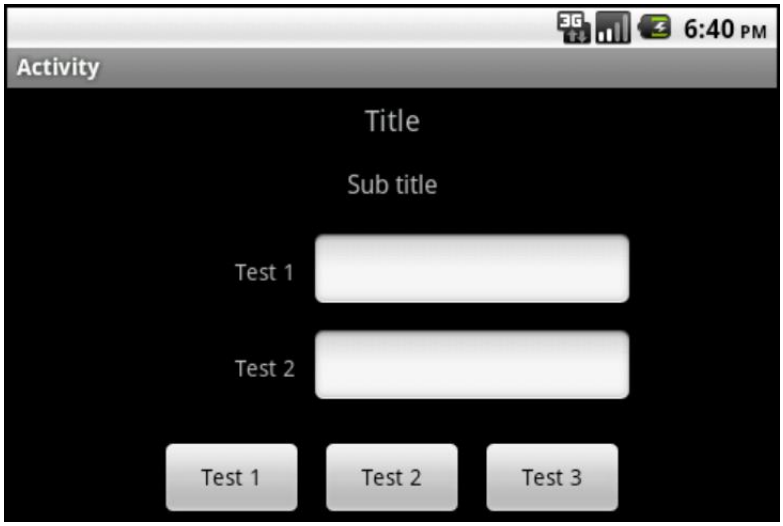


Rate = 1

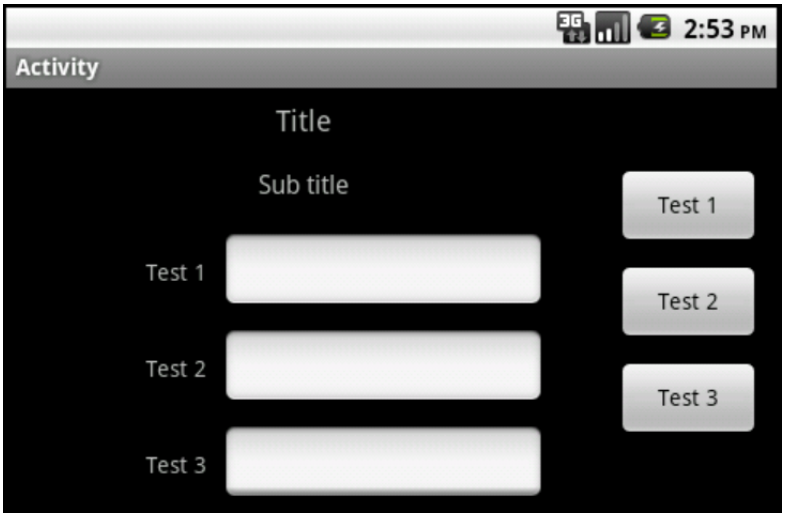
Screenshots of a 320/480 3.5" screen Emulator. The Rate value has no influence.

3.14.2 Same AutoScale example with portrait and landscape layout variants

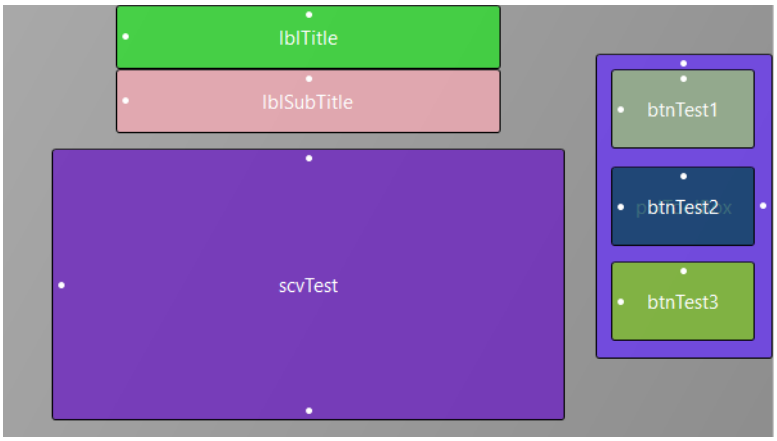
Source code *Getting Started\SourceCode\AutoScaleExample2*:



The previous example doesn't look good on smartphone screens with landscape orientation.



So, we make a new layout variant for landscape where we move the ToolBox with the Buttons to the right side of the screen.



The layout variant in the Main layout file.
Note: Look at the anchors especially for the ToolBox and the ScrollView.

The code in the Designer Script must be changed:

For the portrait variant in the Main layout file we keep in the `All variants script` area only the code below:

```
'All variants script
AutoScaleRate(0.5)
AutoScaleAll
```

Setting the rate value and autoscaling all the views.

