B4X Booklets

B4X Graphics

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 following versions:
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 Graphics / Drawing

This guide covers the use of Graphics in the B4X languages.
All the source code and files needed (layouts, images etc) of the example projects in this guide are included in the SourceCode folder.

There are three folders for each project, one for each platform B4A, B4i and B4J.
This booklet covers graphic and drawing methods for the XUI cross platform objects and for the three products B4A, B4i and B4J.
The XUI cross platform objects are explained in detail in the B4X XUI Booklet.

2.1 Overview

To draw graphics, we need to use a Canvas object.

2.2 Tip

I suggest you, to use XUI graphics wherever you can, even for single platform projects.

It is cross platform, which means that the code is exactly the same for all three products.
This is the reason why the XUI chapters are always the first.

2.3 Canvas object

The Canvas object is different in the different B4X products.

In B4A, B4i and XUI the Canvas object needs a ‘container’ object.
In B4J the Canvas is a Node and doesn’t need a ‘container’ object.

It is possible to draw onto the following views:

XUI: A cross platform canvas, needs the XUI library.

- B4XView,
  only B4A / B4i Panel or B4J Pane views, not on ImageViews!

B4A:

- Activity
- ImageView
- Panel
- Bitmap (mutable)

B4i:

- ImageView
- Panel
- **Don’t use a Canvas with Page.RootPanel, you will get strange behaviors.**

B4J: Canvas is a node on its own, it is not related to another object.

2.3.1 Canvas initialization

The place to declare the views depends on the application.

2.3.1.1 XUI

```
Sub Process_Globals
   Private xui As XUI
   Private xplGraph As B4XView
   Private cvsGraph As B4XCanvas
End Sub
xplGraph = xui.CreatePanel("")
cvsGraph.Initialize(xplGraph)
```

2.3.1.2 BC BitmapCreator

```
Sub Process_Globals
   Private xui As XUI

   Private imvGraph As B4XView ' ImageView
   Private bmcGraph As BitmapCreator
End Sub
bmcGraph.Initialize(Width, Height)
```

2.3.1.3 B4A

```
Sub Globals
   Private pnlGraph As Panel
   Private cvsGraph As Canvas
End Sub
cvsGraph.Initialize(pnlGraph)
```

2.3.1.4 B4i

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

   Private pnlGraph As Panel
   Private cvsGraph As Canvas
End Sub

cvsGraph.Initialize(pnlGraph)
```

2.3.1.5 B4J

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

   Private cvsGraph As Canvas
End Sub
```

2.3.2 Invalidation for B4A and XUI / Refresh for B4i

Depending on where and when you use drawing methods in your code you need to add an Invalidate method for the view you draw on.
Redrawing will only happen when the program can process messages. Usually when it finishes running the current code.

2.3.2.1 XUI

The invalidation is done on the B4XCanvas object.

Example:

```
cvsGraph.Invalidate
```

`cvsGraph` is the B4XCanvas object.

2.3.2.2 B4A

Invalidation is done on the canvases parent object, a Panel, in the example below, or an ImageView.
Three different methods:

- **Invalidate**
  Invalidates the whole view forcing the view to redraw itself.
- **Invalidate2** (Rect As Rect)
  Invalidates the given rectangle.
- **Invalidate3** (Left As Int, Top As Int, Right As Int, Bottom As Int)
  Invalidates the given rectangle.

Example:

```
cvsGraph.DrawLine(0, 0, 100dip, 100dip, Colors.Red, 1dip)
pnlGraph.Invalidate
```

`cvsGraph` is the Canvas and `pnlGraph` is the Panel we draw on.

2.3.2.3 B4i

The invalidation is done on the Canvas object.

Example:

```
cvsGraph.Refresh
```

`cvsGraph` is the B4i Canvas object.

2.3.2.4 B4J

No Invalidation nor Refresh needed !

2.4 Drawing methods

Drawing methods summary of the Canvas objects of the different products (BC = BitmapCreator):
XUI and BC are cross-platform.

Method	B4A	B4i	B4J	XUI	BC	Comment
ClearRect			🟩	🟩		Sets the rectangle transparent
DrawArc					🟩	
DrawArc2					🟩	
DrawBitmap	🟩	🟩		🟩	🟩	B4J DrawImage
DrawBitmapFlipped	🟩					
DrawBitmapRotated	🟩	🟩		🟩		B4J DrawImageRotated
DrawCircle	🟩	🟩	🟩	🟩	🟩	
DrawCircle2					🟩	
DrawColor	🟩	🟩				
DrawDrawable	🟩					
DrawDrawableRotated	🟩					
DrawImage			🟩			Similar to DrawBitmap
DrawImage2			🟩			Similar to DrawBitmap
DrawImageRotated			🟩			Similar to DrawBitmapRotated
DrawLine	🟩	🟩	🟩	🟩	🟩	
DrawLine2					🟩	
DrawOval	🟩			🟩		XUI with path
DrawOvalRotated	🟩			🟩		XUI with path
DrawPath	🟩	🟩		🟩	🟩	
DrawPath2					🟩	
DrawPathRotated				🟩		
DrawPDF		🟩				
DrawPoint	🟩					
DrawRect	🟩	🟩	🟩	🟩	🟩	
DrawRect2					🟩	
DrawRectRotated	🟩	🟩	🟩			
DrawRectRounded		🟩		🟩	🟩	XUI with path
DrawRectRounded2					🟩	
DrawText	🟩	🟩	🟩	🟩		
DrawText2			🟩			
DrawTextRotated	🟩	🟩	🟩	🟩		
DrawView		🟩				

One big difference between B4J (Java) and B4A / B4i is the transparent color.

In B4A and B4i, when you draw with a Transparent color, the background becomes transparent and the underlying views become visible.
This means that you can draw almost any shape transparent.
In B4J, when you draw with a Transparent color, the specified color is drawn with Alpha = 0 onto the canvas bitmap which means: drawing with a transparent color in B4J (Java) has no effect.
The only way, in B4J, to set the background transparent is the Canvas.ClearRect method.
That’s the reason why in the B4J project the moving transparent area is a rectangle and not a circle.

2.4.1 XUI Drawing methods

A B4XCanvas is used for drawing over a B4XView panel.
The drawings will only be updated after a call to Canvas.Invalidate.
Note that you should call Canvas.Release when it is no longer used.
If the hosting view is resized, then the canvas should be released and initialized again.

In the following methods you will find several common parameters.

• Bitmap1 as Bitmap	an Android bitmap
• x, y, x1, y1, x2, y2 As Float	are coordinates, Float variables.
• Color as Int	are color variables. Int variables
• SrcRect, DestRact, Rect1 As Rect	are rectangles, Rect objects
• Filled As Boolean	flag if the surface is filled (True) or not (False)

The most common drawing methods are:

- **DrawBitmap** (Bitmap As B4XBitmap, Destination As B4XRect)
  Draws a bitmap in the given destination. Use B4XBitmap.Crop to draw part of a bitmap.

- **DrawBitmapRotated** (Bitmap As B4XBitmap, Destination As B4XRect, Degrees As Float)
  Similar to DrawBitmap. Draws a rotated bitmap.

- **DrawCircle** (x As Float, y As Float, Radius As Float, Color As Int, Filled As Boolean, StrokeWidth As Float)
  Draws a circle.
  x an y are the centre coordinates of the circle and Radius the circles radius.

- **DrawLine** (x1 As Float, y1 As Float, x2 As Float, y2 As Float, Color As Int, StrokeWidth As Float)
  Draws a straight line.

- **DrawPath** (Path1 As B4XPath, Color As Int, Filled As Boolean, StrokeWidth as Float)
  Draws a path with the given color, filled or not and line width.

- **DrawPathRotated** (Path1 As B4XPath, Color As Int, Filled As Boolean, StrokeWidth As Float, Degrees As Float, CenterX As Float, CenterY As Float)
  Draws a path with the given color, filled or not and line width, rotated by Degrees around the given centre.

- **DrawRect** (Rect As B4XRect, Color As Int, Filled As Boolean, StrokeWidth As Float)
  Draws a rectangle.

- **DrawText** (Text As String, x As Float, y As Float, Font As B4XFont, Color As Int, Alignment As Align Enum)
  Draws the text.
  Text - The text that will be drawn.
  x - The origin X coordinate.
  y - The origin Y coordinate.
  Font - The text font.
  Color - Drawing color.
  Alignment - Sets the alignment relative to the origin. One of the following values: LEFT, CENTER, RIGHT.

- **DrawTextRotated** (Text As String, x As Float, y As Float, Font As B4XFont, Color As Int, Alignment As Align Enum, Degree As Float)
  Similar to DrawText. Rotates the text before it is drawn.

2.4.2 B4A Drawing methods

Canvas is used for drawing over other views.
The drawings will only be updated after a call to ParentView.Invalidate.

In the following methods you will find a number of common parameters.

• Bitmap1 as Bitmap	an Android bitmap
• x, y, x1, y1, x2, y2 As Float	are coordinates, Float variables, use dip values.
• Color as Int	are color variables. Int variables
• SrcRect, DestRact, Rect1 As Rect	are rectangles, Rect objects
• Filled As Boolean	flag if the surface is filled (True) or not (False)

The drawing methods are:

- **DrawBitmap** (Bitmap1 As Bitmap, SrcRect As Rect, DestRect As Rect)
  Draws the given bitmap or only a part of it.
  SrcRect = source rectangle, can be only a part of the original bitmap.
  DestRect = destination rectangle, can be any size.
  To draw with the same size both rectangles must have same width and same height.
  If DestRect is different size than SrcRect the destination drawing is stretched or shrinked depending on the size ratios between the two rectangles.

- **Draw BitmapFlipped** (Bitmap1 As Bitmap, SrcRect As Rect, DestRect As Rect, Vertically As Boolean, Horizontally As Boolean)
  Same method as DrawBitmap, but the image can be flipped vertically and/or horizontally.

- **Draw BitmapRotated** (Bitmap1 As Bitmap, SrcRect As Rect, DestRect As Rect, Degrees As Float)
  Same method as DrawBitmap, but with a rotation of the given Degrees angle around the centre of the bitmap.

- **DrawCircle** (x As Float, y As Float, Radius As Float, Color as Int, Filled As Boolean, StrokeWidth As Float)
  Draws a circle.
  x and y are the centre coordinates of the circle and Radius the circles radius.

- **DrawColor** (Color As Int)
  Fills the whole view with the given color.
  The color can be Colors.Transparent making the whole view transparent.

- **DrawDrawable** (Drawable1 As Drawable, DestRect As Rect)
  Draws a Drawable into the specified rectangle.

- **DrawDrawableRotated** (Drawable1 As Drawable, DestRect As Rect, Degrees As Float)
  Rotates and draws a Drawable into the specified rectangle

- **DrawLine** (x1 As Float, y1 As Float, x2 As Float, y2 As Float, Color as Int, StrokeWidth As Float)
  Draws a straight line.

- **DrawOval** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float)
  Draws a path with the given color, filled or not and line width.
  Draws an oval shape.
  Filled - Whether the rectangle will be filled.
  StrokeWidth - The stroke width. Relevant when Filled = False

- **DrawOvalRotated** (Path1 As Path, Color As Int, Filled As Boolean, StrokeWidth As Float, Degrees As Float)
  Rotates the oval and draws it, filled or not and line width.
  Draws an oval shape.
  Filled - Whether the rectangle will be filled.
  StrokeWidth - The stroke width. Relevant when Filled = False.

- **DrawPath** (Path1 As Path, Color As Int, Filled As Boolean, StrokeWidth As Float)
  Draws a path with the given color, filled or not and line width.

- **DrawPoint** (x As Float, y As Float, Color As Int)
  Draws a point (pixel) at the specified position and color.

- **DrawRect** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float)
  Draws a rectangle with given size, color, filled or not and line width.

- **DrawRectRotated** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float, Degrees As Float)
  Same as DrawRect but rotated by the given angle

- **DrawText** (Text As String, x As Float, y As Float, Typeface1 As TypeFace, TestSize As Float, Color As Int, Align1 As Align)

- **DrawTextRotated** (Text As String, x As Float, y As Float, Typeface1 As TypeFace, TestSize As Float, Color As Int, Align1 As Align, Degrees As Float)

Depending on where and when you use drawing methods in your code you need to add an Invalidate method for the view you draw on.
Redrawing will only happen when the program can process messages. Usually when it finishes running the current code.

- **Invalidate**
  Invalidates the whole view forcing the view to redraw itself.

- **Invalidate2** (Rect As Rect)
  Invalidates the given rectangle.

- **Invalidate3** (Left As Int, Top As Int, Right As Int, Bottom As Int)
  Invalidates the given rectangle.

Example:

```
cvsGraph.DrawLine(0, 0, 100dip, 100dip, Colors.Red, 1dip)
pnlGraph.Invalidate
```

`cvsGraph` is the Canvas and `pnlGraph` is the Panel we draw on.

2.4.3 B4i Drawing methods

Canvas is used for drawing over other views.
The drawings will only be updated after a call to Canvas.Refresh.
Note that you should call Canvas.Release when it is no longer used.
If the hosting view is resized, then the canvas should be released and initialized again.

In the following methods you will find a number of common parameters.

• Bitmap1 as Bitmap	an Android bitmap
• x, y, x1, y1, x2, y2 As Float	are coordinates, Float variables.
• Color as Int	are color variables. Int variables
• SrcRect, DestRact, Rect1 As Rect	are rectangles, Rect objects
• Filled As Boolean	flag if the surface is filled (True) or not (False)

The drawing methods are:

- **DrawBitmap** (Bitmap1 As Bitmap, DestRect As Rect)
  Draws the given bitmap or only a part of it.
  SrcRect = source rectangle, can be only a part of the original bitmap.
  DestRect = destination rectangle, can be any size.
  To draw with the same size both rectangles must have same width and same height.
  If DestRect is different size than SrcRect the destination drawing is stretched or shrinked depending on the size ratios between the two rectangles.

- **Draw BitmapRotated** (Bitmap1 As Bitmap, DestRect As Rect, Degrees As Float)
  Same method as DrawBitmap, but with a rotation of the given Degrees angle around the centre of the bitmap.

- **DrawCircle** (x As Float, y As Float, Radius As Float, Color as Int, Filled As Boolean, StrokeWidth As Float)
  Draws a circle.
  x and y are the centre coordinates of the circle and Radius the circles radius.

- **DrawColor** (Color As Int)
  Fills the whole view with the given color.
  The color can be Colors.Transparent making the whole view transparent.

- **DrawLine** (x1 As Float, y1 As Float, x2 As Float, y2 As Float, Color as Int, StrokeWidth As Float)
  Draws a straight line.

- **DrawPath** (Path1 As Path, Color As Int, Filled As Boolean, StrokeWidth as Float)
  Draws a path with the given color, filled or not and line width.

- **DrawPathRotated** (Path1 As Path, Color As Int, Filled As Boolean, StrokeWidth As Float, Degrees As Float, CenterX As Float, CenterY As Float)
  Draws a path with the given color, filled or not and line width, rotated by Degrees around the given center.

- **DrawPDF** (Document1 As PDDocument, PageNumber As Int, DestRect As Rect)
  Draws a PDF page in the DestRect rectangle.
  Document - PDF Document.
  PageNumber - The page to draw. Note that the first page number is 1.
  DestRect - Destination rectangle.

- **DrawRect** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth as Float)
  Draws a rectangle with given size, color, filled or not and line width.

- **DrawRectRotated** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float, Degrees As Float)
  Same as DrawRect but rotated by the given angle

- **DrawRectRounded** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float, CornerRadius As Float, Degrees As Float)
  Same as DrawRect but rotated by the given angle

- **DrawText** (Text As String, x As Float, y As Float, Typeface1 As TypeFace, TestSize As Float, Color As Int, Align1 As Align)

- **DrawTextRotated** (Text As String, x As Float, y As Float, Typeface1 As TypeFace, TestSize As Float, Color As Int, Align1 As Align, Degrees As Float)

- **DrawView** (View As View, DestRect As Rect)
  Draws the given view in the DestRect rectangle.

2.4.4 B4J Drawing methods

A special node that can be drawn on.
The Canvas node will not be resized automatically when its parent is resized.

In the following methods you will find a number of common parameters.

• Bitmap1 as Bitmap	an Android bitmap
• x, y, x1, y1, x2, y2 As Double	are coordinates, Float variables.
• Paint as Paint	are color variables. Int variables
• Filled As Boolean	flag if the surface is filled (True) or not (False)

The drawing methods are:

- **DrawCircle** (x As Double, y As Double, Radius As Double, Paint As Paint, Filled As Boolean, StrokeWidth As Double)
  Draws a circle.
  x an y are the centre coordinates of the circle and Radius the circles radius.

- **DrawImage** (Image As Image, x As Double, y As Double, Width As Double, Height As Double)
  Draws the given image.
  Image - The image that will be drawn.
  x - The top left corner x coordinate.
  y - The top left corner y coordinate.
  Width - The width of the destination rectangle.
  Height - The height of the destination rectangle.

- **DrawImage2** (Image As Image, SourceX As Double, SourceY As Double, SourceWidth As Double, SourceHeight As Double, DestX As Double, DestY As Double, DestWidth As Double, DestHeight As Double)
  Draws the given image.
  Image - The image that will be drawn.
  SourceX - The top left corner x coordinate of the source rectangle.
  SourceY - The top left corner y coordinate of the source rectangle.
  SourceWidth - The width of the source rectangle.
  SourceHeight - The height of the source rectangle.
  DestX - The top left corner x coordinate of the destination rectangle.
  DestY - The top left corner y coordinate of the destination rectangle.
  DestWidth - The width of the destination rectangle.
  DestHeight - The height of the destination rectangle.

- **DrawImageRotated** (Image As Image, x As Double, y As Double, Width As Double, Height As Double, Degree As Double)
  Draws the given image.
  Image - The image that will be drawn.
  x - The top left corner x coordinate.
  y - The top left corner y coordinate.
  Width - The width of the destination rectangle.
  Height - The height of the destination rectangle.

- **DrawLine** (x1 As Double, y1 As Double, x2 As Double, y2 As Double, Paint As Paint, StrokeWidth As Double)
  Draws a straight line.

- **DrawRect** (x As Double, y As Double, Width As Double, Height As Double, Paint As Paint, Filled As Boolean, StrokeWidth As Double)
  Draws a rectangle with given size, color, filled or not and line width.

- **DrawRectRotated** (x As Double, y As Double, Width As Double, Height As Double, Paint As Paint, Filled As Boolean, StrokeWidth As Double, Degree As Double)
  Same as DrawRect but rotated by the given angle

- **DrawText** (Text As String, x As Double, y As Double, Font As Font, Paint As Paint, Alignment As TextAlignment Enum)
  Draws the text.
  Text - The text that will be drawn.
  x - The origin X coordinate.
  y - The origin Y coordinate.
  Font - The text font.
  Paint - Drawing color.
  TextAlignment - Sets the alignment relative to the origin. One of the following values: LEFT, CENTER, RIGHT.

- **DrawText2** (Text As String, x As Double, y As Double, Font As Font, Paint As Paint, Alignment As TextAlignment Enum, MaxWidth As Double)
  Similar to DrawText. MaxWidth defines the text bounds. The text will be wrapped if it is longer.

- **DrawTextRotated** (Text As String, x As Double, y As Double, Font As Font, Paint As Paint, Alignment As TextAlignment Enum, Degree As Double)
  Similar to DrawText. Rotates the text before it is drawn.

2.5 BitmapCreator

BitmapCreator is a cross platform library.
Its core is made of:

1. A bytes array that represents an image.
2. Platform specific code that efficiently creates a regular bitmap from the bytes array data (Bitmap property).
3. Platform specific code that efficiently extracts the pixels data from a bitmap and copies them to the bytes array (CopyPixelsFromBitmap).

Reading and writing to an array of bytes are very quick operations. This allows low level access that was previously difficult to implement and more or less impossible to implement with cross platform code.

Additional methods in BitmapCreator:

- Methods to get or set a pixel color in various formats.
- Methods to copy a pixel from a different BitmapCreator to this BitmapCreator.
- Methods to draw a bitmap or a different BitmapCreator to this BitmapCreator.
- Method to draw gradients.
- And a few other methods.

Scaling

1. The bitmap created with the Bitmap property has a scale of 1. This means that on Android the OS will resize the bitmap automatically based on the device scale.
Don't use 'dip' units with BitmapCreator.
Note that in the example class the "missing dip" warning is disabled with:
```
#IgnoreWarnings : 6
```

2. Depending on your use case it might be better from performance perspective to create a smaller BitmapCreator and resize the output Bitmap with Bitmap.Resize.

Blending

When a non-opaque pixel (source) is set, the target pixel color should be combined with the source color for the transparency to have its expected effect. This is called blending.
When you call DrawBitmap or DrawBitmapCreator you can choose whether to skip blending or not. It is significantly faster to skip blending, though drawing with blending will be fast enough in most cases.

There is also a BlendPixel method that you can use to blend a single pixel.

BitmapCreator vs. Canvas / B4XCanvas

Use them together. BitmapCreator returns a regular bitmap and Canvas can be used to make a drawing that is then copied or drawn over a BitmapCreator.

Debugging

Many of the methods that you will use with BitmapCreator are computational intensive. The debugger might be too slow, this is especially true if the code is in a resumable sub as the debugger cannot optimize such subs.
As BitmapCreator is cross platform you can write the code in B4J and only test it on the other platforms.

Dependencies:

B4A / B4J - XUI, JavaObject and ByteConverter (https://www.b4x.com/android/forum/threads/6787/#content)
B4i - iXUI, iRandomAccessFile

Updates

Starting from B4J v6.30, B4i v5.0 and B4A v8.3, BitmapCreator is included as an internal library.
Note that it will not work with older versions of B4J, B4i and B4A.

2.5.1 BC Drawing methods (BC = BitmapCreator)

BitmapCreator provides methods to manipulate Bitmaps.

- **DrawArc** (X As Float, Y As Float, Radius As Float, Color As Int, Filled As Boolean, StrokeWidth As Int, StartingAngle As Float, SweepAngle As Float)
  Draws an arc and returns the created brush.
  X / Y - Center position.
  Radius - Arc radius.
  Color - Drawing color.
  Filled - Whether to fill the arc or not.
  StrokeWidth - Stroke width (for non-full shapes).
  StartingAngle - Measured in degrees, starting from hour 3.
  SweepAngle - Measured in degrees. Positive is clockwise.

- **DrawArc2** (X As Float, Y As Float, Radius As Float, Brush As _bcbrush, Filled As Boolean, StrokeWidth As Int, StartingAngle As Float, SweepAngle As
  Similar to DrawArc. Expects a BCBrush instead of a color value.

- **DrawBitmap** (Bitmap1 As B4XBitma, TargetRect As B4XRect , SkipBlending As Boolean)
  Draws a bitmap to the Buffer.
  SkipBlending - Whether non-opaque pixels in the source bitmap should blend with the background.

- **DrawBitmapCreator** (Source As Bitmapcreator, SrcRect As B4XRect, TargetX As Int, TargetY As Int, SkipBlending As Boolean)
  Draws the image stored in the Source BitmapCreator to this BitmapCreator.
  Source - Source BitmapCreator
  SrcRect - Defines the region in the Source BC that will be copied.
  TargetX / TargetY - Top left point in the destination BC
  SkipBlending - Whether non-opaque pixels in the source bitmap should blend with the background.

- **DrawBitmapCreatorsAsync**(Target As Object, EventName As String, DrawTasks As List)
  Asynchronously draws all the draw tasks. Note that no other drawings should be made until the BitmapReady event is raised.
  The 'bmp' parameter will not be initialized in B4J.
  Target must be a class instance.
  Example:
  ```
  bc1.DrawBitmapCreatorsAsync(Me, "BC", Tasks)
  Wait For BC_BitmapReady (bmp As B4XBitmap)
  ```

- **DrawBitmapCreatorTransformed**(Task As DrawTask)
  Draws a scaled and / or rotated BitmapCreator.
  Task.TargetX / TargetY define the target center. This is different than the behavior in other methods.
  Task.SkipBlending = True will treat all pixels with alpha &gt; 0 as solid colors and other pixels as transparent.

- **DrawCircle**(X As Float, Y As Float, Radius As Float, Color As Int, Filled As Boolean, StrokeWidth As Int)
  Draws a circle. Returns the color brush.

- **DrawCircle2**(X As Float, Y As Float, Radius As Float, Brush As BCBrush, Filled As Boolean, StrokeWidth As Int)
  Draws a circle.
  X an Y are the centre coordinates of the circle and Radius the circle radius.
  Brush is a BCBrush

- **DrawCompressedBitmap**(Source As _compressedbc, SrcRect As B4XRect, TargetX As Int, TargetY As Int)
  Draws a CompressedBC object created with BitmapCreator.ExtractCompressBC.

- **DrawLine**(X0 As Float, Y0 As Float, X1 As Float, Y1 As Float, Color As Int, StrokeWidth As Int)
  Draws a line. Returns the color brush.

- **DrawLine2**(X0 As Float, Y0 As Float, X1 As Float, Y1 As Float, Brush As BCBrush, StrokeWidth As Int
  Draws a line.

- **DrawPath**(Path As BCPath, Color As Int, Filled As Boolean, StrokeWidth As Int)
  Draws a BCPath. Don't confuse with B4XPath or Path.
  Returns the created color brush.

- **DrawPath2**(Path As BCPath, Brush As BCBrush, Filled As Boolean, StrokeWidth As Int)
  Draws a BCPath. Don't confuse with B4XPath or Path.
  Non-full paths: Connection segments are drawn between each two lines. You can disable it by setting BCPath.DrawConnectionSegments to False.
  This will make the drawing twice as fast (in most cases it will be fast enough anyway).

- **DrawRect**(Rect As B4XRect, Color As Int, Filled As Boolean, StrokeWidth As Int)
  Draws a rectangle. Returns the color brush.

- **DrawRect2**(Rect As B4XRect, Brush As BCBrush, Filled As Boolean, StrokeWidth As Int)
  Draws a rectangle.

- **DrawRectRounded**(Rect As B4XRect, Color As Int, Filled As Boolean, StrokeWidth As Int, CornersRadius As Int)
  Draws a rectangle with round corners. Returns the color brush.

- **DrawRectRounded2**(Rect As B4XRect, Brush As BCBrush, Filled As Boolean, StrokeWidth As Int, CornersRadius As Int)
  Draws a rectangle with round corners.

- **DrawRotatedCBC**(cbc As CompressedBC, Degrees As Float, Width As Int, Height As Int, AABuffer As InternalAntialiasingBuffer)
  Rotates with antialiasing.

- **FillGradient**(GradColors As Int(), Rect As B4XRect, Orientation As String)
  Fills the rectangle with a gradient. Skips blending.
  GradColors - An array of two or more colors that define the gradient.
  Rect - The region that will be filled.
  Orientation - One of the following: TL_BR, TOP_BOTTOM, TR_BL, LEFT_RIGHT, RIGHT_LEFT, BL_TR, BOTTOM_TOP, BR_TL and RECTANGLE.

- **FillRadialGradient**(GradColors As Int(), Rect As B4XRect)
  Fills the rectangle with a radial gradient. Skips blending.

- **FlipCompressedBitmap**(Source As CompressedBC, Horizontal As Boolean, Vertical As Boolean)
  Flips a compressed BitmapCreator.

2.5.2 BC Other methods or objects

- **ARGBToColor**(ARGB As ARGBColor)
  Converts an ARGB value to a color int value.

- **ARGBToPremultipliedColor**(ARGB As ARGBColor, PMC As PremultipliedColor)
  Converts an ARGB color to PremultipliedColor.
  The Result parameter will hold the output.

- **Buffer** As Byte()
  Gets the internal buffer, read only !
  The buffer is organized in columns and rows.
  Buffer(0) = B of pixel col 0 row 0
  Buffer(1) = G of pixel col 0 row 0
  Buffer(2) = R of pixel col 0 row 0
  Buffer(3) = A of pixel col 0 row 0

  Buffer(4) = B of pixel col 1 row 0
  Buffer(5) = G of pixel col 1 row 0
  Buffer(6) = R of pixel col 1 row 0
  Buffer(7) = A of pixel col 1 row 0

- **GetARGB** (x As Int, y As Int, Result As ARGBColor)
  Gets the color of the given point as an ARGB color
  The Result parameter stores the output.
  ```
  Private imvBitmap As B4XView
  Private col As ARGBColor
  bmcBitmapR.GetARGB(0, 0, col)
  Log(col0.a)       'logs the A value
  Log(col0.r)       'logs the R value
  Log(col0.g)       'logs the G value
  Log(col0.b)       'logs the B value
  ```

- **GetColor**(x As Int, y As Int)
  Gets the color of the given point as an int value.

- **ReplaceSemiTransparentPixels**(NewColor As Int, Rect As B4XRect)
  Replaces all semi-transparent pixels in the Rect with NewColor.
  This can be useful to remove antialiasing effects.

- **SetARGB**(x As Int, y As Int, ARGB As ARGBColor)
  Sets the color of the specified point.

- **SetColor**(x As Int, y As Int, Clr As Int)
  Sets the color of the given pixel.

- **SetHSV**(x As Int, y As Int, alpha As Int, h As Int, s As Float, v As Float)
  Sets the HSV (Hue, Saturation, Value) color of the specified point.

2.5.3 BC Asynchronus Drawing methods

- **AsyncDrawCircle**(X As Float, Y As Float, Radius As Float, Brush As BCBrush, Filled As Boolean, StrokeWidth As Int)
  Asynchronously draws a circle. Returns a DrawTask that should be added to the list of drawing tasks.

- **AsyncDrawLine**(X0 As Float, Y0 As Float, X1 As Float, Y1 As Float, Brush As BCBrush, StrokeWidth As Int)
  Asynchronously draws a line.
  Returns a DrawTask that should be added to the list of drawing tasks.

- **AsyncDrawPath**(Path As BCPath, Brush As BCBrush, Filled As Boolean, StrokeWidth As Int)
  Asynchronously draws a path.
  Returns a DrawTask that should be added to the list of drawing tasks.
  Note that you shouldn't modify the path while it is being drawn. You can use BCPath.Clone to create a copy.

- **AsyncDrawRect**(Rect As B4XRect, Brush As BCBrush, Filled As Boolean, StrokeWidth As Int)
  Asynchronously draws a rectangle.
  Returns a DrawTask that should be added to the list of drawing tasks.
  AsyncDrawRectRounded(Rect As B4XRect, Brush As BCBrush, Filled As Boolean, StrokeWidth As Int, CornersRadius As Int)
  Asynchronously draws a rectangle with round corners.
  Returns a DrawTask that should be added to the list of drawing tasks.

2.5.4 BCBrush BitmapCreator brush

- **BlendAll**

- **BlendBorders**

- **ColorPM**
  Premultiplied color

- **Initialize**
  Initializes the fields to their default value.

- **IsColorSource**

- **IsInitialized**
  Tests whether the object has been initialized.

- **Source**
  Contains : Bitmapcreator

- **SrcOffsetX**
  Contains : Int

- **SrcOffsetY**
  Contains : Int

2.5.5 BCPath BitmapCreator path

There are three types of paths so it can be confusing: Path (native Canvas), B4XPath (B4XCanvas) and BCPath. As you can guess BitmapCreator works with BCPath.
A path defines a list of points.
You can either draw lines between these points or fill the polygon shape.
The shape is closed automatically if needed.

- **Clone**
  Creates a copy of the path. This is used by BitmapCreator.AsyncDrawPath.

- **DrawConnectionSegments**

- **FindBoundingRect**
  Returns the path bounding rectangle.

- **Initialize**(X As Float, Y As Float)
  Initializes the path and sets the first point.

- **InternalCounterClockWise**

- **Invalidate**
  Call Invalidate if you modifed the Points list directly.

- **IsInitialized**
  Tests whether the object has been initialized.

- **LineTo**(X As Float, Y As Float)
  Adds a line to the given point.

- **Points**
  List containg the points.

- **RemoveLastPoint**
  Removes the last point.

- **Reset**(X As Float, Y As Float)
  Resets the path.

2.6 BitmapCreatorEffects b4xlib

The BitmapCreatorEffects b4xlib library contains the following methods:

- **AdjustColors** (Bmp As B4XBitmap, HueOffset As Int, SaturationFactor As Float)
  Returns a B4XBitmap.
- **Blur** (bmp As B4XBitmap) Returns a B4XBitmap.
- **Brightness** (Bmp As B4XBitmap, Factor As Float) Returns a B4XBitmap.
- **DrawOutsideMask** (Bmp As B4XBitmap, Mask As B4XBitmap) Returns a B4XBitmap.
  Assuming that mask is the same size or larger than Bmp.
- **DrawThroughMask** (Bmp As B4XBitmap, Mask As B4XBitmap) Returns a B4XBitmap.
  Assuming that mask is the same size or larger than Bmp.
- **FadeBorders** (bmp As B4XBitmap, Width As Int) Returns a B4XBitmap.
- **FlipHorizontal** (bmp As B4XBitmap) Returns a B4XBitmap.
- **FlipVertical** (bmp As B4XBitmap) Returns a B4XBitmap.
- **GreyScale** (bmp As B4XBitmap) Returns a B4XBitmap.
- **ImplodeAnimated** (Duration As Int, Bmp As B4XBitmap, ImageView As B4XView, PieceSize As Int)
  Returns a ResumableSub.
- **Negate** (bmp As B4XBitmap) Returns a B4XBitmap.
- **Pixelate** (Bmp As B4XBitmap, BoxSize As Int) Returns a B4XBitmap.
- **PixelateAnimated** (Duration As Int, Bmp As B4XBitmap, FromBoxSize As Int, ToBoxSize As Int, ImageView As B4XView)
  Returns a ResumableSub.
- **ReplaceColor** (bmp As B4XBitmap, OldColor As Int, NewColor As Int, KeepAlphaLevel As Boolean)
  Replaces OldColor with NewColor.
  KeepAlphaLevel - If true then the alpha level of the source color is kept.
  Returns a B4XBitmap.
- **TransitionAnimated** (Duration As Int, FromBmp As B4XBitmap, ToBmp As B4XBitmap, ImageView As B4XView)
  Returns a ResumableSub.

2.7 Drawing program First steps

The product specific projects are in:
GraphicsSourceCode\GraphicsFirstSteps \B4A\GraphicsFirstSteps.b4a
GraphicsSourceCode\GraphicsFirstSteps \B4i\iGraphicsFirstSteps.b4i
GraphicsSourceCode\GraphicsFirstSteps \B4J\jGraphicsFirstSteps.b4j

The XUI cross platform projects are in:
XUISourceCode\GraphicsFirstSteps \B4A\xGraphicsFirstSteps.b4a
XUISourceCode\GraphicsFirstSteps \B4i\ixGraphicsFirstSteps.b4a
XUISourceCode\GraphicsFirstSteps \B4J\jxGraphicsFirstSteps.b4a

To draw something, we need a Canvas object which is simply a drawing tool.

XUI
B4XCanvas is a cross platform Canvas, it is a wrapper of the native Canvases of B4A, B4i and B4J.
It must be ‘connected’ to a B4XView object in the Initialize method.
The example program is written as a CustomView with the SAME module for the three products.

B4A, B4i
The Canvas draws onto a Bitmap. This Bitmap can be the background bitmap of views.
The most common views to draw on are: Activity, Panel, ImageView or a Bitmap.

The Canvas must be 'connected' to a bitmap or a view background image in the Initialize method.

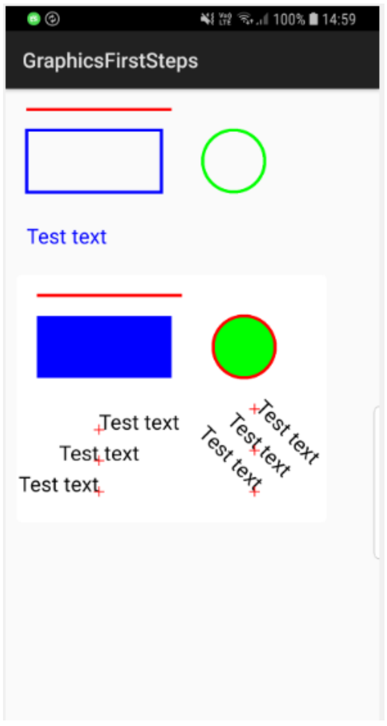
- Initialize(Target View)
- Initialize2(Target Bitmap)   B4A only

If we want to draw on different views at the same time, we need one Canvas for each view.

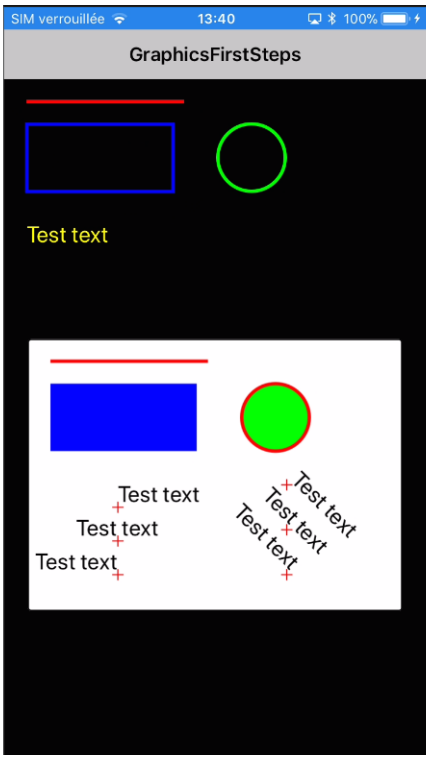
In the example programs we'll use several drawing methods and draw onto the Activity and onto a Panel `pnlGraph` defined in the 'main' layout file. Here we need two canvases.

B4J
The Canvas is a node, it is not related to any other object like in B4A or B4i.
In the example program we'll use several drawing methods and draw onto the Canvas.

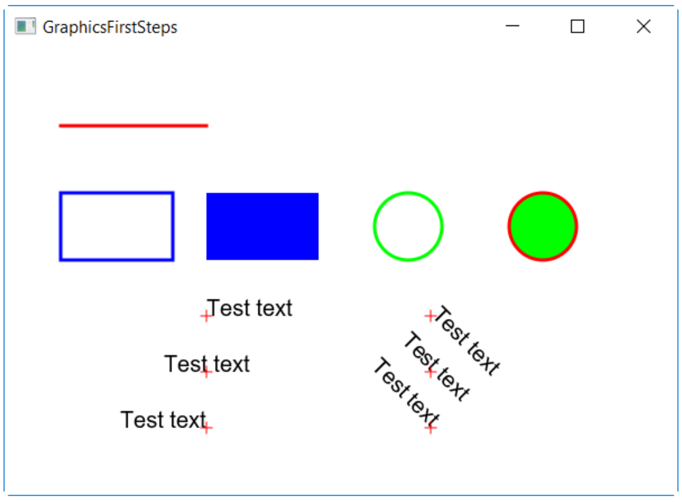
And the result.



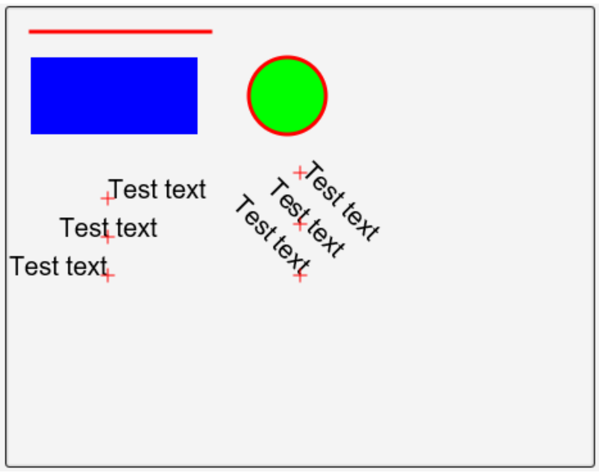
B4A



B4i



B4J



XUI / B4J

2.7.1 Start and initialization

2.7.1.1 XUI

We declare a B4XView, the XUI library, the B4XCanvas and a B4XFont.
This example is made as a CustomView.

```
Sub Class_Globals
	Private mEventName As String 'ignore
	Private mCallBack As Object 'ignore

	Private xui As XUI
	Private xplGraph As B4XView
	Private cvsGraph As B4XCanvas
	Private xFont As B4XFont
End Sub
```

We initialize the B4XFont to draw the text, it is a cross platform xui method.

```
Public Sub Initialize (Callback As Object, EventName As String)
	mEventName = EventName
	mCallBack = Callback

	'define a default font with a size of 20
	xFont = xui.CreateDefaultFont(20)
End Sub
```

Here we set the Base object from the Designer to the B4XView xplGraph and ‘connect’ the B4XCanvas to the B4XView.

```
Public Sub DesignerCreateView (Base As Object, Lbl As Label, Props As Map)
	xplGraph = Base

	cvsGraph.Initialize(xplGraph)
End Sub
```

2.7.1.2 B4A

First, we must declare the different views and objects:
We have:

- the Panel `pnlGraph`
- the Canvas `cvsActivity` for the Activity
- the Canvas `cvsPanel` for the Panel

```
Sub Globals
   Private pnlGraph As Panel
   Private cvsActivity, cvsGraph As Canvas
End Sub
```

Then we must load the layout file, initialize the two Canvases and run the Drawing routine:

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

   ' initialize the Canvas for the activity
   cvsActivity.Initialize(Activity)

   ' initialize the Canvas for the pnlGraph panel
   cvsGraph.Initialize(pnlGraph)

   ' call the Drawing routine
   Drawing
End Sub
```

2.7.1.3 B4i

First we must declare the different views and objects:
We have:

- the Panel `pnlGraph`
- the Canvas `cvsPage` for the Page
- the Canvas `cvsPanel` for the Panel

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

   Private pnlGraph As Panel
   Private cvsPage, cvsGraph As Canvas
End Sub
```

The first three objects are added automatically, needed by the OS.

Then we must load the layout file:

```
Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")

   ' load the layout file
   Page1.RootPanel.LoadLayout("main")
   ' show page 1
   NavControl.ShowPage(Page1)
End Sub
```

And finally, we initialize the two Canvases and run the Drawing routine, we need to do this in the **`Page1_Resize`** routine because the dimensions of the views are only known here:

```
Private Sub Page1_Resize(Width As Int, Height As Int)
   ' initialize the Canvas for the activity
   cvsPage.Initialize(Page1.RootPanel)

   ' initialize the Canvas for the pnlGraph panel
   cvsGraph.Initialize(pnlGraph)

   ' call the Drawing routine
   Drawing
End Sub
```

2.7.1.4 B4J

We declare only the Canvas because it is a node, no need to connect it to another node (object) like in B4A, B4i and XUI:

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

   Private cvsGraph As Canvas
End Sub
```

The first two objects are added automatically, needed by the OS.

Then we load the layout file and call the Drawing routine, no initialization needed:

```
Sub AppStart (Form1 As Form, Args() As String)
   MainForm = Form1
   MainForm.RootPane.LoadLayout("Main") 'Load the layout file.
   MainForm.Show

   Drawing
End Sub
```

2.7.2 Draw a line

Then we draw a horizontal line onto the Activity or MainPage:
The method is :
DrawLine (x1 As Float, y1 As Float, x2 As Float, y2 As Float, Color as Int, StrokeWidth As Float)
Where:

- x1, y1 are the coordinates of the start point in pixels
- x2, y2 are the coordinates of the end point in pixels
- Color is the line color
- StrokeWidth the line thickness in pixels

Then we draw a horizontal line onto pnlGraph with the same coordinates:
The coordinates are relative to the upper left corner of the view we draw on, the Panel `pnlGraph` in this case.

And the code :

2.7.2.1 XUI

```
' draw a horizontal line onto xplGraph
cvsGraph.DrawLine(20dip, 20dip, 160dip, 20dip, xui.Color_Red, 3dip)
```

2.7.2.2 B4A

```
' draw a horizontal line onto the Activity
cvsActivity.DrawLine(20dip, 20dip, 160dip, 20dip, Colors.Red, 3dip)


' draw a horizontal line onto pnlGraph
cvsGraph.DrawLine(20dip, 20dip, 160dip, 20dip, Colors.Red, 3dip)
```

2.7.2.3 B4i

```
' draw a horizontal line onto the Page
cvsPage.DrawLine(20, 20, 160, 20, Colors.Red, 3)

' draw a horizontal line onto pnlGraph
cvsGraph.DrawLine(20, 20, 160, 20, Colors.Red, 3)
```

2.7.2.4 B4J

```
' draw a horizontal line
cvsGraph.DrawLine(20, 20, 150, 20, fx.Colors.Red, 3)
```

2.7.3 Draw a rectangle

Then we draw an empty rectangle onto the Activity or MainPage :
The method is:
DrawRect (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth as Float)
Where :

- Rect1 is a rectangle object
- Color is the border or rectangle color
- Filled: False = only the border is drawn
  True = the rectangle is filled
- StrokeWidth is the line thickness of the border, not relevant when Filled = True

To draw a rectangle, we need a Rect object.
We:

- Define it with the name `rect1`.
- Initialize it with the coordinates of the upper left corner and the coordinates of the lower right corner.
- Draw it

Then we draw a filled rectangle onto pnlGraph with the same coordinates:

We don't need to define nor initialize a new rectangle because the coordinates are the same, so we use the same Rect object.

And the code:

2.7.3.1 XUI

```
' draw a filled rectangle onto xplGraph
Private rect1 As B4XRect
rect1.Initialize(20dip, 40dip, 150dip, 100dip)
cvsGraph.DrawRect(rect1, xui.Color_Blue, True, 3dip)
```

2.7.3.2 B4A

```
' draw an empty rectangle onto the Activity
Private rect1 As Rect
rect1.Initialize(20dip, 40dip, 150dip, 100dip)
cvsActivity.DrawRect(rect1, Colors.Blue, False, 3dip)

' draw a filled rectangle onto pnlGraph
cvsGraph.DrawRect(rect1, Colors.Blue, True, 3dip)
```

2.7.3.3 B4i

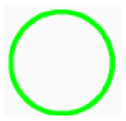
```
' draw an empty rectangle onto the Page
Dim rect1 As Rect
rect1.Initialize(20, 40, 150, 100)
cvsPage.DrawRect(rect1, Colors.Blue, False, 3)

' draw a filled rectangle onto pnlGraph
cvsGraph.DrawRect(rect1, Colors.Blue, True, 3)
```

2.7.3.4 B4J

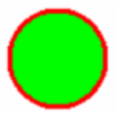
No need for a Rect object. The coordinates of the upper left corner and the width and height are given directly in the method.

```
' draw an empty rectangle
cvsGraph.DrawRect(20, 80, 100, 60, fx.Colors.Blue, False, 3)

' draw a filled rectangle
cvsGraph.DrawRect(150, 80, 100, 60, fx.Colors.Blue, True, 3)
```

2.7.4 Draw a circle

Then we draw an empty circle onto the Activity or MainPage : 
The method is :
DrawCircle (x As Float, y As Float, Radius As Float, Color as Int, Filled As Boolean, StrokeWidth As Float)
Where :

- x, y are the coordinates of the center in pixels.
- Radius is the radius in pixels.
- Color is the border or circle color
- Filled: False = only the border is drawn True = the circle is filled
- StrokeWidth is the line thickness of the border, not relevant when Filled = True

Then we draw a filled circle with a border with a different color on the panel. 

There is no direct method to draw a filled circle with a border with a different color.
So, we first draw the filled circle and then the circle border in two steps.
Instead of using fixed values like 220dip we can also use variables like in the code below.
When a same value is used several times it's better to use variables because if you need to change the value you change it only once the value of the variable all the rest is changed automatically by the variable.

And the code:

2.7.4.1 XUI

```
' draw a filled circle with a boarder onto xplGraph
Private centerX, centerY, radius As Float
centerX = 220dip
centerY = 70dip
radius = 30dip
cvsGraph.DrawCircle(centerX, centerY, radius, xui.Color_Green, True, 3dip)
cvsGraph.DrawCircle(centerX, centerY, radius, xui.Color_Red, False, 3dip)
```

2.7.4.2 B4A

```
' draw an empty circle onto the Activity
cvsActivity.DrawCircle(220dip, 70dip, 30dip, Colors.Green, False, 3dip)

' draw a filled circle with a boarder onto pnlGraph
Private centerX, centerY, radius As Float
centerX = 220dip
centerY = 70dip
radius = 30dip
cvsGraph.DrawCircle(centerX, centerY, radius, Colors.Green, True, 3dip)
cvsGraph.DrawCircle(centerX, centerY, radius, Colors.Red, False, 3dip)
```

2.7.4.3 B4i

```
' draw an empty circle onto the Page
cvsPage.DrawCircle(220, 70, 30, Colors.Green, False, 3)

' draw a filled circle with a boarder onto pnlGraph
Dim centerX, centerY, radius As Float
centerX = 220
centerY = 70
radius = 30
cvsGraph.DrawCircle(centerX, centerY, radius, Colors.Green, True, 3)
cvsGraph.DrawCircle(centerX, centerY, radius, Colors.Red, False, 3)
```

2.7.4.4 B4J

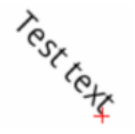
```
' draw an empty circle
cvsGraph.DrawCircle(330, 110, 30, fx.Colors.Green, False, 3)

' draw a filled circle with a boarder
Private centerX, centerY, radius As Double
centerX = 450
centerY = 110
radius = 30
cvsGraph.DrawCircle(centerX, centerY, radius, fx.Colors.Green, True, 3)
cvsGraph.DrawCircle(centerX, centerY, radius, fx.Colors.Red, False, 3)
```

2.7.5 Draw a text

Then we draw a text.

The method is:
DrawText (Text As String, x As Float, y As Float, Typeface1 As TypeFace, TestSize As Float, Color As Int Align1 As Align)
Where:

- Text is the text to draw
- x, y are the coordinates of the reference point (depending on the Align1 value) in pixels. The reference point is on the texts baseline.
- TypeFace1 is the text style
- TextSize is the text size in a typographic unit called 'point'.
  The text size is independent of the screen density !
  Don't use dip values !
- Color is the text color
- Align1 is the alignment of the text according to the refence point.
  Possible values: "LEFT", "CENTER", "RIGHT".



Then we draw a rotated text onto pnlGraph.

And we draw a cross on the reference point to show where it is and how the align does work.
The method is DrawTextRotated, it's the same as DrawText but with an additional parameter Degrees, the rotation angle.
Instead of using fiyed dip values in the routine we define three variables:

refX and refY	the coordinates of the reference point.
hl	the half of the cross line length.

And the code:

2.7.5.1 XUI

```
Private refX, refY, hl As Float
refX = 80dip
refY = 150dip
hl = 5dip

' draw texts with three alignments onto xplGraph
cvsGraph.DrawText("Test text", refX, refY, xFont, xui.Color_Black, "LEFT")
DrawCross(refX, refY, hl)

' draw a cross on the reference point
cvsGraph.DrawLine(x - l, y, x + l, y, xui.Color_Red, 1dip)
cvsGraph.DrawLine(x, y - l, x, y +l, xui.Color_Red, 1dip)
```

2.7.5.2 B4A

```
' draw a text onto the Activity
cvsActivity.DrawText("Test text", 20dip, 150dip, Typeface.DEFAULT, 20, _
Colors.Yellow, "LEFT")

Private refX, refY, hl As Float
refX = 150dip
refY = 180dip
hl = 5dip
' draw a rotated text onto pnlGraph
cvsGraph.DrawTextRotated("Test text", refX, refY, Typeface.DEFAULT, _
20, Colors.Black, "RIGHT", 45)
DrawCross(refX, refY, hl)

' draw a cross on the reference point
cvsGraph.DrawLine(refX - hl, refY, refX + hl, refY, Colors.Red, 1dip)
cvsGraph.DrawLine(refX, refY - hl, refX, refY + hl, Colors.Red, 1dip)
```

2.7.5.3 B4i

```
' draw a text onto the Page
cvsPage.DrawText("Test text", 20, 150, Font.CreateNew(20), Colors.Yellow, "LEFT")

Dim refX, refY, hl As Float
refX = 80
refY = 150
hl = 5
' draw texts with three alignments onto cvsGraph
cvsGraph.DrawText("Test text", refX, refY, Font.CreateNew(20), Colors.Black, "LEFT")
DrawCross(refX, refY, hl)

' draw a cross on the reference point
cvsGraph.DrawLine(x - l, y, x + l, y, Colors.Red, 1)
cvsGraph.DrawLine(x, y - l, x, y +l, Colors.Red, 1)
```

2.7.5.4 B4J

```
Private refX, refY, hl As Double
refX = 150
refY = 190
hl = 5
' draw texts with the three alignments
Private Font1 As Font
Font1 = fx.CreateFont("Arial", 20, False, False)
cvsGraph.DrawText("Test text", refX, refY, Font1, fx.Colors.Black, "LEFT")
DrawCross(refX, refY, hl)     ' draw a cross on the reference point

' draw a cross on the reference point
cvsGraph.DrawLine(x - l, y, x + l, y, fx.Colors.Red, 1)
cvsGraph.DrawLine(x, y - l, x, y +l, fx.Colors.Red, 1)
```

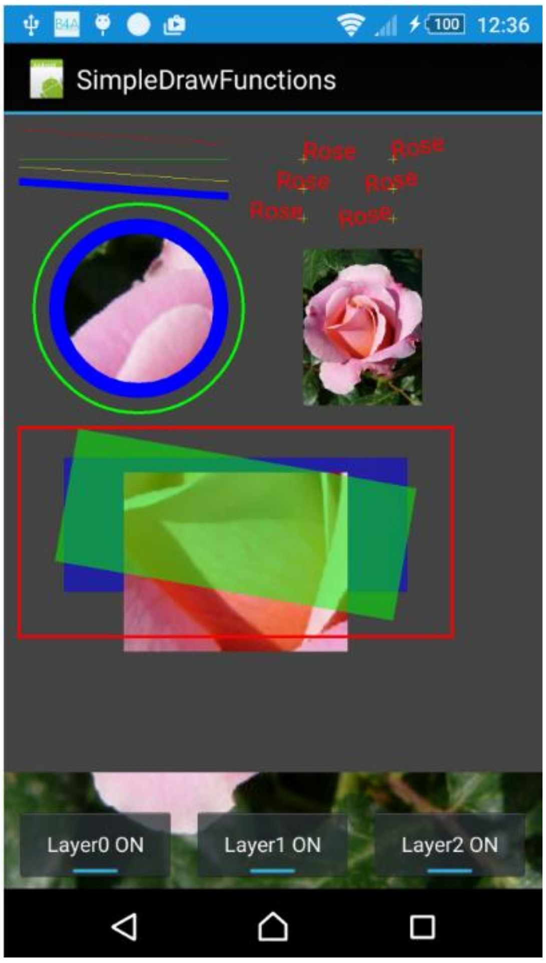
2.8 Drawing program SimpleDrawMethods

In the second drawing program, SimpleDrawMethods, we use the other common drawing methods.

The projects are in: GraphicsSourceCode\Graphics\SimpleDrawMethods directory.
There is one project for each product: B4A, B4i, B4J and XUI.

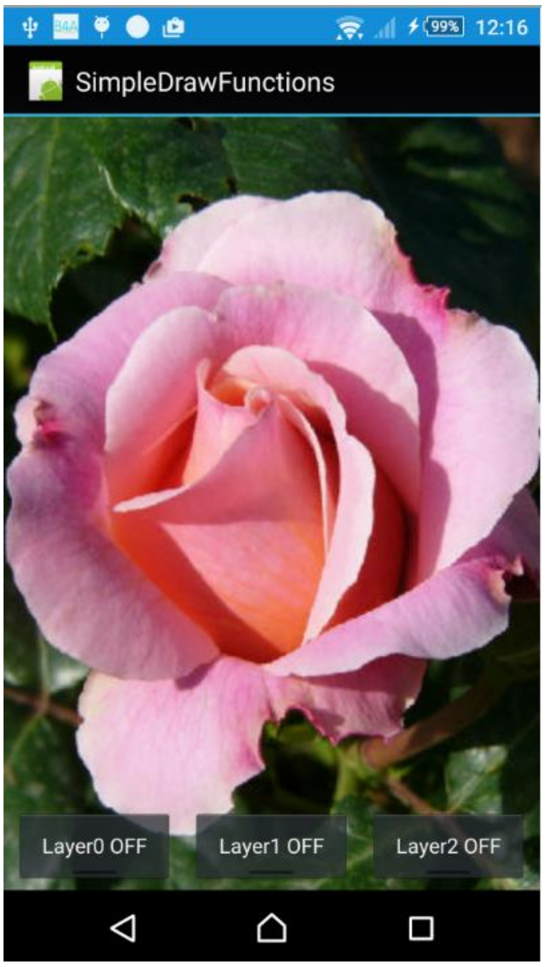
The program has no other purpose than to show what can be done with drawings.

The program has three Panels which we use as layers and three ToggleButtons allowing us to show or hide each layer.
Layer(0) has a grey background and the two other layers have a transparent background.



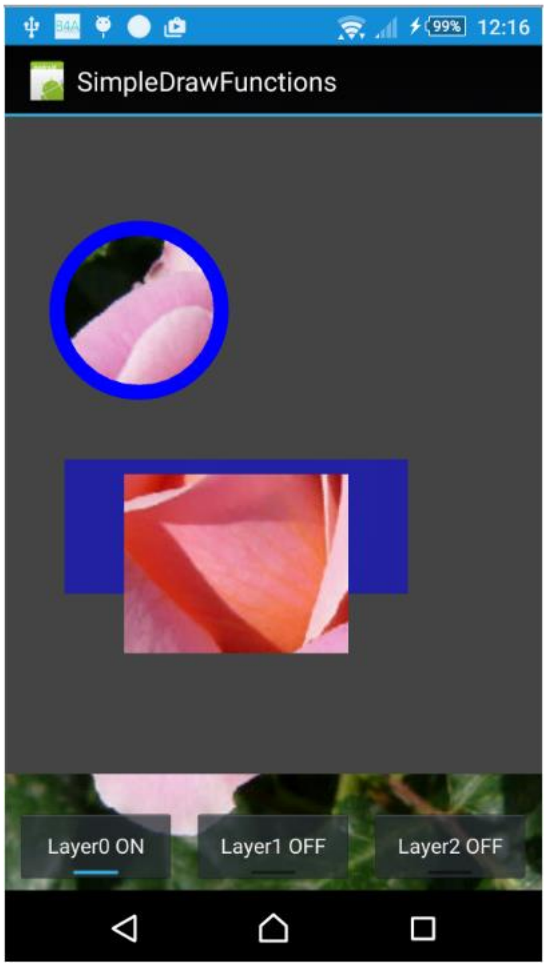
You can play with the buttons to observe the different combinations of visible and hidden layers.

Only the B4A version is shown.



In this screenshot we solely see the background image of the activity.

We use the ToggleButtons to either show or hide the different layers.

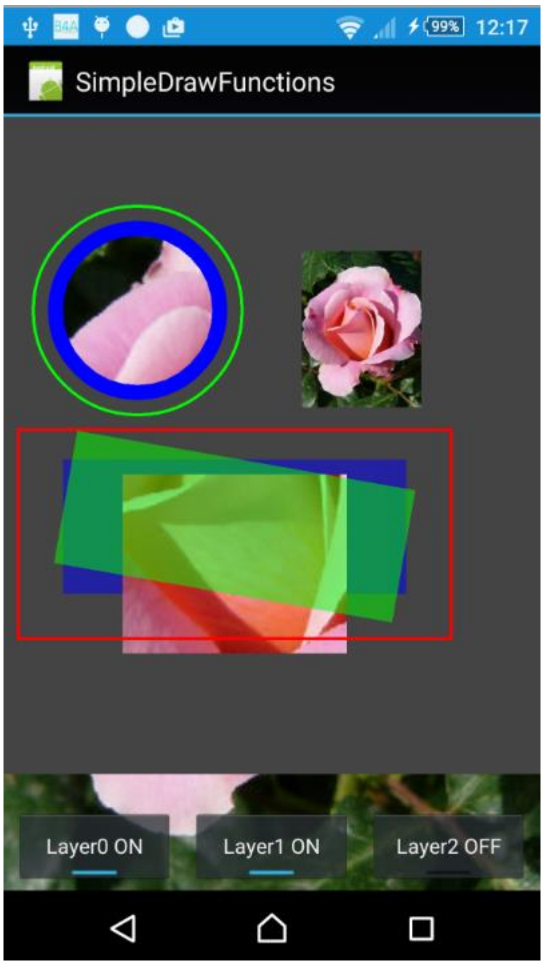


Here we show only layer(0).

The panel has a dark gray background with:

- a blue circle.
- a transparent circle, the activity's background is inside this circle.
- a blue rectangle
- a transparent rectangle, the activity's background is inside this rectangle.

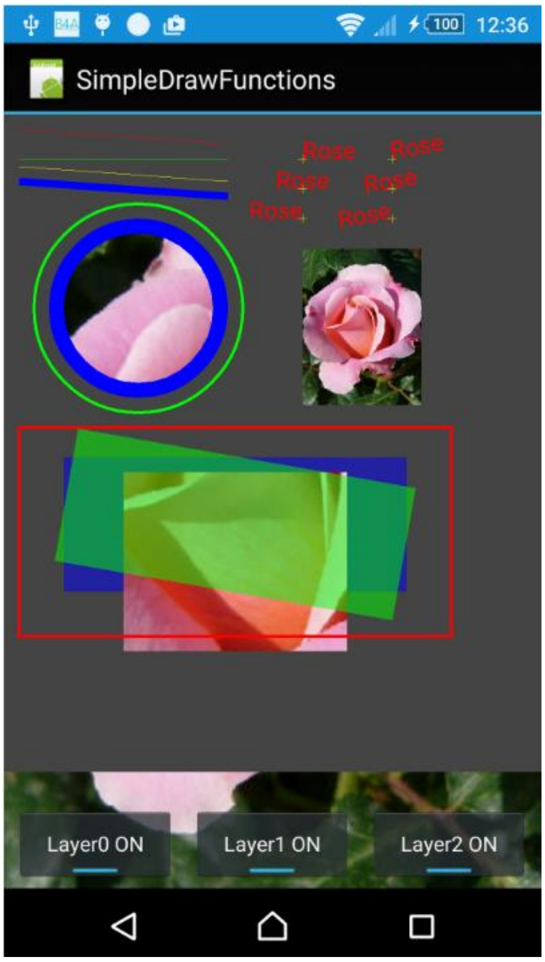
Touching the screen and moving the finger moves the blue and transparent circles on layer(0).



Here we show layer(0) plus layer(1).

The panel has a transparent background with:
- a green circle.
- a small copy of the activity's background image.
- a green, rotated semi-transparent rectangle.

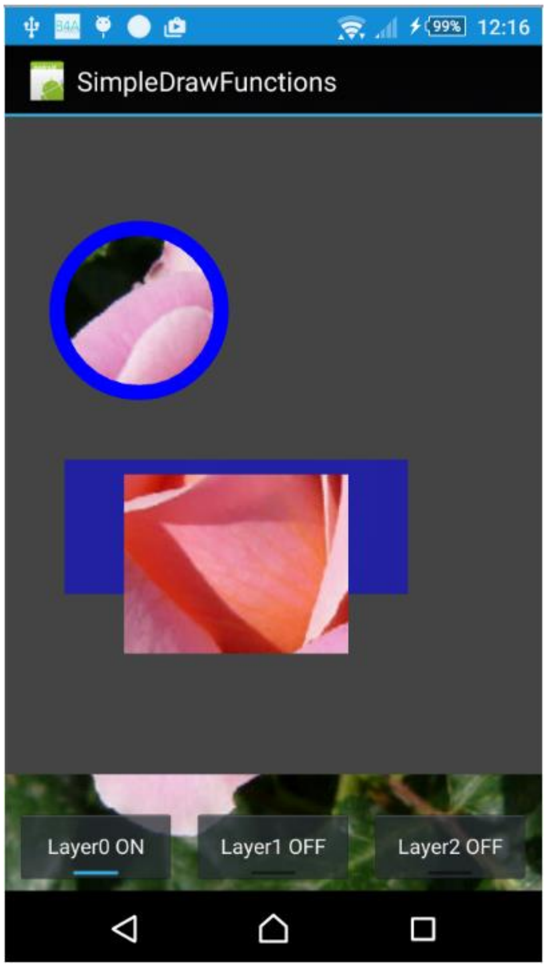
We see that the rectangle covers the activity's background because layer 1 is in front of layer 0.



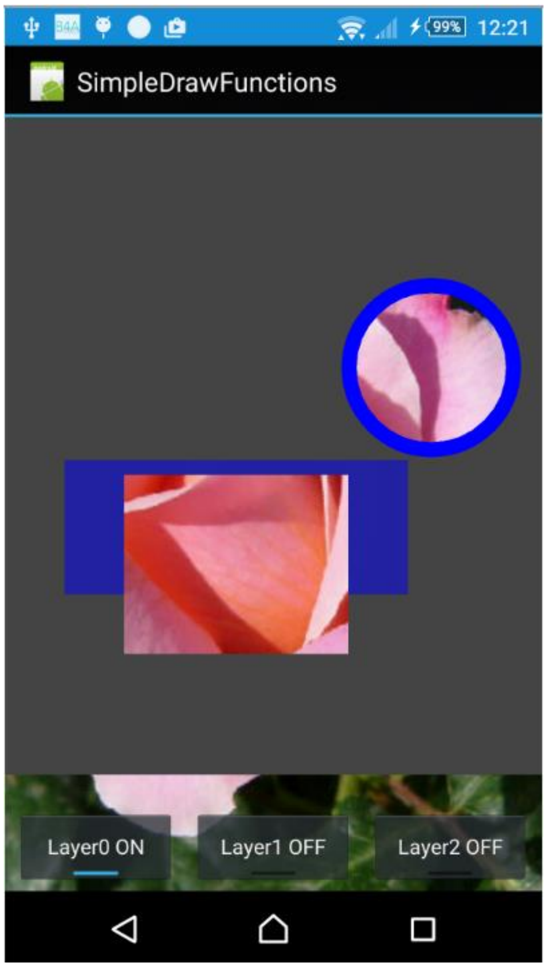
Here we show all three layers.

The panel has a transparent background with:
- 4 lines on top.
- 3 horizontal texts with the three different alignments.
- 3 rotated texts with the three different alignments.
- a point for each text showing the position of the reference point.

You can play with the buttons to show the different combinations of visible and hidden layers.



Touching the screen with the finger and moving it, moves the blue and transparent circles.



On each move, the background image of the activity appears.

Analysis of the code:

There is no layout file, all views are added by code.

In the Process_Globals routine we declare the bitmap.

```
Sub Process_Globals
  Private bmpBackground As Bitmap
End Sub
```

In the Sub Globals routine we declare the different views and variables:

```
Sub Globals
  Private pnlLayer(3) As Panel
  Private cvsLayer(3) As Canvas
  Private btnLayer(3) As ToggleButton
  Private rect1 As Rect
  Private bdwBackground As BitmapDrawable
  Private xc, yc, x1, y1, x2, y2, r1, r2, h, w As Float
End Sub
```

We have:

- 3 Panels
- 3 Canvases
- 3 ToggleButtons
- 1 Rect, rectangle used to draw rectangles
- 1 Bitmap, holding the activity's background image
- 1 BitmapDrawable, holds the activity's background
- different variables used for the drawing.

Note that we use arrays of views for the three panels, canvases and togglebuttons.
`Private pnlLayer(3) As Panel` instead of `Private pnlLayer0, pnlLayer1, pnlLayer2 As Panel`.

In the Sub Activity_Create routine we initialize the different views and add them to the activity:

```
Sub Activity_Create(FirstTime As Boolean)
  Private i As Int

  If FirstTime Then
    bmpBackground.Initialize(File.DirAssets,"Rose2.jpg")
  End If
  bdwBackground.Initialize(bmpBackground)
  Activity.Background = bdwBackground
```

We:

- initialize the views only if FirstTime = True.
- load the `Rose2.jpg` image file into the bitmap.
- initialize the background image of the activity.
- set the activity's background image

```
    x1 = 2%x
    w = 30%x
    y1 = 100%y - 55dip
    h = 50dip
```

- initialize some variables.

```
     For i = 0 To 2
       pnlLayer(i).Initialize("pnlLayer" & i)
       Activity.AddView(pnlLayer(i), 0, 0, 100%x, 85%y)
       cvsLayer(i).Initialize(pnlLayer(i))
       pnlLayer(i).Tag = i

       btnLayer(i).Initialize("btnLayer")
       x2 = x1 + i * 33%x
       Activity.AddView(btnLayer(i), x2, y1, w, h)
       btnLayer(i).TextOn = "Layer" & i & " ON"
       btnLayer(i).TextOff = "Layer" & i & " OFF"
       btnLayer(i).Checked = True
       btnLayer(i).Tag = i
     Next
   End If
 End Sub
```

In a loop we:

- initialize the layer Panels.
  we define an individual EventName for each of the three Panels
  we use only the event for pnlLayer0.
- add the panels to the activity.
- initialize the layer Canvases.
- set the Panels Tag property to the index.

- initialize the layer ToggleButtons.
  we define a single EventName for all three ToggleButtons.
  we manage the showing and hiding of the Panels in one single event routine.
- calculate the left coordinate for each ToggleButton.
- set the texts for the two states.
- set the Checked property to True.
- set the Tag property to the index.

In the Sub Activity_Resume routine we call the Drawing routine.

```
Sub Activity_Resume
  Drawing
End Sub
```

In the Sub Drawing routine we:

```
Sub Drawing
   cvsLayer(0).DrawColor(Colors.DarkGray)
   cvsLayer(1).DrawColor(Colors.Transparent)
   cvsLayer(2).DrawColor(Colors.Transparent)
```

- draw the layout(0) background dark grey.
- draw the layout(1) and layout(2) background transparent.

```
   x1 = 10dip
   y1 = 10dip
   x2 = 150dip
   y2 = 20dip
   cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Red, 0)
   y1 = 30dip
   y2 = 30dip
   cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Green, 0.99dip)
   y1 = 35dip
   y2 = 45dip
   cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Yellow, 0.99dip)
   y1 = 45dip
   y2 = 55dip
   cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Blue, 5dip)
```

- draw four lines onto layer(2)
  `cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Red, 0)`
  the last StrokeWidth parameter is '0', this means hairline mode, the width is one pixel.
  `cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Green, 0.99dip)`
  here we use 0.99dip instead of 1dip because in some cases no line or only parts of it are drawn. This is a known bug in Android with a StrokeWidth of '1'.

```
   xc = 90dip
   yc = 130dip
   r1 = 70dip
   cvsLayer(1).DrawCircle(xc, yc, r1, Colors.Green, False, 2dip)
   r1 = 60dip
   cvsLayer(0).DrawCircle(xc, yc, r1, Colors.Blue, True, 3dip)
   r2 = 50dip
   cvsLayer(0).DrawCircle(xc, yc, r2, Colors.Transparent, True, 1dip)
```

- draw a green circle line on layer(1).
- draw a filled blue circle on layer(0).
- draw a filled transparent circle on layer(0).

```
   rect1.Initialize(10dip, 210dip, 300dip, 350dip)
   cvsLayer(1).DrawRect(rect1, Colors.Red, False, 2dip)
   rect1.Initialize(40dip, 230dip, 270dip, 320dip)
   cvsLayer(0).DrawRect(rect1, Colors.ARGB(128, 0, 0, 255), True, 2dip)
   cvsLayer(1).DrawRectRotated(rect1, Colors.ARGB(128, 0, 255, 0),True, 2dip,10)
   rect1.Initialize(80dip, 240dip, 230dip, 360dip)
   cvsLayer(0).DrawRect(rect1, Colors.Transparent, True, 2dip)
```

- define the coordinates of a rectangle.
- draw a red rectangle on layer(1).
- define the coordinates of a rectangle.
- draw a semi-transparent blue rectangle on layer(0).
- draw a semi-transparent green rotated rectangle on layer(1).
- define the coordinates of a rectangle.
- draw a transparent rectangle on layer(0).
- define the coordinates of a rectangle.
- draw a red rectangle on layer(1).

```
   rect1.Initialize(200dip, 90dip, 280dip, 195dip)
   cvsLayer(1).DrawBitmap(bmpBackground,Null,rect1)
```

Note: `Null` as the source rectangle means the whole bitmap.

- define the coordinates of a rectangle.
- draw the activity's background image in a smaller rectangle on layer(1)

```
   x1 = 200dip
   y1 = 30dip
   cvsLayer(2).DrawText("Rose", x1, y1, Typeface.DEFAULT,16,Colors.Red,"LEFT")
   DrawCross(x1, y1, Colors.Yellow)
   y1 = 50dip
   cvsLayer(2).DrawText("Rose", x1, y1, Typeface.DEFAULT,16,Colors.Red,"CENTER")
   DrawCross(x1, y1, Colors.Yellow)
   y1 = 70dip
   cvsLayer(2).DrawText("Rose", x1, y1, Typeface.DEFAULT,16,Colors.Red,"RIGHT")
   DrawCross(x1, y1, Colors.Yellow)
```

- draw the text "Rose" with the three different possible alignments.
- draw the reference point for each text.

```
   x1 = 260dip
   y1 = 30dip
   cvsLayer(2).DrawTextRotated("Rose", x1,y1,Typeface.DEFAULT,16,Colors.Red,"LEFT",-10)
   DrawCross(x1, y1, Colors.Yellow)
   y1 = 50dip
   cvsLayer(2).DrawTextRotated("Rose", x1,y1,Typeface.DEFAULT,16,Colors.Red,"CENTER",-10)
   DrawCross(x1, y1, Colors.Yellow)
   y1 = 70dip
   cvsLayer(2).DrawTextRotated("Rose", x1,y1,Typeface.DEFAULT,16,Colors.Red,"RIGHT",-10)
   DrawCross(x1, y1, Colors.Yellow)
End Sub
```

- same as above but rotated texts.

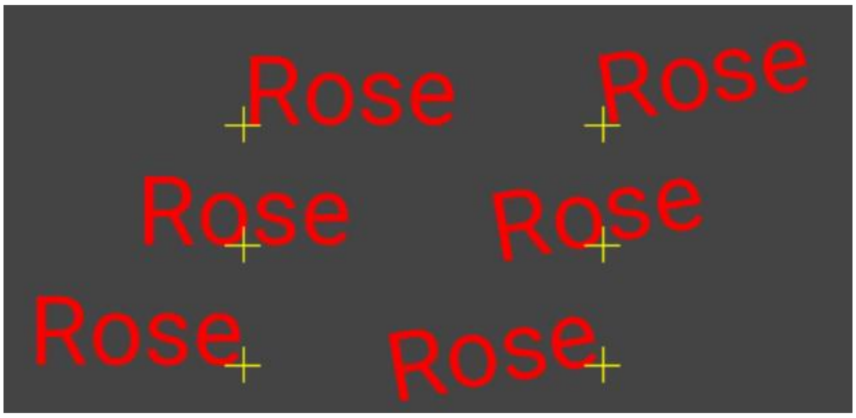
The DrawCross routine:

```
   Sub DrawCross(x As Int, y As Int, color As Int)
      Private d = 3dip As Int

      cvsLayer(2).DrawLine(x - d, y, x + d, y, color, 1)
      cvsLayer(2).DrawLine(x, y - d, x, y + d, color, 1)
   End Sub
```

Looking closer on the displayed texts we see the reference point for each text.

```
   cvsLayer(2).DrawText("Rose", x1, y1, Typeface.DEFAULT,16,Colors.Red,"LEFT")
   DrawCross(x1, y1, Colors.Yellow)
```

These are the x1 and y1 coordinates used to display the texts.



LEFT alignment.

CENTER alignment.

RIGHT alignment.

In the Sub btnLayer_Checked routine we:

- declare a local Button to get the view that raised the event.
- set Send to the Sender view
- change the Visible property from True to False or from False to True.

```
Sub btnLayer_CheckedChange(Checked As Boolean)
   Private Send As Button

   Send = Sender
   pnlLayer(Send.Tag).Visible = Not(pnlLayer(Send.Tag).Visible)
End Sub
```

In the Sub pnlLayer0_Touch routine we:

- draw a dark gray circle to erase the previous blue and transparent circle.
- set and yc to the new coordinates of the circle centres.
- draw a blue and transparent circle on layer(1).
- invalidate pnlLayout(1) to force the update of the drawing.

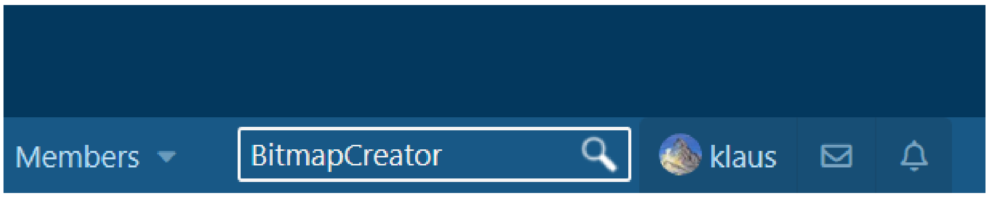
```
Sub pnlLayer0_Touch (Action As Int, X As Float, Y As Float)
   cvsLayer(0).DrawCircle(xc, yc, r1, Colors.DarkGray, True, 3dip)
   xc = X
   yc = Y
   cvsLayer(0).DrawCircle(xc, yc, r1, Colors.Blue, True, 3dip)
   cvsLayer(0).DrawCircle(xc, yc, r2, Colors.Transparent, True, 1dip)
   pnlLayer(0).Invalidate
End Sub
```

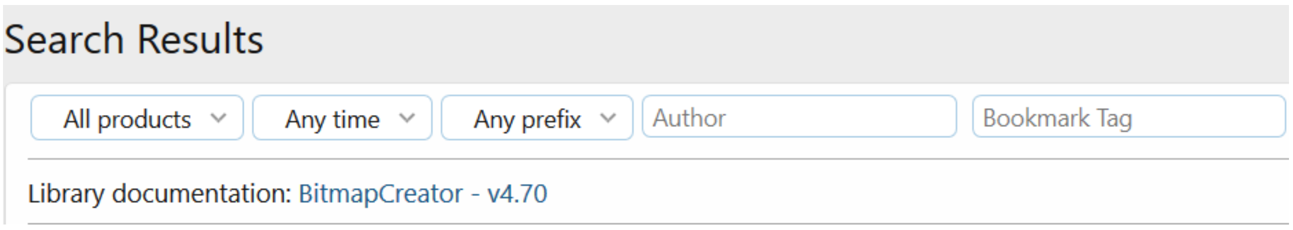
2.9 BitmapCreator Example programs

Erel has written some demo projects

- [B4X] [XUI] BitmapCreator - Pixels, Drawings and More
- [B4X] BitmapCreator Effects

To find more examples, enter [B4X] [BitmapCreator] in the Search field.



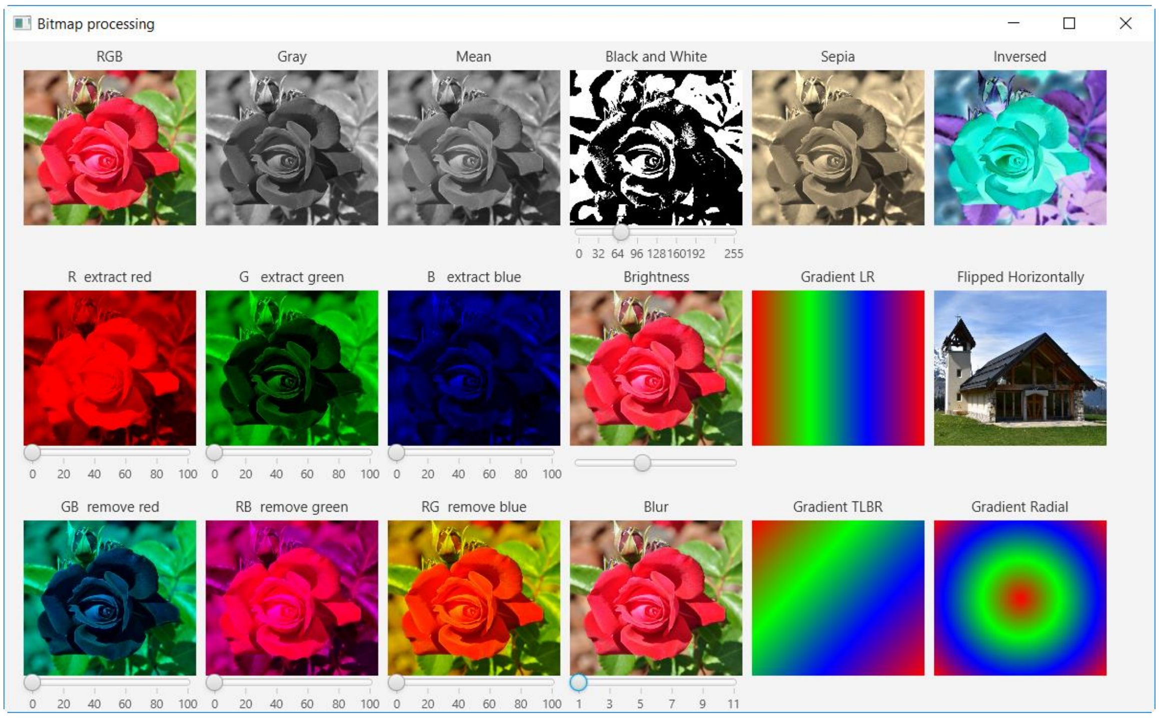

Search Results

Library documentation: BitmapCreator - v4.70

I have also played a bit with BitmapCreator in the two example programs.

2.9.1 BitmapCreatorDemo1

The source code is in the *GraphicsSourceCode\BitmapCreator1* folder.
Only the B4J program is shown, all the routines are valid for B4A and B4i.



The demo program shows different image processing routines:

• RGB	source image.
• Gray	gray scale.
• Mean	colors mean values R = G = B = (R + G + B) / 3.
• Black and White	black and white with a given threshold.
• Sepia	colors converted to sepia.
• Inversed	R = 255 - R, G = 255 - G, B = 255 - B.
• R extract red	extracts the Red part.
• GB remove red	removes the Red part and leaves the Green and Blue parts.
• G extract green	extracts the Green part.
• RB remove green	removes the Green part and leaves the Red and Blue parts.
• B extract blue	extracts the Blue part.
• RG remove blue	removes the Blue part and leaves the Red and Green parts.
• Brightness	changes the brightness.
• Blur	blurs the image.
• Gradient LR	Left - Right gradient.
• Gradient TLBR	Top Left - Bottom Right gradient.
• Gradient radial	radial gradient.
• Flip	flipped horizontally, vertically and both.

Not all routines will be explained, only two, many are similar.

2.9.1.1 Remove green

The goal is to remove the green color.

```
'remove the Green part
Public Sub RemoveGreen (Image As B4XBitmap,  Value As Double) As B4XBitmap
   Private x, y As Int
   Private col0, col1 As ARGBColor
   Private Fact As Double
   Private bmcImage, bmcResult As BitmapCreator

   bmcImage.Initialize(Image.Width, Image.Height)
   bmcImage.CopyPixelsFromBitmap(Image)
   bmcResult.Initialize(Image.Width, Image.Height)

   Fact = Value / 100
   For y = 0 To Image.Height -1
      For x = 0 To Image.Width -1
         bmcImage.GetARGB(x, y, col0)
         col1.a = col0.a
         col1.r = col0.r
         col1.g = col0.g * Fact
         col1.b = col0.b
         bmcResult.SetARGB(x, y, col1)
      Next
   Next
   Return bmcResult.Bitmap
End Sub
```

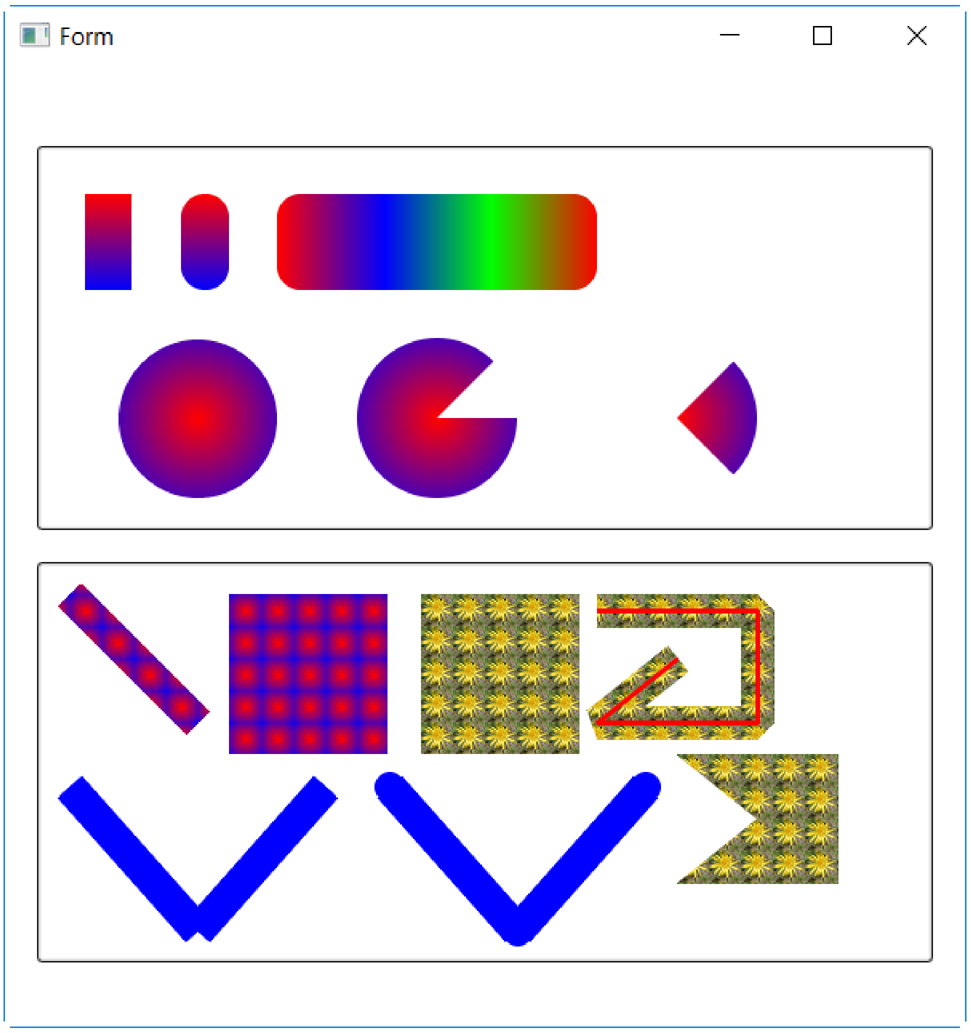
Fill gradient Top-Bottom, FillGradientTB

```
Private Sub FillGradientTB(Colors() As Int, Rect As B4XRect) As B4XBitmap
   Private k, x, y As Int
   Private col, col1, col2 As ARGBColor
   Private y1, fr, fg, fb As Double
   Private bmcResult As BitmapCreator

   bmcResult.Initialize(Rect.Width, Rect.Height)

   y1 = Rect.Height / (Colors.Length - 1)
   For y = 0 To Rect.Height - 1
      k = Min(Floor(y / y1), Colors.Length - 2)
      bmcResult.ColorToARGB(Colors(k), col1)
      bmcResult.ColorToARGB(Colors(k + 1), col2)
      fr = (col2.r - col1.r) / y1
      fg = (col2.g - col1.g) / y1
      fb = (col2.b - col1.b) / y1
      col.a = col1.a
      col.r = col1.r + fr * (y - y1 * k)
      col.g = col1.g + fg * (y - y1 * k)
      col.b = col1.b + fb * (y - y1 * k)
      For x = 0 To Rect.Width - 1
         bmcResult.SetARGB(x, y, col)
      Next
   Next
```

2.9.2 BitmapCreatorDemo2

This program uses some other methods from BitmapCreator, like BCPath and BCBrush.

There is only a B4J program, the principles are exactly the same for B4A and B4i.

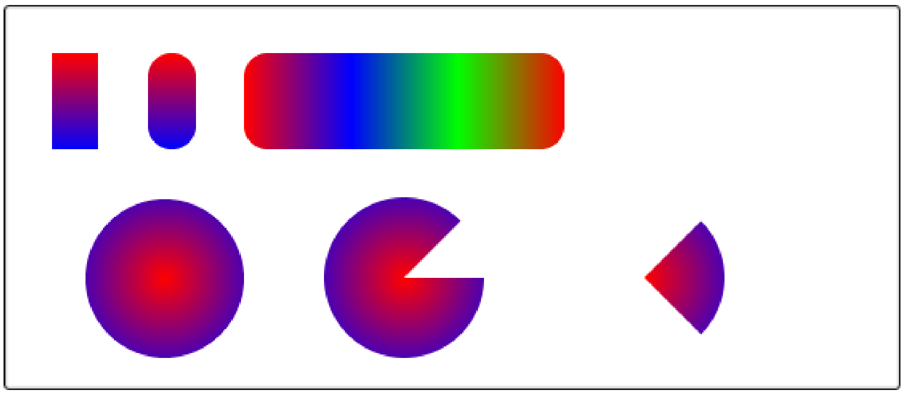


The upper rectangle is B4XView (Pane / Panel), the shapes are drawn with a B4XCanvas.
With Canvas.DrawBitmap.
The routines are in a Code module, called from the main module.

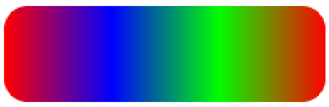
The lower rectangle is a B4XView (ImageView), the drawings are done with BitmapCreator methods onto the BitmapCreator.Bitmap and this one is set to the B4XView, which must be an ImageView.

This shows two drawing possibilities for BitmapCreator.

2.9.2.1 Drawing with B4XCanvas



RoundRectGradient

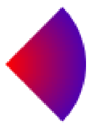
The calling code for: 

```
   'Draws a gradient round rectangle with four colors
   rect1.Initialize(150, 30, 350, 90)
   GradientColors = Array As Int(xui.Color_Red, xui.Color_Blue, xui.Color_Green, xui.Color_Red)
   Draw.RoundRectGradient(cvsTest, rect1, 15, GradientColors, "LEFT_RIGHT")
```

And the drawing routine, I hope that the comments are enough self-explaining:

```
'Draws a rounded rectangle with gradient colors
'cvs - B4XCanvas
'Rect - The region that will be filled.
'Radius - corner radius
'GradColors - An Array of two or more colors that define the gradient.
'Orientation - One of the following: TL_BR, TOP_BOTTOM, TR_BL, LEFT_RIGHT, RIGHT_LEFT,
BL_TR, BOTTOM_TOP, BR_TL And RECTANGLE
Public Sub RoundRectGradient(cvs As B4XCanvas, Rect As B4XRect, Radius As Float,
GradColors() As Int, Orientation As String)
   'Inner rectangle for the bitmap
   Private rct1 As B4XRect
   rct1.Initialize(0, 0, Rect.Width, Rect.Height)

   'BitmapCreator object for the gradient
   Private Gradient As BitmapCreator
   Gradient.Initialize(rct1.Width, rct1.Height)
   Gradient.FillGradient(GradColors, rct1, Orientation)

   'Create the gradient as a brush
   Private GradientBrush As BCBrush = Gradient.CreateBrushFromBitmapCreator(Gradient)

   'Create the gradient rectangle as a bitmap
   Private BC As BitmapCreator
   BC.Initialize(rct1.Width, rct1.Height)
   BC.DrawRectRounded2(BC.TargetRect, GradientBrush, True, 0, Radius)

   'Draw the bitmap
   cvs.DrawBitmap(BC.Bitmap, Rect)
End Sub
```

ArcRadialGradient

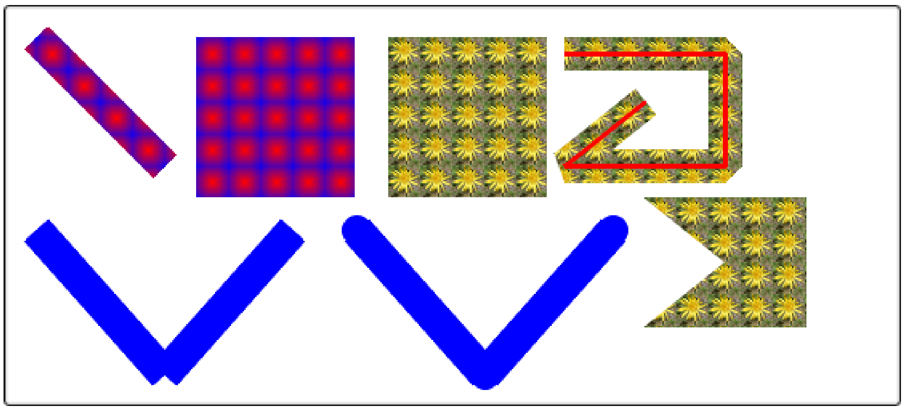
The calling routine for: 

```
   'Draw a radial gradient arc
   GradientColors = Array As Int(xui.Color_Red, xui.Color_Blue)
   Draw.ArcRadialGradient(cvsTest, 400, 170, 50, -45, 90, GradientColors)
```

And the drawing routine, I hope that the comments are enough self-explaining:

```
'Draws an arc with radial gradient
'cvs - B4XCanvas
'CenterX and CenterY - center coordinates
'Radius - circle radius
'StartAngle - starting angle
'SwipeAngle - swipe angle
'GradColors - An Array of two or more colors that define the gradient.
Public Sub ArcRadialGradient(cvs As B4XCanvas, CenterX As Float, CenterY As Float,
Radius As Float, StartAngle As Float, SwipeAngle As Float, GradColors() As Int)
   'Outer rectangle of the circle with destination (outer) coordinates
   Private rectCircle As B4XRect
   rectCircle.Initialize(CenterX - Radius, CenterY - Radius, CenterX + Radius, CenterY +
Radius)

   'Outer rectangle of the circle with inner coordinates
   Private rb As B4XRect
   rb.Initialize(0, 0, 2 * Radius, 2 * Radius)

   'Path for the arc
   Private mPath As B4XPath
   mPath.InitializeArc(CenterX, CenterY, Radius, StartAngle, SwipeAngle)

   'Set the clipping
   cvs.ClipPath(mPath)

   'BitmapCreator object for the final bitmap
   Private BC As BitmapCreator
   BC.Initialize(2 * Radius, 2 * Radius)
   BC.FillRadialGradient(GradColors, rb)
   cvs.DrawBitmap(BC.Bitmap, rectCircle)

   cvs.RemoveClip
End Sub
```

2.9.2.2 Drawing with BitmapCreator methods



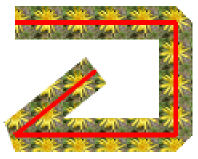
I think that the code in the project is enough self-explanatory.

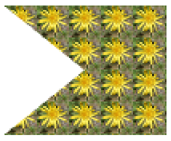
A few comments:

- The left blue V is drawn with two individual lines.
  We see that the joint is not optimal.

- The second blue V is also drawn with two individual lines.
  Three circles have been added to improve the start, the end and the joint.

- We could have drawn the V as a path.
  The start and end would be the same as in the left V and the joints like in the upper right path.

- This shape is an open path  .

- This shape is a closed path  , the relative coordinates of both are the same.

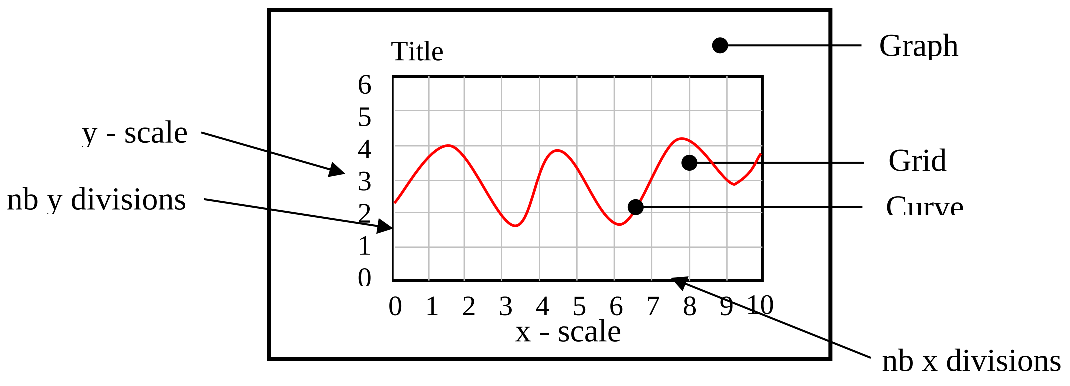
3 Graphs

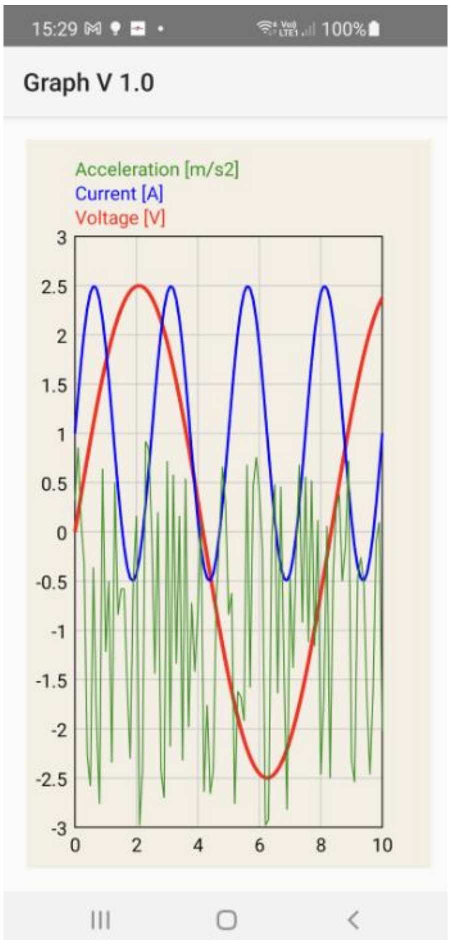
Two-dimensional data can be drawn on a plane with a Cartesian coordinate system.

A Cartesian coordinate system in two dimensions (also called a rectangular coordinate system) is defined by an ordered pair of perpendicular lines (axes), a single unit of length for both axes, and an orientation for each axis. The point where the axes meet, is taken as the origin for both, thus turning each axis into a number line.
In mathematical illustrations of two-dimensional Cartesian systems, the first coordinate (traditionally called the abscissa or x - axis) is measured along a horizontal axis, oriented from left to right. The second coordinate (the ordinate or y - axis) is then measured along a vertical axis, usually oriented from bottom to top (source Wikipedia).

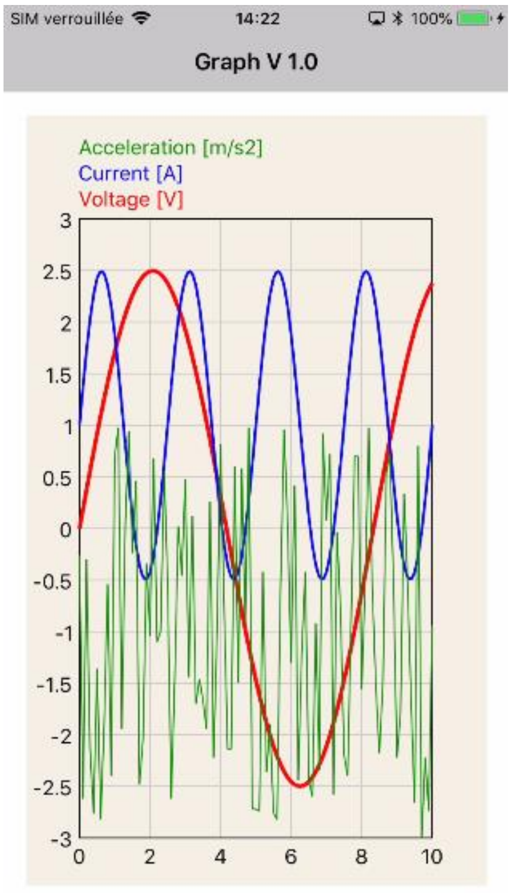
The source code of this example program, *Graph*, is found in the GraphicsSourceCode folder.

We use a B4XView Panel and a B4XCanvas to draw the graphics, we call it the *graph*.
On the *graph* we have the *grid*, the surface where the curve is drawn.

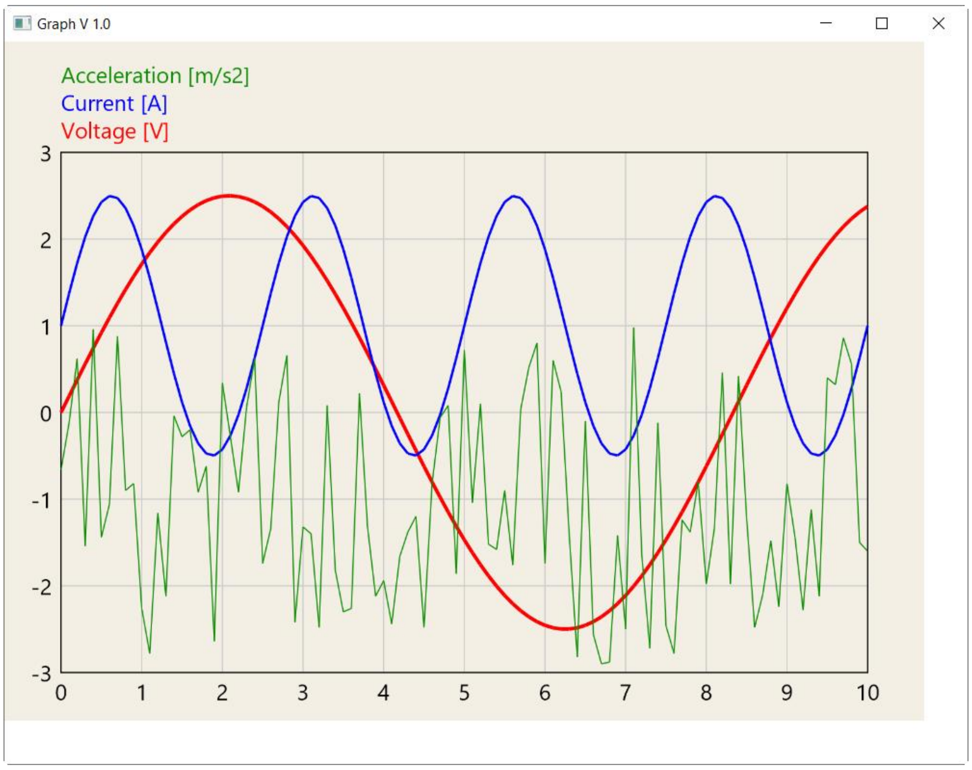



15:29
100%
Graph V 1.0
Acceleration [m/s2]
Current [A]
Voltage [V]
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
2
4
6
8
10

B4A


SIM verrouillée
14:22
100%
Graph V 1.0
Acceleration [m/s2]
Current [A]
Voltage [V]
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
2
4
6
8
10

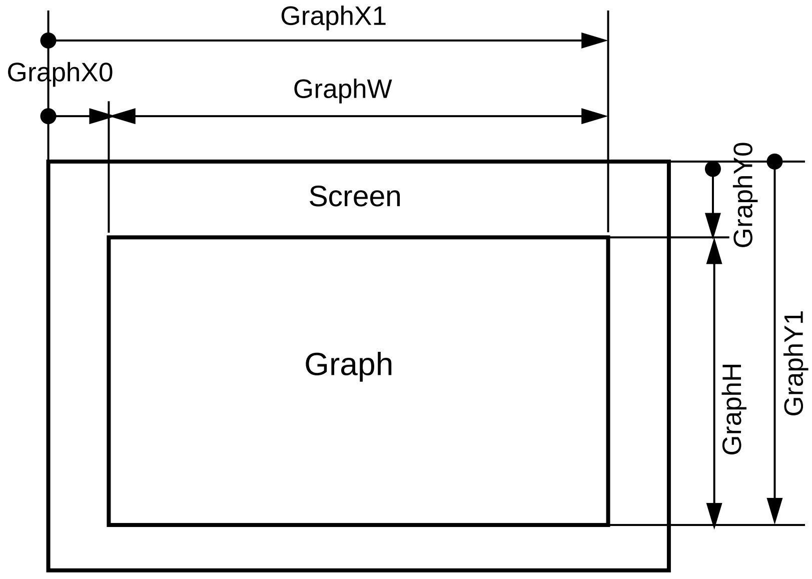
B4i


Graph V 1.0
Acceleration [m/s2]
Current [A]
Voltage [V]
3
2
1
0
-1
-2
-3
0
1
2
3
4
5
6
7
8
9
10

B4J

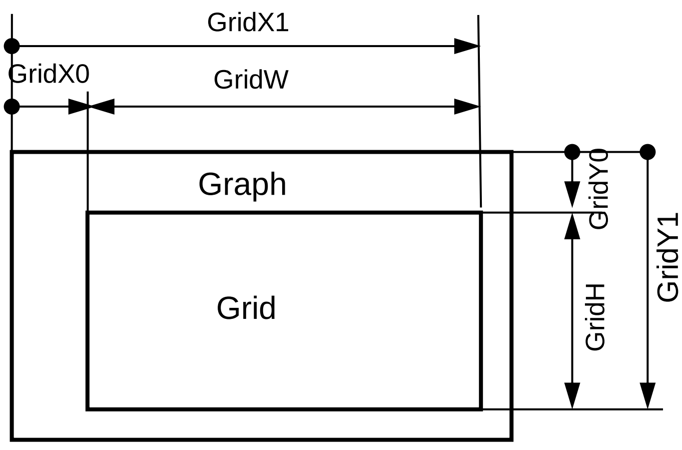
3.1 Definition of the Graph variables

The Graph variables are the position and size of the B4XPanel on the screen.



3.2 Definition of the Grid variables

The Grid variables are the position and size of the grid insides the B4XPanel.
The dimensions are relative to the B4XPanel and not relative to the screen!



3.3 Source code

The code is commented, and I think (hope) enough self-explanatory.
It should not be considered as a project for itself but more as a demonstrator.

The code may seem complicated, but it really is not. I prefer to use a lot of variables rather than constants because they have a meaning, and it is easier to maintain or modify the code later-on.

3.3.1 Product specific code

The project is a B4XPages project, the product specific code is in the Main module.
Which is the default Main module when you create a B4XPages project, nothing to modify.

The code in the B4XMainPage module is the same for all three platforms!

All the common variables and B4X objects are declared in the Class_Globals routine.
The general initialization is done in the B4XPage_Create routine.
The object and graph sizing and the drawing is done in the B4XPage_Created routine.

3.3.2 Common code B4XMainPage module

3.3.2.1 B4X objects declararions

The B4XPages objects are declared on top of the Class_Globals routine.

```
Sub Class_Globals
   Private Root As B4XView
   Private xui As XUI

   'B4X objects
   Private xpnlGraph As B4XView
   Private xcvsGraph As B4XCanvas
   Private rectGraph As B4XRect          ' Graph rectangle
   Private rectGrid As B4XRect           ' Grid rectangle
   Private ScaleTextFont As B4XFont
   Private CurveTextFont As B4XFont
```

3.3.2.2 Common variables

All the common variables are also declared the Class_Globals routine.

```
' Common variables
Public ProgName = "Graph" As String
Public ProgVersion = "V 1.0" As String

' Graph variables
Public GraphX0 As Int                       ' left position of the Graph
Public GraphY0 As Int                       ' top position of the Graph
Public GraphX1 As Int                       ' right position of the Graph
Public GraphY1 As Int                       ' bottom position of the Graph
Public GraphW As Int                        ' width of the Graph
Public GraphH As Int                        ' height of the Graph
Public GraphColor As Int                    ' Graph background color

' Grid variables
Public GridX0 As Int                        ' left position of the Grid
Public GridY0 As Int                        ' top position of the Grid
Public GridX1 As Int                        ' right position of the Grid
Public GridY1 As Int                        ' bottom position of the Grid
Public GridW As Int                         ' width of the Grid
Public GridH As Int                         ' height of the Grid
Public GridNbDivX As Int                    ' number of Grid X divisions
Public GridNbDivY As Int                    ' number of Grid Y divisions
Public GridDeltaX As Int                    ' Grid X division
Public GridDeltaY As Int                    ' Grid Y division
Public GridColor As Int                     ' Grid background color
Public GridLineColor As Int                 ' Grid line color
Public GridFrameColor As Int                ' Grid frame color

' Curve variables
Public CurveNb = 3 As Int                   ' number of curves
Public CurveNbPoints = 100 As Int           ' number of points in the curve
' array of curve point values,
' first index, curve index
' second index, point index
Public Curve(CurveNb, CurveNbPoints + 1) As Double
Public CurveLineColor(CurveNb) As Int
Public CurveLineStroke(CurveNb) As Float
Public CurveName(CurveNb) As String
Public CurveUnit(CurveNb) As String
Public CurveTextSize As Float
Public CurveTextHeight As Float

' Scale variables
Public ScaleX As Double                     ' X scale, ratio pixels / physic unit
Public ScaleY As Double                     ' Y scale, ratio pixels / physic unit
Public ScaleXDelta As Double                ' width of one X division in physic units
Public ScaleYDelta As Double                ' width of one Y division in physic units
Public ScaleXMin As Double                  ' min X physical scale value
Public ScaleYMin As Double                  ' min Y physical scale value
Public ScaleXMax As Double                  ' max X physical scale value
Public ScaleYMax As Double                  ' max Y physical scale value
Public ScaleTextColor As Int
Public ScaleTextSize As Float
Public ScaleTextHeight As Float

'Screen variables
Public ScreenWidth As Int
Public ScreenHeight As Int
```

3.3.2.3 ClearGraph

Clears the graph, draws a rectangle with the GraphColor.

```
Sub ClearGraph
	' clear the graph, draw a rectangle with the background color
	xcvsGraph.DrawRect(rectGraph, GraphColor, True, 1)
End Sub
```

3.3.2.4 InitGraph

Initializes the graph.

```
Sub InitGraph
	' initialize the Graph variables
	
	' all dimensions are expressed in % of height
	GraphX0 = .03 * ScreenHeight
	GraphW = ScreenWidth - 2 * GraphX0
	GraphX1 = GraphX0 + GraphW
	
	GraphY0 = GraphX0
	GraphH = ScreenHeight - 2 * GraphY0
	GraphY1 = GraphY0 + GraphH
	
	xpnlGraph.Left = GraphX0
	xpnlGraph.Top = GraphY0
	xpnlGraph.Width = GraphW
	xpnlGraph.Height = GraphH
	xcvsGraph.Resize(GraphW, GraphH)
	
	' get the min value of the xcvsGraph canvas width or height
	Private GraphSize As Int
	GraphSize = Min(xcvsGraph.TargetRect.Width, xcvsGraph.TargetRect.Height)
	GraphSize = GraphSize / xui.Scale   'needed for B4A, adapts to dip values
	
	' set curve text size according to the screen size
	CurveTextSize = (1 + (GraphSize - 250)/1000) * 14
	CurveTextFont = xui.CreateDefaultFont(CurveTextSize)
	CurveTextHeight = MeasureTextHeight("Ag",  CurveTextFont) + 2dip
	
	' set scale text size according to the screen size
	ScaleTextSize = (1 + (GraphSize - 250)/1000) * 14
	ScaleTextFont = xui.CreateDefaultFont(ScaleTextSize)
	ScaleTextHeight = MeasureTextHeight("Ag", ScaleTextFont) + 2dip
	
	rectGraph.Initialize(0, 0, GraphW, GraphH)
	GraphColor = xui.Color_RGB(244, 239, 228)
End Sub
```

There is a difference between B4A and B4i / B4J.
To calculate the ‘standardized’ graph size, we need to consider the scale in B4A.
This done with the code below, which remains the same for all platforms:

```
	GraphSize = GraphSize / xui.Scale
```

3.3.2.5 InitGrid

Initializes the grid variables.

```
Sub InitGrid
  ' initialize the Grid variables
  ' all dimensions are expressed proportional to the curves text height

  ' define the number of divisions for each axis
  If ScreenWidth > ScreenHeight Then
      GridNbDivX = 10
      GridNbDivY = 6
  Else
      GridNbDivX = 5
      GridNbDivY = 12
  End If

  ' horzontal dimensions
  GridX0 = 2 * CurveTextHeight
  GridW = GraphW - 2 * GridX0

  ' calculate the division dimensions in pixels
  GridDeltaX = GridW / GridNbDivX
  GridW = GridDeltaX * GridNbDivX
  GridX0 = (GraphW - GridW) / 2
  GridX1 = GridX0 + GridW

  ' verical dimensions
  GridY0 = 4 * CurveTextHeight
  GridH = GraphH - GridY0 - 1.5 * CurveTextHeight
  GridDeltaY = GridH / GridNbDivY
  GridH = GridDeltaY * GridNbDivY
  GridY1 = GridY0 + GridH



  ' calculate the division dimensions in pixels
  GridDeltaX = GridW / GridNbDivX
  GridDeltaY = GridH / GridNbDivY

  ' assign the grid rectangle
  rectGrid.Initialize(GridX0, GridY0, GridX1, GridY1)

  ' set the different colors
  GridColor = xui.Color_White
  GridLineColor = xui.Color_LightGray
  GridFrameColor = xui.Color_Black

  ScaleTextColor = xui.Color_Black
End Sub
```

3.3.2.6 InitScale

Intializes the scale values.

```
Sub InitScale
   ' initilize the scales according to the grid dimensions.
   ScaleXDelta = ScaleXMax / GridNbDivX
   ScaleX = GridW / (ScaleXMax - ScaleXMin)

   ScaleYDelta = (ScaleYMax - ScaleYMin) / GridNbDivY
   ScaleY = GridH / (ScaleYMax - ScaleYMin)
End Sub
```

3.3.2.7 InitCurves

Intializes the curve property values.

```
Sub InitCurves
   ' set curve line color
   CurveLineColor(0) = xui.Color_Red
   CurveLineColor(1) = xui.Color_Blue
   CurveLineColor(2) = xui.Color_RGB(10, 140, 0)

   ' set curve line width (stroke)
   CurveLineStroke(0) = 3dip
   CurveLineStroke(1) = 2dip
   CurveLineStroke(2) = 1dip
End Sub
```

3.3.2.8 DrawGrid

Draws the graph grid.

```
Sub DrawGrid
  ' draw the Grid
  Private i As Int
  Private x0, y0 As Float

  ' draw vertical lines
  For i = 1 To GridNbDivX - 1
    x0 = GridX0 + i * GridDeltaX
    xcvsGraph.DrawLine(x0, GridY0, x0, GridY1, GridLineColor, 1dip)
  Next

  ' draw horizontal lines
  For i = 1 To GridNbDivY - 1
    y0 = GridY0 + i * GridDeltaY
    xcvsGraph.DrawLine(GridX0, y0, GridX1, y0, GridLineColor, 1dip)
  Next

  ' draw the frame
  xcvsGraph.DrawRect(rectGrid, GridFrameColor, False, 1dip)

  ' draw the scales
  DrawScaleY
  DrawScaleX

  ' invalidate (update) the Graph
  xcvsGraph.Invalidate
End Sub
```

3.3.2.9 DrawScaleX

Draws the X scale values.

```
Sub DrawScaleX
  ' draw X scale
  Private i As Int
  Private txt As String
  Private x, y As Float

  y = GridY1 + ScaleTextHeight
  For i = 0 To GridNbDivX
    txt = (ScaleXMin + i * ScaleXDelta)
    x = GridX0 + i * GridDeltaX
    xcvsGraph.DrawText(txt, x, y, ScaleTextFont, ScaleTextColor, "CENTER")
  Next
End Sub
```

3.3.2.10 DrawScaleY

Draws the X scale values.

```
Sub DrawScaleY
  ' draw Y scale
  Private i As Int
  Private txt As String
  Private x, y As Float

  x = GridX0 - ScaleTextHeight / 3
  For i = 0 To GridNbDivY
    txt = (ScaleYMax - i * ScaleYDelta)
    y = GridY0 + ScaleTextHeight/3 + i * GridDeltaY
    xcvsGraph.DrawText(txt, x, y, ScaleTextFont, ScaleTextColor, "RIGHT")
  Next
End Sub
```

3.3.2.11 DrawCuves

Draws the curves.

```
Sub DrawCurves(i As Int)
  ' draw the curve of index i
  Private n  As Int
  Private d, x0, y0, x1, y1 As Float

  ' draw the curve
  x0 = GridX0
  y0 = GridY0 + (ScaleYMax - Curve(i, 0)) * ScaleY
  d = GridW / CurveNbPoints
  For n = 1 To CurveNbPoints
    x1 = GridX0 + n * d
    y1 = GridY0 + (ScaleYMax - Curve(i, n)) * ScaleY
    xcvsGraph.DrawLine(x0, y0, x1, y1, CurveLineColor(i), CurveLineStroke(i))
    x0 = x1
    y0 = y1
  Next

  ' draw curve name
  y0 = GridY0 - CurveTextHeight / 2
  xcvsGraph.DrawText(CurveName(i) & " " & CurveUnit(i), GridX0, y0 - i *
CurveTextHeight, CurveTextFont, CurveLineColor(i), "LEFT")
End Sub
```

3.3.2.12 InitCurveValues

Initializes the numeric values for the curves.

```
Sub InitCurveValues
   Private i, n As Int
   Private t As Double
   Private Amplitude(CurveNb) As Double
   Private Offset(CurveNb) As Double
   Private Omega(CurveNb) As Double

   ' set curve amplitude
   Amplitude(0) = 2.5
   Amplitude(1) = 1.5
   Amplitude(2) = .02

   ' set curve offset
   Offset(0) = 0
   Offset(1) = 1
   Offset(2) = -1

   ' set curve omega
   Omega(0)= 2.4 * cPI
   Omega(1)= 8 * cPI

   ' calculate curve point values
   For i = 0 To CurveNb - 1
      For n = 0 To CurveNbPoints
         t = n / 100
         If i = 2 Then
            Curve(i, n) = Offset(i) + Amplitude(i) * Rnd(-100, 100)
         Else
            Curve(i, n) = Offset(i) + Amplitude(i) * Sin(Omega(i) * t)
         End If
      Next
   Next

   ' set curve names and units
   CurveName(0) = "Voltage"
   CurveUnit(0) = "[V]"
   CurveName(1) = "Current"
   CurveUnit(1) = "[A]"
   CurveName(2) = "Acceleration"
   CurveUnit(2) = "[m/s2]"

   ' set scale values
   ScaleXMax = CurveNbPoints / 10
   ScaleXMin = 0

   ScaleYMax = 3
   ScaleYMin = -3
End Sub
```

3.3.2.13 MeasureTextWidth

Measures the width of the given text, it is not used in this project.

```
Private Sub MeasureTextWidth(Text As String, Font1 As B4XFont) As Int
   Private rct As B4XRect
   rct = xcvsGraph.MeasureText(Text, Font1)
   Return rct.Width
End Sub
```

3.3.2.14 MeasureTextHeight

Measures the height of the given text.

```
Private Sub MeasureTextHeight(Text As String, Font1 As B4XFont) As Int
   Private rct As B4XRect
   rct = xcvsGraph.MeasureText(Text, Font1)
   Return rct.Width
End Sub
```

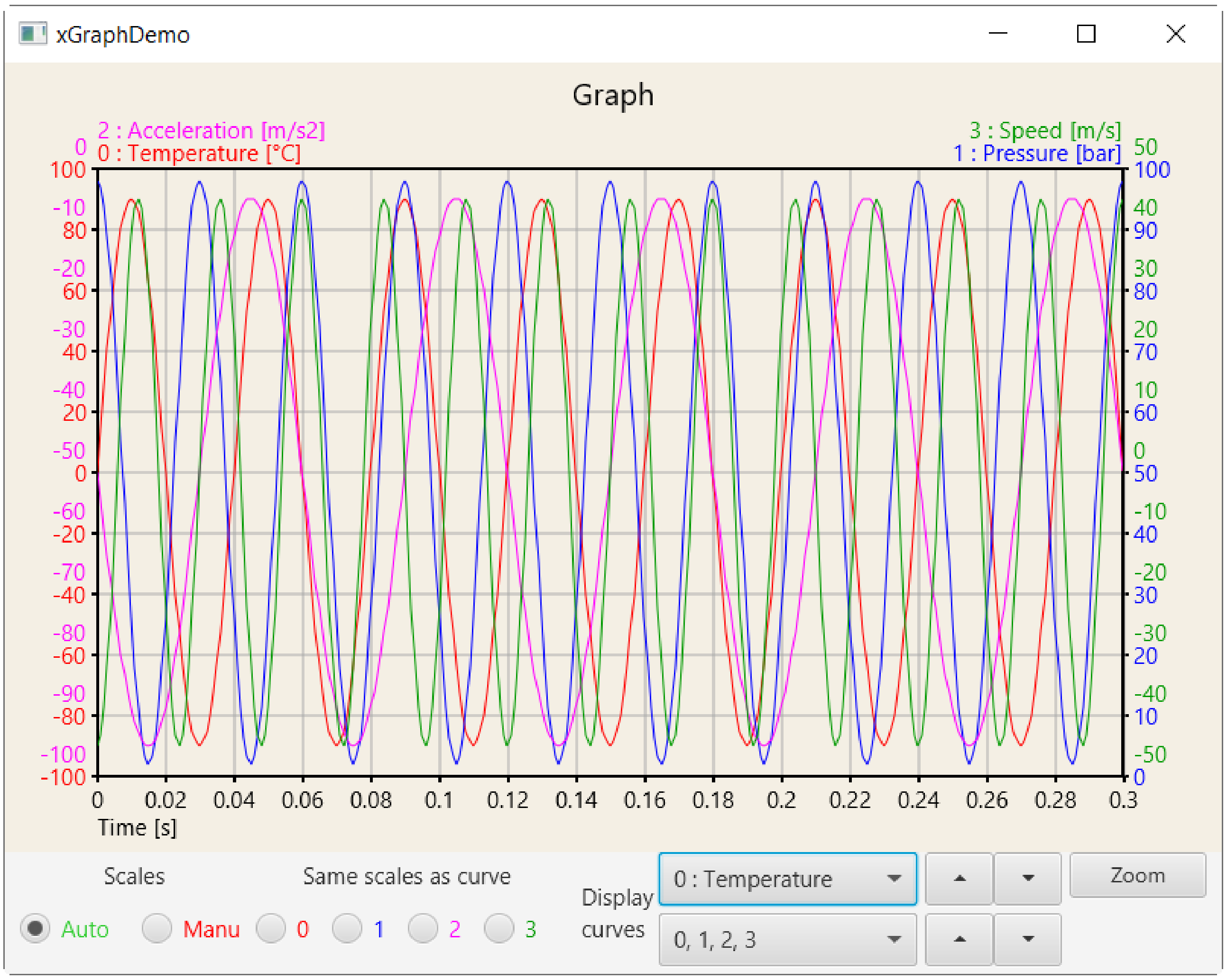
4 xGraph XUI CustomView

The image below is a screenshot of the demo program for the xGraph cross platform XUI CustomView class, which is also available as a B4XLibrary.

What can be done:

- Draw up to 4 curves on the same graph.
- Show values when moving the cursor on the graph.
- Zooming, either with the cursor or setting in the code.
- Different scale modes.
- Different methods on curves.
    - Copy a curve to another location.
    - Add a value to a curve.
    - Multiply a curve by a value.
    - Add two curves and save the result in another curve.
    - Multiply two curves and save the result in another curve.
    - Calculate the integral of a curve and save the result in another curve.
    - Calculate the derivative of a curve and save the result in another curve.

This picture below is a screenshot of the B4J demo program.



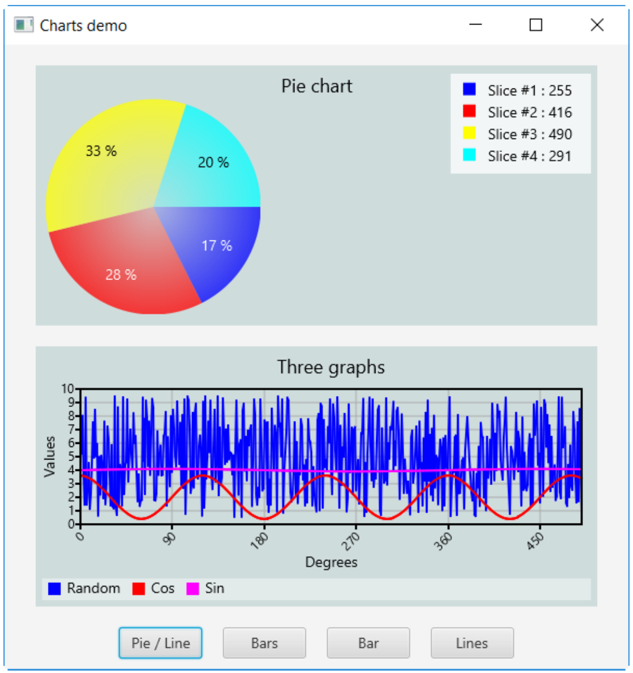
5 xChart XUI CustomView

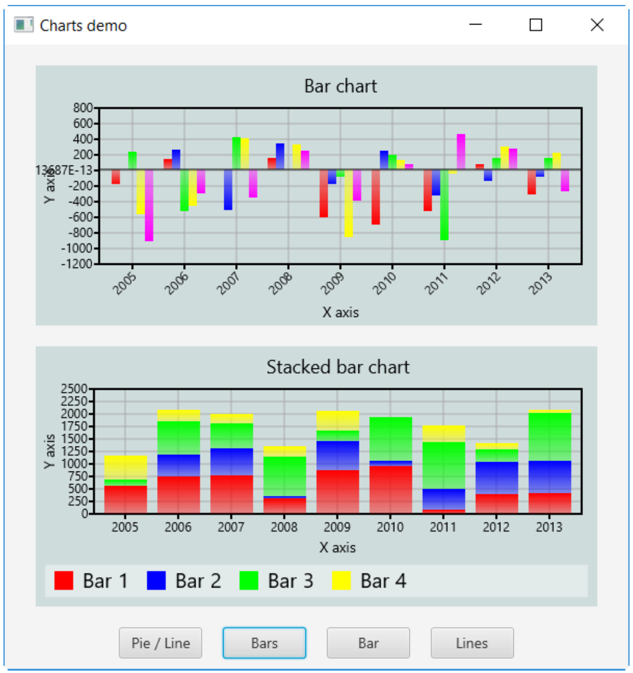
The charts below are drawn with the xChart cross platform XUI CustomView class.
It is also available as a B4XLibrary.

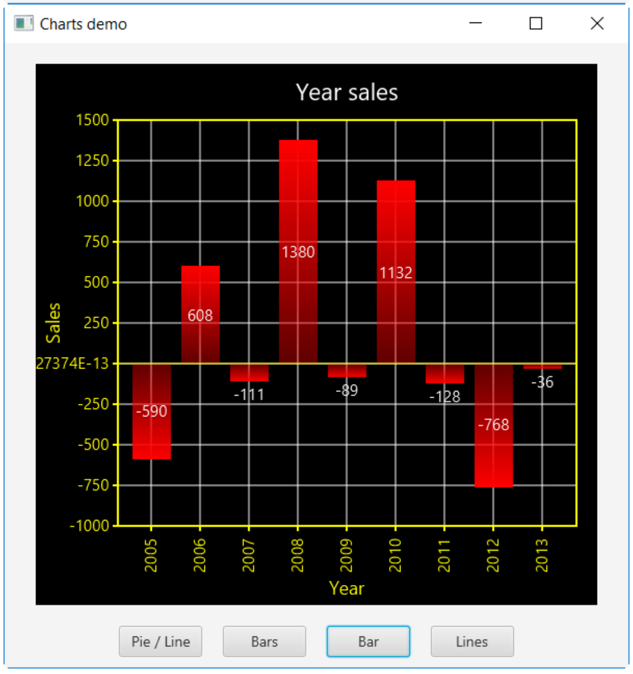
It allows to draw different chart types:

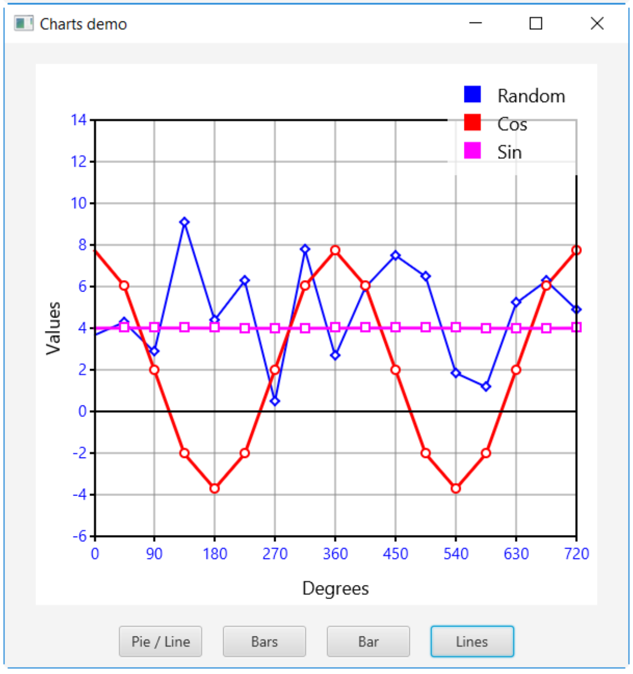
- Pie
- Bar
- Stacked bar
- Line

The screenshots are made with the B4J demo program.





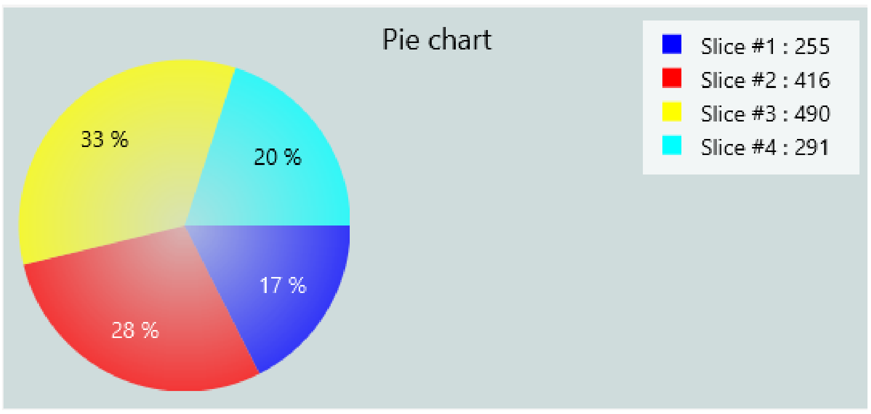




The code is relatively simple.

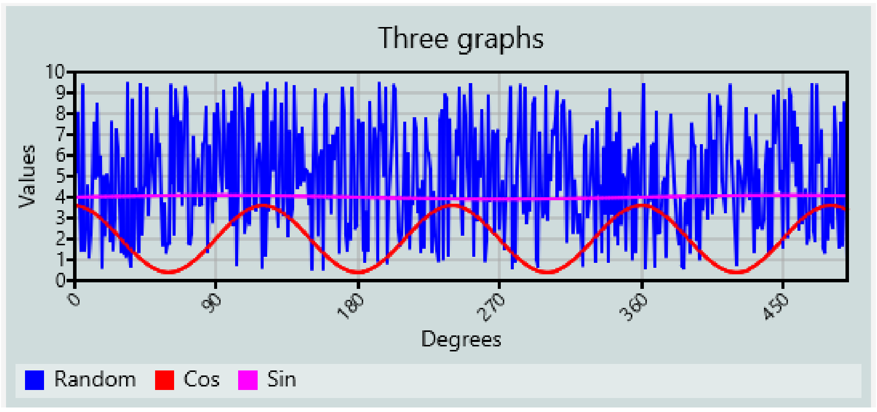
Example code to draw a Pie chart in the first picture:
PieChart1 is an xChart CustomView.

```
Sub CreatePieData
   'Initialize the pie chart data
   PieChart1.ClearData

   Private i, Values(4) As Int

   For i = 0 To 3
       Values(i) = Rnd(50, 501)
   Next

   PieChart1.AddPie("Slice #1", Values(0), xui.Color_Blue) '0 = random color
   PieChart1.AddPie("Slice #2", Values(1), xui.Color_Red)
   PieChart1.AddPie("Slice #3", Values(2), xui.Color_Yellow)
   PieChart1.AddPie("Slice #4", Values(3), xui.Color_Cyan)
   PieChart1.GradientColorsAlpha = 48

   PieChart1.DrawChart
End Sub
```



Example code to draw the Line chart in the first picture:
LineChart1 is an xChart CustomView.

```
Private Sub CreateLineChart1Data
  ' Initialize the line data
  LineChart1.ClearData

  LineChart1.Title = "Three graphs"
  LineChart1.XAxisName = "Degrees"
  LineChart1.YAxisName = "Values"
  LineChart1.YScaleMaxValue = 1
  LineChart1.YScaleMinValue = -1
  LineChart1.IncludeLegend = "BOTTOM"
  LineChart1.AutomaticScale = True
  LineChart1.XScaleTextOrientation = "45 DEGREES"
  LineChart1.AddLine("Random", xui.Color_Blue) '0 = random color
  LineChart1.AddLine("Cos", xui.Color_Red)
  LineChart1.AddLine("Sin", xui.Color_Magenta)

  ' Add the line points.
  Dim Ampl1, Ampl2, Ampl3 As Double
  Ampl1 = Rnd(1, 10001) / 500
  Ampl2 = Rnd(1, 10001) / 500
  Ampl3 = Rnd(1, 10001) / 5002
  For i = 0 To 490
    ' In the case of 2 lines or more we are adding an array of values.
    ' One for each line.
    ' Make sure to create an array for each point.
    ' You cannot reuse a single array for all points.
    LineChart1.AddLineMultiplePoints(i, Array As Double(Rnd(-100, 101) / 300 * Ampl1 + 
5, Ampl2 * CosD(3 * i) + 2, Ampl3 * SinD(i) + 4), i Mod 90 = 0)
  Next

