How to implement touch image scrolling in Java ME

Introduction

As of S60 5th Edition and Series 40 6th Edition, Feature Pack 1, a new generation of **touch-based applications** can be implemented on Nokia devices. This article explains how to implement a **touch-based scroll functionality** for a single Image in Java ME.

Touch Devices and Backwards Compatibility

Though the main scope of this article is the Symbian Touch devices as well as the Series 40 Touch and Type devices, it is possible to keep the application backwards compatible, by implementing the keyRepeated() method of a class that extends the Canvas object. In this way the image can be moved on standard keyboard devices by using the four navigation keys UP, DOWN, LEFT and RIGHT.

Variables and Constructor

Use of low level LCDUI Components is needed for this example, therefore the supporting class extends the Canvas object:

```
public class ScrollableImageCanvas extends Canvas
```

We also need to define the variables that we will use in order to manage the following:

- image size
- image translation
- user touch interaction

```
// image-related properties
int imageHeight = 0;
int imageWidth = 0;
Image image = null;

// scroll properties
protected int translationX = 0;
protected int translationY = 0;

// touch properties
protected int lastPointerX = -1;
protected int lastPointerY = -1;
```

The constructor is quite straightforward, and only needs an Image wrapper for the actual binary file as argument:

```
public ScrollableImageCanvas(Image image)
{
 this.image = image;
 this.imageWidth = image.getWidth();
 this.imageHeight = image.getHeight();
}
```

Painting the image at the given coordinates

This is how an image can be simply drawn at the given pair of x and y coordinates:

```
protected void paint(Graphics g)
```


(C) Copyright Nokia 2013. All rights reserved.

```
{
 g.setColor(0xffffff);
 g.fillRect(0, 0, getWidth(), getHeight());

 g.drawImage(image, - translationX, - translationY, Graphics.TOP | Graphics.LEFT);
}
```

Receiving touch event

Both Symbian Touch and Series 40 Touch and Type devices provide the same methods for receiving touch events. In order to implement the touch functionality, we need to override the 3 pointer handlers provided by the Canvas object:

- pointerPressed: called when the screen is tapped
- pointerReleased: called when tapping is over
- pointerDragged: called for dragging events

```
protected void pointerPressed(int x, int y)
{
 lastPointerX = x;
 lastPointerY = y;
}
protected void pointerReleased(int x, int y)
{
 lastPointerX = -1;
 lastPointerY = -1;
}
protected void pointerDragged(int x, int y)
{
 scrollImage(lastPointerX - x, lastPointerY - y);

 lastPointerX = x;
 lastPointerY = y;
}
```

The scrollImage method

The **scrollImage()** method does the following tasks:

- increments the current translation x and y, which is coordinated by the given x and y deltas
- normalizes the new translation for the x and y coordinates, so that the Image does not get out of bounds

```
void scrollImage(int deltaX, int deltaY)
{
 if(imageWidth > getWidth())
 {
  translationX += deltaX;

  if(translationX < 0)
   translationX = 0;
  else if(translationX + getWidth() > imageWidth)
   translationX = imageWidth - getWidth();
 }

 if(imageHeight > getHeight())
 {
  translationY += deltaY;

  if(translationY < 0)
   translationY = 0;
  else if(translationY + getHeight() > imageHeight)
   translationY = imageHeight - getHeight();
 }

 repaint();
}
```

Keeping the application backwards compatible with keyboard devices

By adding the keyRepeated method below,it is possible to call the scrollImage method when the keyboard navigation buttons are continuously pressed. The value for the image's new placement on each axis is 20 in this example. Decreasing this value causes slower smoother scrolling, while increasing it causes faster and more abrupt scrolling.

```
 protected void keyRepeated(int keyCode)
 {
  if(keyCode==-4)
  {
   scrollImage(20,0);
  }
  if(keyCode==-3)
  {
   scrollImage(-20,0);
  }
  if(keyCode==-2)
  {
   scrollImage(0,20);
  }
  if(keyCode==-1)
  {
   scrollImage(0,-20);
  }
 }
```

The main MIDlet's code

Since most of the functionality is implemented within the custom Canvas, the code for the main MIDlet, only instantiates the canvas by passing the Image

(C) Copyright Nokia 2013. All rights reserved.

as argument and then sets the current display of the MIDlet to the instantiated canvas.

```
try {
            image=Image.createImage("/Image1.jpg");
      } catch (IOException e) {System.out.println("Can't create image object!");}
canvas = new ScrollableImageCanvas(image);
display = Display.getDisplay(this);
display.setCurrent(canvas);
```

Resources

The source code of this MIDlet is available for download from here: File:CanvasScrollableImageSource.zip

The binary files of this MIDlet are available for download from here: File:CanvasScrollableImageBinaries.zip

Further development

A lot of **improvements** can be made to the code presented in this article. Just some examples are:

- **scrollbars**: a fundamental element to let the user know how much he or she can scroll in both the vertical and horizontal axis
- **smooth scrolling**: using some inertia when applying scrolling movement will make the scrolling effect more realistic

See also

Touch UI in Nokia's Online Java Developer's Library

The article has been slightly modified, since its original version, to include support for keyboard devices. The original source code for the ScrollableImageCanvas class is available here: ScrollableImageCanvas.java

In order to get a real feeling of how scrolling is implemented in the original code, you can visit this page: Touch scrollable image in action

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.