All the other code is moved to the `Variant specific script: 320x480,scale=1` area:

```
'Variant specific script: 320x480,scale=1

'Center the Labels horizonally to the middle of the screen
lblTitle.HorizontalCenter = 50%x
lblSubTitle.HorizontalCenter = 50%x

'Center the ToolBox Panel horizontally to the middle of the screen
pnlToolBox.HorizontalCenter = 50%x

'Center the ScrollView horizontally to the middle of the screen
scvTest.HorizontalCenter = 50%x
```

For the landscape variant we have in the `All variants script` area the same code as for the portrait variant:

```
'All variants script
AutoScaleRate(0.5)
AutoScaleAll
```

And in the `'Variant specific script: 480x320,scale=1` area:

We center the Title and SubTitle Labels to the middle of the space between the left screen border and the left ToolBox boarder with:

```
lblTitle.HorizontalCenter = pnlToolBox.Left / 2
lblSubTitle.HorizontalCenter = pnlToolBox.Left / 2
```

We center the ToolBox vertically to the middle of the screen height with:

```
pnlToolBox.VerticalCenter = 50%y
```

We set the right border of the ToolBox to right border of the screen with:

```
pnlToolBox.Right= 100%x
```

We set the Vertical Anchor property of the ScrollView to BOTH to fill the space between the bottom SubTitle Label border and the bottom screen border with.

And the whole code:

```
'Variant specific script: 480x320,scale=1

'Center the ToolBox Panel vertically
pnlToolBox.VerticalCenter = 50%y

'Center the Labels horizontally to the middle
'of the space between the left screen border
'and the left boarder of the ToolBox Panel
lblTitle.HorizontalCenter = pnlToolBox.Left / 2
lblSubTitle.HorizontalCenter = pnlToolBox.Left / 2

'Center the ScrollView horizontally to the middle
'of the space between the left screen border
'and the left ToolBox Panel border
scvTest.HorizontalCenter = pnlToolBox.Left / 2
```

For the Panel layout file:

The code for the portrait variant remains the same.

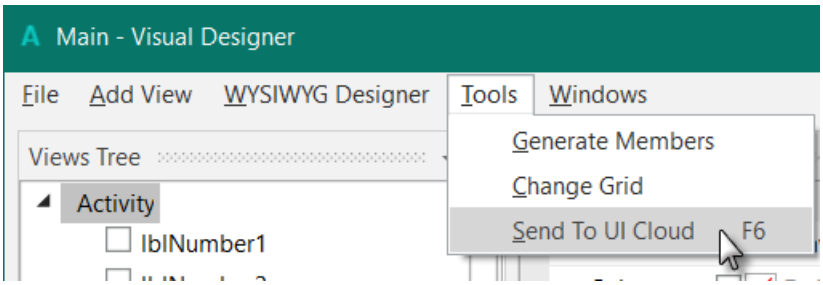
We add the same code for the landscape variant:

```
'All variants script
AutoScaleRate(0.5)
AutoScaleAll
```

Here too, no code in the `'Variant specific script: 480x320,scale=1` area.

3.15 UI Cloud B4A and B4i

With UI Cloud you can check how layouts look on different devices.



