Layouts

(Information and examples taken from the DroidScript documentation)

Description

Layouts are the base to position controls over the screen of your Android device. Layouts are container objects which are used to visually organize graphical objects (controls), such as text, buttons and images on the screen. There are 3 types of layout:

- “Linear”
- “Frame”
- “Absolute”

Create layouts using the **CreateLayout** function of the app object:

```
lay.AddChild( object );
```

Add child objects to a layout using the **AddChild** function of the layout object.

```
lay = app.CreateLayout( type, options );
```

The alignment of child objects within a layout can be set by adding the options **“Left”**, **“Right”**, **“Bottom”** and **“VCenter”**, by default objects will be aligned **“Top,Center”**.
Remove layouts using the **RemoveLayout** function of the app object:

```
app.RemoveLayout( lay );
```

Linear Layouts

Linear layouts allow you to add controls one after another without overlapping them, in a vertical or horizontal way. Linear layouts are probably the most useful type and are used to organize controls in either the **“Vertical”** or **“Horizontal”** direction on screen.

Example - Vertical

```
function OnStart()
{
  lay = app.CreateLayout( "Linear", "Vertical" );

  btnA = app.CreateButton( "A", 0.2, 0.1 );
  lay.AddChild( btnA );

  btnB = app.CreateButton( "B", 0.2, 0.1 );
  lay.AddChild( btnB );

  btnC = app.CreateButton( "C", 0.2, 0.1 );
  lay.AddChild( btnC );

  app.AddLayout( lay );
}
```

Example - Horizontal

```
function OnStart()
{
  lay = app.CreateLayout( "Linear", "Horizontal,FillXY" );

  btnA = app.CreateButton( "A", 0.2, 0.1 );
  lay.AddChild( btnA );

  btnB = app.CreateButton( "B", 0.2, 0.1 );
  lay.AddChild( btnB );

  btnC = app.CreateButton( "C", 0.2, 0.1 );
  lay.AddChild( btnC );

  app.AddLayout( lay );
}
```

By default Layouts will auto-size to wrap their contents but you have 3 more options as to how a layout sizes within it's parent: **“FillX”**, **“FillY”** and **“FillXY”**.

Example - Combined

```
function OnStart()
{
  layVert = app.CreateLayout( "Linear", "Vertical,FillXY" );

  btnA = app.CreateButton( "A", 0.6, 0.1 );
  layVert.AddChild( btnA );

  layHoriz = app.CreateLayout( "Linear", "Horizontal" );
  layVert.AddChild( layHoriz );

  btnB1 = app.CreateButton( "B1", 0.2, 0.1 );
  layHoriz.AddChild( btnB1 );
  btnB2 = app.CreateButton( "B2", 0.2, 0.1 );
  layHoriz.AddChild( btnB2 );
  btnB3 = app.CreateButton( "B3", 0.2, 0.1 );
```

```
  layHoriz.AddChild( btnB3 );

  btnC = app.CreateButton( "C", 0.6, 0.1 );
  layVert.AddChild( btnC );

  app.AddLayout( layVert );
}
```

Frame Layouts

Frame layouts are used to display objects in front or behind each other. Every time the **AddChild** function is called on a Frame layout, the child object is placed in a new layer **in front** of the previously added object at the top left of the frame. Frame Layouts are useful if you wish to do **animated Flips** or **Slides** to reveal layers of objects or use **transparency.**

Example - Image Swap

```
function OnStart()
{
  lay = app.CreateLayout( "Linear", "Vertical,FillXY" );


  layFrm = app.CreateLayout( "Frame" );
  img1 = app.CreateImage( "/Sys/Img/Droid1.png" );
  layFrm.AddChild( img1 );

  img2 = app.CreateImage( "/Sys/Img/Droid2.png" );
  img2.SetVisibility( "Hide" );
  layFrm.AddChild( img2 );
  lay.AddChild( layFrm );

  btn = app.CreateButton( "Press Me" );
  btn.SetMargins( 0,0.1,0,0 );
  btn.SetOnTouch( btn_OnTouch );
  lay.AddChild( btn );

  app.AddLayout( lay );
}

function btn_OnTouch()
{
  if( img2.GetVisibility()=="Hide" )
   img2.SetVisibility( "Show" );
  else
   img2.SetVisibility( "Hide" );
}
```

Absolute Layouts

Absolute layouts allows you to add controls in any position relative to the width and height of your screen from 0 to 1. Absolute layouts ignore all alignment options and allow the absolute positioning of controls by calling the **SetPosition** and **SetSize** functions of each of the child objects. This type of layout is rarely used and you are encouraged use Linear layouts for most of your programs.