  LineChart1.DrawChart
End Sub
```



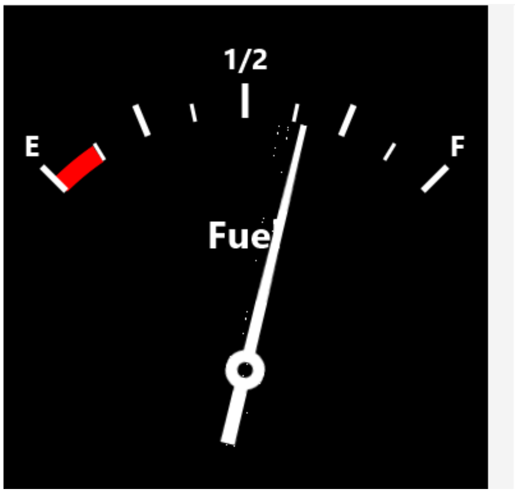
6 xGauges XUI CustomView

The gauges below are drawn with the xGauges cross platform XUI CustomView class.
It is also available as a B4XLibrary.

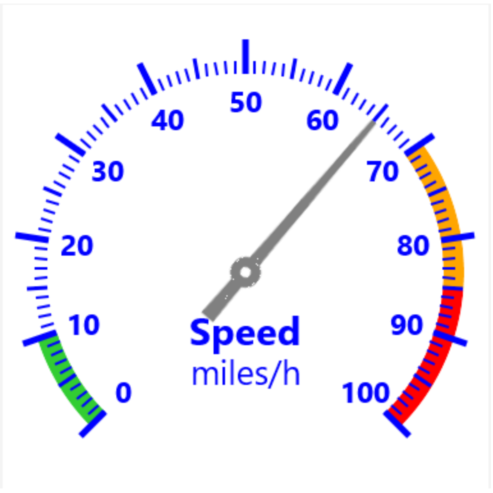
It allows to draw different chart types:

- 90° Top
- 180°
- 270°
- 90° Left
- Custom scale angles

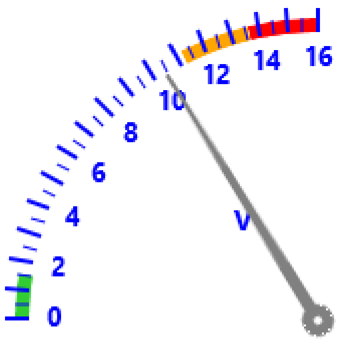
The screenshots are made with the B4J demo program.



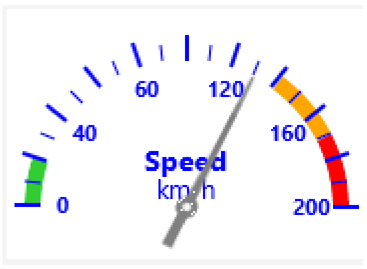
90° Top



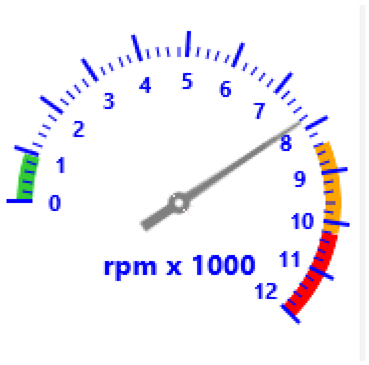
270°



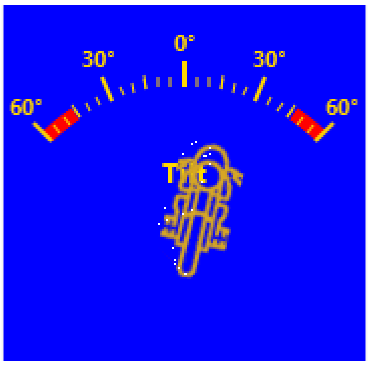
90° Left



180°

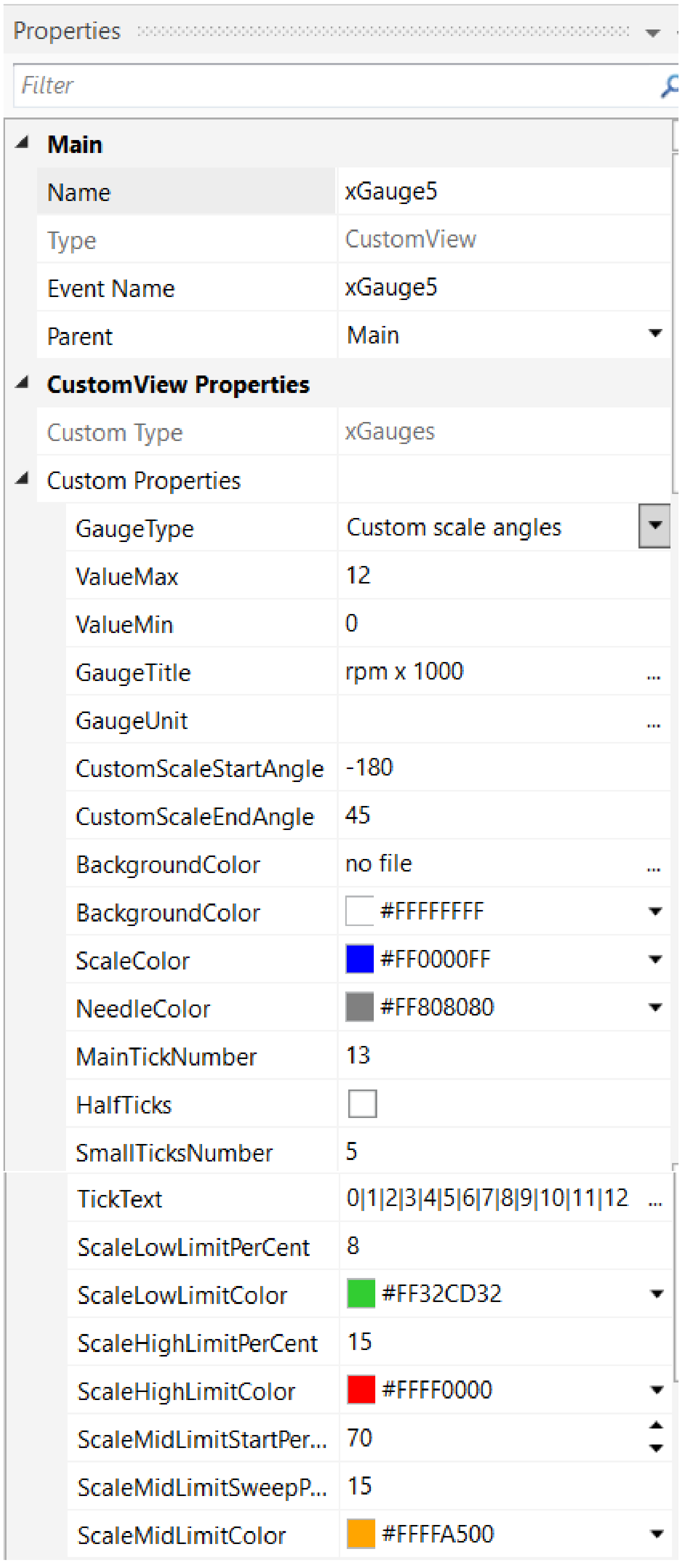


Custom scale angles



90° Top with a bitmap needle

All properties can be set in the Designer like, properties of the Custom scale angles example:



Properties

Filter

Main	
Name	xGauge5
Type	CustomView
Event Name	xGauge5
Parent	Main
CustomView Properties	
Custom Type	xGauges
Custom Properties	
GaugeType	Custom scale angles
ValueMax	12
ValueMin	0
GaugeTitle	rpm x 1000
GaugeUnit	
CustomScaleStartAngle	-180
CustomScaleEndAngle	45
BackgroundColor	no file
BackgroundColor	⬜ #FFFFFFFF
ScaleColor	🟦 #FF0000FF
NeedleColor	⬛ #FF808080
MainTickNumber	13
HalfTicks	☐
SmallTicksNumber	5
TickText	0|1|2|3|4|5|6|7|8|9|10|11|12
ScaleLowLimitPerCent	8
ScaleLowLimitColor	🟩 #FF32CD32
ScaleHighLimitPerCent	15
ScaleHighLimitColor	🟥 #FFFF0000
ScaleMidLimitStartPer...	70
ScaleMidLimitSweepP...	15
ScaleMidLimitColor	🟧 #FFFFA500

7 B4A view background drawing

7.1 View Drawables

The views have default backgrounds when they are defined either in the Designer or by code.
There exist four background objects.

- ColorDrawables
- GradientDrawables
- BitmapDrawables
- StateListDrawable

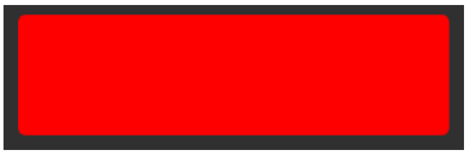
They can be assigned in the Designer but will need extra code to make it work properly.

The source code is in the B4ABackGrounds folder.

7.1.1 ColorDrawable

The ColorDrawable object has a solid single color and the corners can be rounded or square.
The code below sets a ColorDrawable background to a panel.

```
Private pnlColor As Panel

pnlColor.Initialize("")
Activity.AddView(pnlColor, 10%x, 20dip, 80%x, 80dip)
Private cdwColor As ColorDrawable
cdwColor.Initialize(Colors.Red, 5dip)
pnlColor.Background = cdwColor
```



```
cdwColor.Initialize(Colors.Red, 5dip)
```

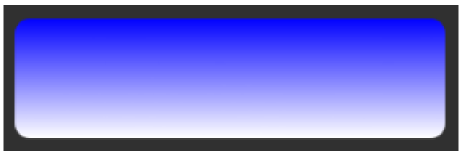
The Initialize method of the ColorDrawable object needs two properties :

• Color	`Colors.Red`
• CornerRadius	`5dip`

7.1.2 GradientDrawable

The GradientDrawable object has two colors with a gradient change from the first to the second color.
The code below sets a GradientDrawable background to a panel.

```
Private pnlGradient As Panel

pnlGradient.Initialize("")
Activity.AddView(pnlGradient, 10%x, 120dip, 80%x, 80dip)
Private gdwGradient As GradientDrawable
Private Cols(2) As Int
Cols(0) = Colors.Blue
Cols(1) = Colors.White
gdwGradient.Initialize("TOP_BOTTOM", Cols)
gdwGradient.CornerRadius = 10dip
pnlGradient.Background = gdwGradient
```



```
gdwGradient.Initialize("TOP_BOTTOM", Cols)
```

The GradientDrawable Initialize method needs two parameters :

- a string with the orientation `"TOP_BOTTOM"`
- a color array with two colors `Cols`

The possible orientations are:

- TOP_BOTTOM
- TR_BL (Top - Right to Bottom - Left)
- RIGHT_LEFT
- BR_TL (Bottom - Right to Top - Left
- BOTTOM_TOP
- BL_TR (Bottom - Left to Top - Left
- LEFT_RIGHT
- TL_BR (Top - Left to Bottom - Right)

The CornerRadius is another separate property.

```
gdwGradient.CornerRadius = 10dip
```

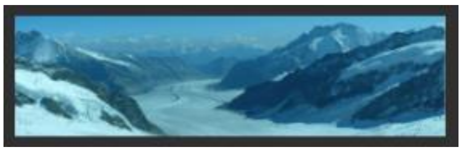
7.1.3 BitmapDrawable

The BitmapDrawable object has two properties a Bitmap and a Gravity property.

The BitmapDrawable object has no rounded corner property, if you want rounded corners these must be part of the bitmap.

The code below sets a BitmapDrawable background to a panel.

```
Private pnlBitmap As Panel

pnlBitmap.Initialize("")
Activity.AddView(pnlBitmap, 10%x, 220dip, 80%x, 80dip)
Private bdwBitmap As BitmapDrawable
bdwBitmap.Initialize(LoadBitmap(File.DirAssets, "background.png"))
bdwBitmap.Gravity = Gravity.FILL
pnlBitmap.Background = bdwBitmap
```


Aletsch glacier, picture taken from the Jungfraujoch.

```
bdwBitmap.Initialize(LoadBitmap(File.DirAssets, "background.png")
```
Sets the bitmap property, in this case a file loaded from the File.DirAssets folder.
But it could be any bitmap.

It is also possible to draw onto the panel's background with a Canvas which has this Panel as the target. The Canvas.Bitmap property points to the `bdwBitmap.Bitmap` property.

```
bdwBitmap.Gravity = Gravity.FILL
```
Sets the Gravity property.

The Gravity property values can be:
- BOTTOM
- CENTER
- CENTER_HORIZONTAL
- CENTER_VERTICAL
- FILL
- LEFT
- NO_GRAVITY
- BOTTOM
- TOP

The Gravity property can be a combination of above values.
Examples :
```
bdwBitmap.Gravity = Gravity.TOP + Gravity.LEFT
bdwBitmap.Gravity = Gravity.BOTTOM + Gravity.CENTER_HORIZONTAL
```

In the Designer there are only three values available: Fill, Center and Top-Left.

7.1.4 StateListDrawable

The StateListDrawable is a drawable that holds other drawables and chooses the current one based on the view's state.

The Background property of Buttons is a StatelistDrawable, it can be defined either in the Designer or in the code.

In the Designer there are two options:

• DefaultDrawable	default colors	set by default
• StatelistDrawable	custom colors	

The button StatelistDrawable has three states.

- Enabled Drawable
- Disabled Drawable
- Pressed Drawable

Each state has its own Drawable, that could be one of the three ColorDrawable, GradientDrawable or BitmapDrawable.

7.1.4.1 Button with ColorDrawables

Example code for a Button with a ColorDrawable :

The source code is the in the ButtonStateDrawables folder.

```
Private btnColor As Button

btnColor.Initialize("btnColor")
Activity.AddView(btnColor, 10%x, 320dip, 25%x, 60dip)
btnColor.Text = "Color"

' Define a color for Enabled state
Private cdwGreenColorEnabled As ColorDrawable
cdwGreenColorEnabled.Initialize(Colors.Green,10)

' Define a color for Pessed state
Private cdwGreenColorPressed As ColorDrawable
cdwGreenColorPressed.Initialize(Colors.RGB(255,182,18),10)

' Define a StateListDrawable
Private stdColor As StateListDrawable
stdColor.Initialize
stdColor.AddState(stdColor.State_Pressed, cdwGreenColorPressed)
stdColor.AddState(stdColor.State_Enabled, cdwGreenColorEnabled)

' Set StateListDrawable to button background
btnColor.Background = stdColor
```

7.1.4.2 Button with GradientDrawables

Example code for a Button with a GradientDrawable :

```
Private btnGradient As Button

btnGradient.Initialize("btnGradient")
Activity.AddView(btnGradient, 40%x, 320dip, 25%x, 60dip)
btnGradient.Text = "Gradient"

' Define two gradient colors for Enabled state
Private colsEnabled(2) As Int
colsEnabled(0) = Colors.RGB(255,196,196)
colsEnabled(1) = Colors.RGB(255,10,10)

' Define a GradientDrawable for Enabled state
Private gdwEnabled As GradientDrawable
gdwEnabled.Initialize("TOP_BOTTOM",colsEnabled)
gdwEnabled.CornerRadius = 5dip

' Define two gradient colors for Pressed state
Private colsPressed(2) As Int
colsPressed(0) = Colors.RGB(10,255,10)
colsPressed(1) = Colors.RGB(255,255,255)

' Define a GradientDrawable for Pressed state
Private gdwPressed As GradientDrawable
gdwPressed.Initialize("TOP_BOTTOM",colsPressed)
gdwPressed.CornerRadius = 5dip

' Define a StateListDrawable
Private stdGradient As StateListDrawable
stdGradient.Initialize
stdGradient.AddState(stdGradient.State_Pressed, gdwPressed)
stdGradient.AddState(stdGradient.State_Enabled, gdwEnabled)

' Set stdRedGradient to button background
btnGradient.Background = stdGradient
```

7.1.4.3 Button with BitmapDrawables

Example code for a Button with BitmapDrawables :

```
Private btnBitmap As Button

btnBitmap.Initialize("btnBitmap")
Activity.AddView(btnBitmap, 70%x, 320dip, 60dip, 60dip)

' Define a bitmap for Enabled state
Private bdwEnabled As BitmapDrawable
bdwEnabled.Initialize(LoadBitmap(File.DirAssets, "btnArrowDown0.png"))

' Define a bitmap for Pressed state
Private bdwPressed As BitmapDrawable
bdwPressed.Initialize(LoadBitmap(File.DirAssets, "btnArrowDown1.png"))

' Define a StateListDrawable
Private stdBitmap As StateListDrawable
stdBitmap.Initialize
stdBitmap.AddState(stdBitmap.State_Pressed, bdwPressed)
stdBitmap.AddState(stdBitmap.State_Enabled, bdwEnabled)

' Set stdBitmap to button btnBitmap
btnBitmap.Background = stdBitmap
```

7.1.4.4 ToggleButton with BitmapDrawable

Example code for a ToggleButton with BitmapDrawables :

The ToggleButoon states are:

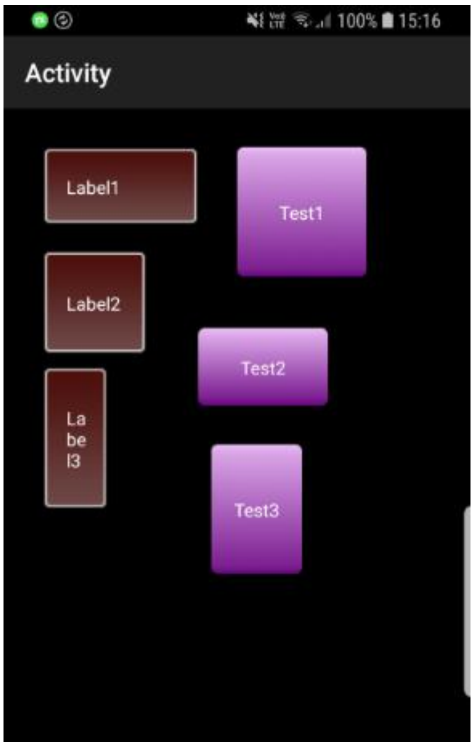
- UnChecked
- Checked

```
Private btnToggleBitmap As ToggleButton

btnToggleBitmap.Initialize("btnBitmap")
btnToggleBitmap.TextOff = ""
btnToggleBitmap.TextOn = ""

Activity.AddView(btnToggleBitmap, 10%x, 400dip, 60dip, 60dip)

' Define a bitmap for Enabled state
Private bdwUnChecked As BitmapDrawable
bdwUnChecked.Initialize(LoadBitmap(File.DirAssets, "btnArrowDown0.png"))

' Define a bitmap for Pressed state
Private bdwChecked As BitmapDrawable
bdwChecked.Initialize(LoadBitmap(File.DirAssets, "btnArrowDown1.png"))

' Define a StateListDrawable
Private stdBitmap As StateListDrawable
stdBitmap.Initialize
stdBitmap.AddState(stdBitmap.State_UnChecked, bdwUnChecked)
stdBitmap.AddState(stdBitmap.State_Checked, bdwChecked)

' Set stdBitmap to button btnBitmap
btnToggleBitmap.Background = stdBitmap
```

7.1.5 NinePatchDrawable

This is a copy of Erel's tutorial in the forum.

The example code is NinePatchExample in the SourceCode folder.



Android supports a special format of PNG images that can be resized by replicating specific parts of the image.
These images also include padding information.
These images are named nine-patch images.

You can read more about this format here: Canvas and Drawables | Android Developers

In the example three labels use the same background nine-patch image and three button using another nine-patch image.

Android SDK includes a tool named draw9patch.bat that can help you with building and modifying such images. This tool is available under: <android>\Tools
You can read more about it here :
Draw 9-patch | Android Developers

The following steps are required to use a nine patch image as a view background:
- Copy the image to <project folder>\Objects\res\drawable
- Set the image to be read-only (otherwise it will be deleted during compilation).
- Add the following sub to your code (requires Reflection library):

```
Sub SetNinePatchDrawable(Control As View, ImageName As String)
   Dim r As Reflector
   Dim package As String
   Dim id As Int
   package = r.GetStaticField("anywheresoftware.b4a.BA", "packageName")
   id = r.GetStaticField(package & ".R$drawable", ImageName)
   r.Target = r.GetContext
   r.Target = r.RunMethod("getResources")
   Control.Background = r.RunMethod2("getDrawable", id, "java.lang.int")
End Sub
```

For buttons you should use this sub which creates a StateListDrawable from two nine-patch images:

```
Sub SetNinePatchButton(Btn As Button, DefaultImage As String, PressedImage As String)
   Dim r As Reflector
   Dim package As String
   Dim idDefault, idPressed As Int
   package = r.GetStaticField("anywheresoftware.b4a.BA", "packageName")
   idDefault = r.GetStaticField(package & ".R$drawable", DefaultImage)
   idPressed = r.GetStaticField(package & ".R$drawable", PressedImage)
   r.Target = r.GetContext
   r.Target = r.RunMethod("getResources")
   Dim sd As StateListDrawable
   sd.Initialize
   sd.AddState(sd.State_Pressed, r.RunMethod2("getDrawable", idPressed, "java.lang.int"))
   sd.AddCatchAllState( r.RunMethod2("getDrawable", idDefault, "java.lang.int"))
   Btn.Background = sd
End Sub
```

Now you should use this sub to set the views backgrounds:

```
Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("1")
   SetNinePatchDrawable(Label1, "label_bg")
   SetNinePatchDrawable(Label2, "label_bg")
   SetNinePatchDrawable(Label3, "label_bg")
End Sub
```

Tips
- Don't modify the image files located under res\drawable directly with the draw9patch tool. It removes the read-only attribute and then the image will be deleted.
- The image name must be lower case (allowed characters a - z, 0 - 9, . , _).
- After adding a new image you should clean the project by choosing Tools - Clean Project. This causes a generated file (R.java) to be recreated and include the new resources.