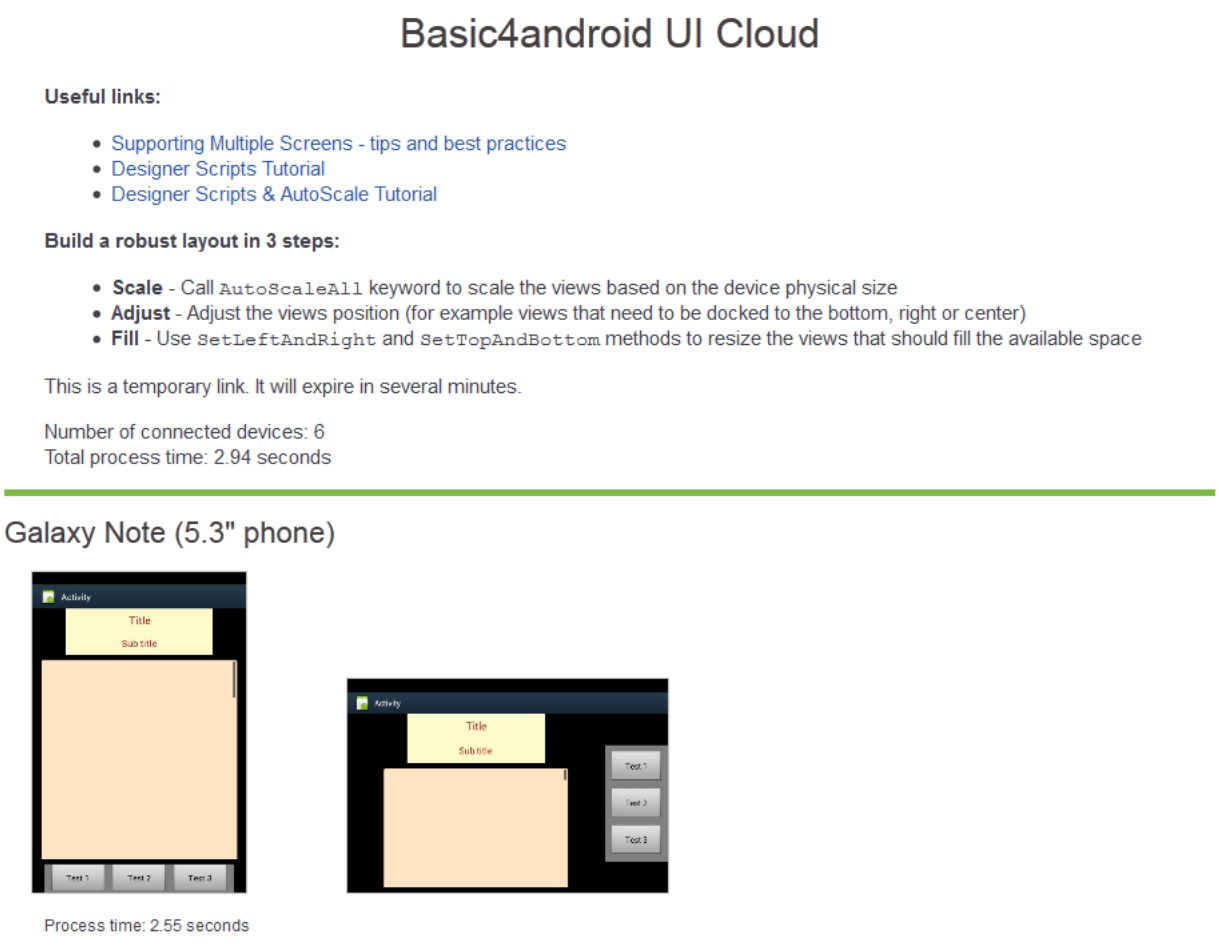
When you have defined a layout in the Designer Scripts you can send it to the UI Cloud in the tools menu.

The layout file is sent to the B4A site and you get a page showing your layout on different devices with different screen resolutions and densities.

It's a very convenient tool to check the layout look without needing to have physical devices.

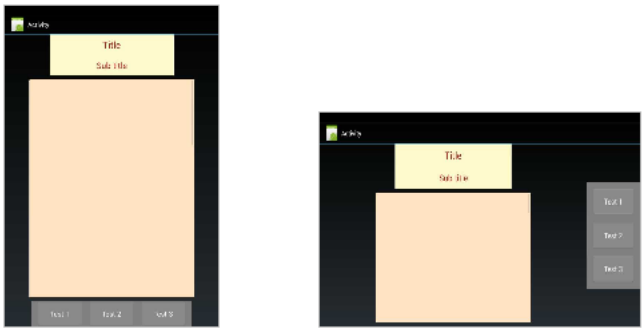
UI Cloud checks only layouts defined in the Designer, not layouts defined in the code !

Example of a UI Cloud screen:



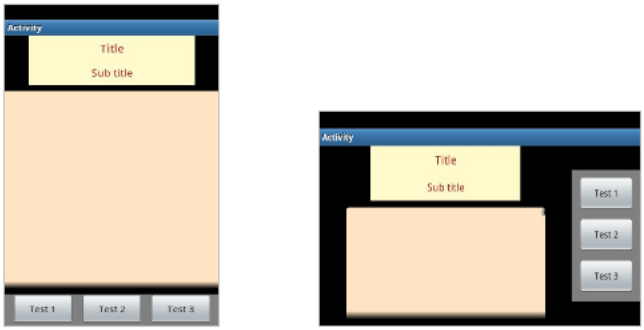
Some other devices:

Nexus 7 (7" tablet)



Process time: 2.44 seconds

Samsung I9000 (4" phone)



Process time: 2.33 seconds

You can click on an image to show it in real size: