

This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. The article is believed to be still valid for the original topic scope.

Contents

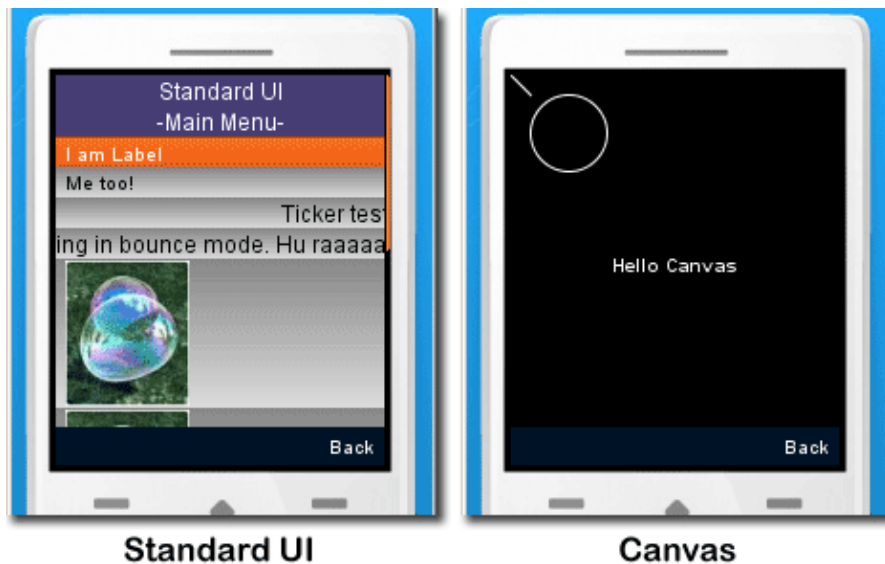
- [1 Introduction](#)
- [2 Standard UI vs Canvas](#)
- [3 Usage Example](#)
 - ◆ [3.1 widget.xml](#)
 - ◆ [3.2 .he Source File](#)
- [4 Drawing Operation](#)
 - ◆ [4.1 setColor](#)
 - ◆ [4.2 getColor](#)
 - ◆ [4.3 setFont](#)
 - ◆ [4.4 getFont](#)
 - ◆ [4.5 drawLine](#)
 - ◆ [4.6 drawRect](#)
 - ◆ [4.7 drawArc](#)
 - ◆ [4.8 drawString](#)
 - ◆ [4.9 drawImage](#)
 - ◆ [4.10 fillRect](#)
 - ◆ [4.11 fillTriangle](#)
 - ◆ [4.12 fillArc](#)
 - ◆ [4.13 setClip](#)
 - ◆ [4.14 getClip](#)
 - ◆ [4.15 translate](#)
 - ◆ [4.16 getTranslate](#)
- [5 Code Snippet](#)
- [6 See Also](#)

Introduction

For UI implementation in WidSets widget, not only Standard UI can be used but also freely drawable Canvas. This article will show you how to use it.

Standard UI vs Canvas

Standard UI is instant UI and ready to use. In this type of UI, you can't change its style and layout much. While Canvas is freely drawable area to make your own UI from simple UI to a fantastic game UI.



Usage Example

Let's learn by example.

widget.xml

First, add style sheet to widget.xml

```
<stylesheet>
...
  canvas {
    color-1: black;
    color-2: red;
    color-3: white;
    font-3: small bold underlined;
  }
</stylesheet>
```

color-1, **color-2**, **color-3** are predefined colors and can be used in your widget. Maximum amount of predefined colors is 4 (color-1 to color-4). In the same way, **font-3** is predefined font. Maximum amount is 4

too.

.he Source File

Add Canvas construction in **openWidget** function.

```
Shell openWidget ()
{
    final Canvas canvas = new Canvas(getStyle("canvas"));
    final Shell shell = new Shell(canvas);

    //place canvas over automatic Scrollable created by Shell
    shell[0] = shell[0][0];

    return shell;
}
```

And then add **paint** function like this.

```
void paint(Component c, Graphics g, Style style, int width, int height)
{
    g.setColor(style.color(0));
    g.fillRect(0, 0, width, height);
    g.setColor(0xFF8888);
    g.setFont(style.font(2));
    g.drawLine(0, 0, 10, 10);
    g.fillArc(10, 10, 40, 40, 0, 360);
    g.drawString(getParameter("widgetname"), width/2, height/2, TOP|HCENTER);
    g.drawImage(getImage("mnicon.png"), 0, height, BOTTOM|LEFT);
}
```

paint is callback function that will be called by Canvas object. 2nd argument, **Graphic g**, is used for draw thing on Canvas.

