Adding a sliding effect to Images in Java ME

Introduction

The following `SlidingImage` class can be used to easily add a sliding transition effect to Java ME Images.

To see how this effect works, you can check here: J2ME Image sliding effect in action.

Source code: SlidingImage class

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;

public class SlidingImage
{
 // constants defining the 2 types of sliding
 public static final int SLIDE_IN = 0;
 public static final int SLIDE_OUT = 1;

 // image type of sliding
 public int slideType = SLIDE_OUT;

 // pieces of the sliding image
 public int pieces = 4;

 // direction of sliding (up, right, down or left)
 public int direction = Canvas.LEFT;

 // image object and properties
 Image image = null;
 int imageWidth = 0;
 int imageHeight = 0;

 // transition time properties
 int duration = 0;
 long startTime = 0;

 // transition state
 public boolean sliding = false;
 public boolean ended = false;

 /**
  * Class constructor
  *
  * @param image image to be slided
  * @param pieces number of pieces of the sliding image
  * @param slideType type of slide, can be SLIDE_IN or SLIDE_OUT
  */
 public SlidingImage(Image image, int pieces, int slideType)
 {
  this.image = image;

  this.imageWidth = image.getWidth();
  this.imageHeight = image.getHeight();

  this.slideType = slideType;
  this.pieces = pieces;
 }

 /**
  * Resets the image to its starting state
  *
  */
 public void reset()
 {
  ended = false;
  sliding = false;
 }

 /**
  * Resets the image to its starting state, specifying new pieces and slideType values
  *
  * @param pieces number of pieces of the sliding image
  * @param slideType type of slide, can be SLIDE_IN or SLIDE_OUT
  */
 public void reset(int pieces, int slideType)
 {
  reset();

  this.pieces = pieces;
  this.slideType = slideType;
```


(C) Copyright Nokia 2013. All rights reserved.

```
  }
  /**
   * Starts the sliding effect
   *
   * @param direction direction of sliding (Canvas.UP, RIGHT, DOWN or LEFT)
   * @param duration duration of sliding in milliseconds
   */
  public void slide(int direction, int duration)
  {
   this.direction = direction;

   this.duration = duration;
   this.startTime = System.currentTimeMillis();

   ended = false;
   sliding = true;
  }
  /**
   * Draws the sliding image by using the anchor point
   *
   * @param g Graphics object to paint the image
   * @param x the x coordinate of the anchor point
   * @param y the y coordinate of the anchor point
   * @param anchor the anchor point for positioning the image
   */
  public void paint(Graphics g, int x, int y, int anchor)
  {
   if(!sliding && ((ended && slideType == SLIDE_IN) || (!ended && slideType == SLIDE_OUT)))
   {
    g.drawImage(image, x, y, anchor);
   }
   else if(sliding)
   {
    // we translate accordingly to coordinates and anchor point
    int translateX = (anchor & Graphics.RIGHT) > 0 ? x - imageWidth :
     ((anchor & Graphics.HCENTER) > 0 ? x - imageWidth / 2 :
      x);
    int translateY = (anchor & Graphics.BOTTOM) > 0 ? y - imageHeight :
     ((anchor & Graphics.VCENTER) > 0 ? y - imageHeight / 2 :
      y);

    g.translate(translateX, translateY);

    // now we get current clipping properties
    int cx, cy, cw, ch;

    cx = g.getClipX();
    cy = g.getClipY();
    cw = g.getClipWidth();
    ch = g.getClipHeight();

    // let's get current duration and check if effect has ended
    int timeDiff = (int)(System.currentTimeMillis() - startTime);

    if(timeDiff > duration)
    {
     timeDiff = duration;

     ended = true;

     sliding = false;
    }
    // if it's a SLIDE_IN effect, we reverse the current time
    if(slideType == SLIDE_IN)
    {
     timeDiff = duration - timeDiff;
    }


    // now, let's pre-calculate some coords/properties
    int pieceEnd = 0;
    int pieceCoord = 0;
    int pieceSize = (direction == Canvas.LEFT || direction == Canvas.RIGHT) ?
     imageHeight / pieces : imageWidth / pieces;

    int singleSlideDuration = (int)duration / pieces;

    switch(direction)
    {
    case Canvas.LEFT:
     pieceEnd = cx - imageWidth;break;
    case Canvas.RIGHT:
     pieceEnd = cx + cw;break;
    case Canvas.UP:
     pieceEnd = cy - imageHeight;break;
    case Canvas.DOWN:
     pieceEnd = cy + ch;break;
    }

    // finally, we can paint single slide pieces
    for(int i = 0; i < pieces; i++)
    {
     // check if this piece has not to be painted (is out of clipping region)
     if(timeDiff > (i + 1) * singleSlideDuration)
     {
      continue;
     }
     else
     {
      pieceCoord = (timeDiff < i * singleSlideDuration) ? 0 :
       pieceEnd * (timeDiff - i * singleSlideDuration) / singleSlideDuration;

      if(direction == Canvas.LEFT || direction == Canvas.RIGHT)
      {
       g.setClip(pieceCoord, i * pieceSize, imageWidth, pieceSize);
       g.drawImage(image, pieceCoord, 0, Graphics.TOP | Graphics.LEFT);
      }
      else
      {
       g.setClip(i * pieceSize, pieceCoord, pieceSize, imageHeight);
       g.drawImage(image, 0, pieceCoord, Graphics.TOP | Graphics.LEFT);
      }
     }
    }
    //restore clipping properties and translate back
    g.setClip(cx, cy, cw, ch);

    g.translate(- translateX, - translateY);
   }
  }
```


(C) Copyright Nokia 2013. All rights reserved.

```
}
```

Source code: sample usage

Here is a simple Canvas using a SlidingImage to create a "slide out" effect.
The steps to use a SlidingImage are these:

- **Instantiate** a SlidingImage
- **Start** the sliding effect via its slide() method
- **Paint** it via its paint() method
- You can always check if transition has ended via the **ended** instance variable
- To reset the effect, you can use the reset() methods, one of which allows to redefine the *pieces* and *slideType* properties

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;

public class SimpleSlidingCanvas extends Canvas implements Runnable
{
 SlidingImage image = null;

 public SimpleSlidingCanvas()
 {
  try
  {
   this.image = new SlidingImage(Image.createImage("/image1.png"), 10, SlidingImage.SLIDE_OUT);

   new Thread(this).start();

   this.image.slide(Canvas.RIGHT, 3000);
  }
  catch(Exception e)
  {
   e.printStackTrace();
  }
 }
 public void run()
 {
  while(true)
  {
   repaint();

   try
   {
    synchronized(this)
    {
     wait(50L);
    }
   }
   catch(Exception e)
   {
    e.printStackTrace();
   }
  }
 }

 protected void paint(Graphics g)
 {
  g.setColor(0xffffff);
  g.fillRect(0, 0, getWidth(), getHeight());

  if(image != null)
  {
   this.image.paint(g, getWidth() / 2, getHeight() / 2, Graphics.HCENTER | Graphics.VCENTER);
  }
  else
  {
   g.setColor(0x000000);
   g.drawString("No image available", getWidth() / 2, getHeight() / 2, Graphics.HCENTER | Graphics.BASELINE);
  }
 }

}
```

Download

- SlidingImage class source code
- sample Canvas using SlidingImage

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.