Padding and Margins

In Linear and Frame Layouts, you can use the **SetPadding** function of a layout to keep a layout's child objects away from the edges of the layout:

```
lay.SetPadding( left, top, right, bottom );
```

Also every child object within a layout can have margins added by using the SetMargins function of the child object:

```
obj.SetMargins( left, top, right, bottom );
```

Using Linear layouts and setting margins on child objects is usually the best way to position your App's graphical objects.

Backgrounds

The background of a layout will be transparent by default, but you can set a color using the **SetBackColor** function.

```
lay.SetBackColor( colorCode );
```

Colors are given as hex **color codes** which can be copied from various graphics programs or simply exprimented with until you get the color you want. The format is (#alpha:red:green:blue) where each value can range from 0 to 255 in base 16 which is 00 to ff.
For example “#ff00ff00” would be full strength green and “#ff000088” would be around half strength blue and “#44ff00ff” would be semi-transparent full strength purple.
You can also use a gradient background for a layout using the **SetBackGradient** and **SetBackGradientRadial** functions.

```
lay.SetBackGradient( colorCode1, colorCode2 );
```

```
lay.SetBackGradientRadial( x, y, radius, colorCode1, colorCode2 );
```

Visibility

The visibility of both layouts and child objects can be controlled using the **SetVisibility** function. Use the values **“Show”** or **“Hide”** to make objects invisible or visible and **“Gone”** to exclude the object from the layout completely (surrounding objects will re-arrange).

```
obj.SetVisibility( mode );
```

Methods

Method	Description
Layout.AddChild(child,order)	

Layout.Animate(type,callback)	known types:
	
	`SlideFromLeft`
	`ScaleFromLeft`
	`SlideToLeft`
	`ScaleToLeft`
	`SlideFromRight`
	`ScaleFromRight`
	`SlideToRight`
	`ScaleToRight`
	`SlideFromTop`
	`ScaleFromTop`
	`SlideToTop`
	`ScaleToTop`
	`SlideFromBottom`
	`ScaleFromBottom`
	`SlideToBottom`
	`ScaleToBottom`

Layout.ChildToFront(child)	
Layout.Destroy()	
Layout.DestroyChild(child)	
Layout.GetAbsHeight()	
Layout.GetAbsWidth()	
Layout.GetChildOrder(child)	
Layout.GetHeight()	
Layout.GetPosition()	
Layout.GetType()	
Layout.GetVisibility()	
Layout.GetWidth()	
Layout.Release()	
Layout.RemoveChild(child)	
Layout.SetBackColor(colorCode)	
Layout.SetBackGradient(color1,color2,color3,p4,p5,p6,p7)	
Layout.SetBackGradientRadial(x,y,r,color1,color2,color3,p7)	
Layout.SetBackground(imageFile, options)	options is an optional string that can be “repeat”
Layout.SetMargins(left,top,right,bottom)	
Layout.SetOrientation(orient)	“Portrait” or “Landscape”
Layout.SetPadding(left, top, right, bottom)	
Layout.SetPosition(left, top, width, height)	
Layout.SetScale(x,y)	
Layout.SetSize(width, height)	
Layout.SetTouchable(touchable)	

Method	Description
Layout.SetVisibility(visibility)	“Hide”, “Show” or “Gone”

Options

You can combine options too. Use the comma to seperate the options:

```
app.CreateLayout("Linear", "Horizontal,FillXY,TouchThrough");
```

Vertical	
Horizontal	
VCenter	
TopCenter	Default
FillX	
FillY	
FillXY	
Right	
Left	
Bottom	
TouchThrough	