This line will set current color to predefined color number 0 (color-1 in style sheet). In this example, color-1 is black.

```
g.setColor(style.color(0));
```

Fill the entire screen with current color (black).

```
g.fillRect(0, 0, width, height);
```

Change current color to Pink (0xFF8888 in the form of 0xRRGGBB)

```
g.setColor(0xFF8888);
```

Set font to predefined font number 2 (font-3)

```
g.setFont(style.font(2));
```

Draw line from (0, 0) to (10, 10)

```
g.drawLine(0, 0, 10, 10);
```

WidSets_for_Rookie_EP_8_: Canvas

Fill arc in rect (10, 10, 10+40, 10+40) from degree 0 to 360. In the other word, it will fill the circle.

```
g.fillArc(10, 10, 40, 40, 0, 360);
```

Draw predefined string **widgetname** in **widget.xml** at the center of screen.

```
g.drawString(getParameter("widgetname"), width/2, height/2, TOP|HCENTER);
```

Draw embedded Image at bottom left position.

```
g.drawImage(getImage("mnicon.png"), 0, height, BOTTOM|LEFT);
```

Here is the result.



Drawing Operation

setColor

Change current color to specific color. There are 2 ways to define color, use predefined color in widget.xml and use instant value.

Predefined color

```
g.setColor(style.color(0)); // use value from color-1
```

Instant value

```
g.setColor(0xFF8888);
```

getColor

```
int color = g.getColor();
```

Get current color in form 0xRRGGBB.

setFont

Change current font to specific font defined in style sheet

```
g.setColor(style.font(0)); // use value from font-1
```

getFont

```
Font font = g.getFont();
```

Get current font in Font object.

drawLine

```
g.drawLine(int x1, int y1, int x2, int y2);
```

Draws a line between the coordinates (x1,y1) and (x2,y2) using the current color and stroke style.

drawRect

```
g.drawRect(int x, int y, int width, int height);
```

Draws the outline of the specified rectangle using the current color and stroke style. The resulting rectangle will cover an area (width + 1) pixels wide by (height + 1) pixels tall. If either width or height is less than zero, nothing is drawn.

drawArc

```
g.drawArc(int x, int y, int width, int height, int startAngle, int arcAngle);
```

Draws a circular or elliptical arc covering the specified rectangle.

To draw circle or ellipse, arcAngle has to be set to 360.

drawString

```
g.drawString(String str, int x, int y, int anchor);
```

Draws the specified String using the current font and color. The x,y position is the position of the anchor point.

anchor can be these values.

- **Horizontal Position**
 - ◆ LEFT
 - ◆ HCENTER
 - ◆ RIGHT
- **Vertical Position**
 - ◆ TOP
 - ◆ VCENTER
 - ◆ BOTTOM

drawImage

```
g.drawImage(Image img, int x, int y, int anchor);
```

Draws the specified image by using the anchor point. The image can be drawn in different positions relative to the anchor point by passing the appropriate position constants.

The easiest way to create Image object in first argument is to use **getImage(...)** function as you see in example above.

fillRect

```
g.fillRect(int x, int y, int width, int height);
```

Fills the specified rectangle with the current color. If either width or height is zero or less, nothing is drawn.

fillTriangle

```
g.fillTriangle(int x1, int y1, int x2, int y2, int x3, int y3);
```

Fills the specified triangle with the current color. The lines connecting each pair of points are included in the filled triangle.

fillArc

```
g.drawArc(int x, int y, int width, int height, int startAngle, int arcAngle);
```

Fills a circular or elliptical arc covering the specified rectangle.

setClip

```
g.setClip(int x, int y, int width, int height, boolean absolute);
```

Sets the current clip to the rectangle specified by the given coordinates. If the absolute is false the current clip is intersected with the specified. **Rendering operations have no effect outside of the clipping area.**

getClip

```
int clipX, int clipY, int clipWidth, int clipHeight = g.getClip();
```

Returns the current clipping area.

translate

```
g.translate(int x, int y)
```

Translates the origin of the graphics context to the point (x, y) in the current coordinate system. **All coordinates used in subsequent rendering operations on this graphics context will be relative to this new origin.** The effect of translate() calls are cumulative. The application can set an absolute origin (ax, ay) using the following technique:

```
g.translate(ax - g.getTranslateX(), ay - g.getTranslateY());
```

getTranslate

```
int translateX, int translateY = g.getTranslate();
```

Returns the current translation origin.

Code Snippet

You can download source code for this tutorial from [File:WidSets Canvas Example.zip](#)

See Also

- [WidSets for Rookie EP 1 : First Step to WidSets SDK](#)
- [WidSets for Rookie EP 2 : First Compilation with WidSets SDK](#)
- [WidSets for Rookie EP 3 : Understand Hello World](#)
- [WidSets for Rookie EP 4 : Fasten WidSets Development](#)
- [WidSets for Rookie EP 5 : EditPlus Integration](#)
- [WidSets for Rookie EP 6 : Softkey Menu](#)
- [WidSets for Rookie EP 7 : Standard UI](#)
- **WidSets for Rookie EP 8 : Canvas**

WidSets_for_Rookie_EP_8:_Canvas

- [WidSets for Rookie EP 9 : Timer](#)
- [WidSets for Rookie EP 10 : Key Handling](#)
- [WidSets for Intermediate EP 1 : HTTP Request](#)
- [WidSets for Intermediate EP 2 : HTTP with XML Filter](#)
- [WidSets for Advance EP 1 : Life Pictures Project](#)
- [WidSets SDK Tips : Emulator Language Changing](#)
- [WidSets SDK Tips : Emulator Skin Changing](#)
- [WidSets SDK Tips : Add Custom Emulator Skin](#)