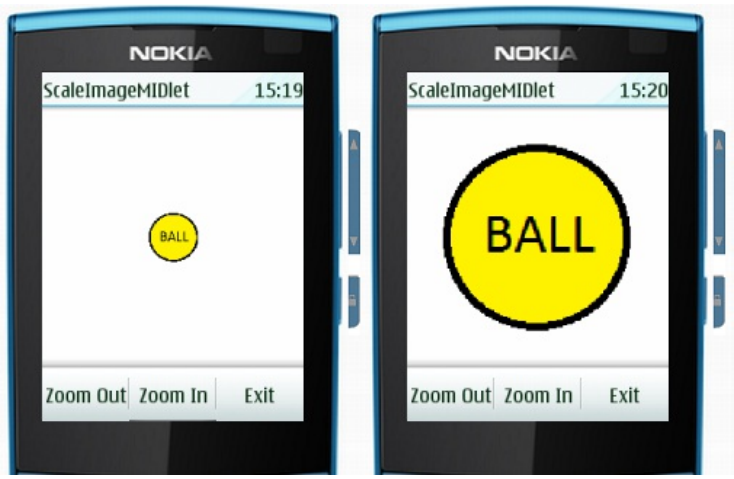
Scaling bitmaps in Java ME

Overview

This code snippet demonstrates how to scale a bitmap in Java ME. The method `scaleImage()` represents the main part of the functionality. The function uses `createRGBImage()` to create the image from array data. So it can also resize images with transparent pixels. The user can resize the image with the "Zoom In" and "Zoom Out" commands.



Source file: ScaleImageMIDlet.java

```
import java.io.IOException;

import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;

import javax.microedition.midlet.MIDlet;


public class ScaleImageMIDlet extends MIDlet {

 /**
  * Reference to MIDlet instance for using it inside ScaleCanvas
  */
 static ScaleImageMIDlet instance;

 /**
  * The Constructor.
  */
 public ScaleImageMIDlet() {
  ScaleImageMIDlet.instance = this;
 }

 /**
  * Called when the MIDlet is started.
  */
 public void startApp() {
  Display.getDisplay(this).setCurrent(new ScaleCanvas());
 }

 /**
  * From MIDlet.
  * Called to signal the MIDlet to enter the Paused state.
  */
 public void pauseApp() {
  // No implementation required.
 }

 /**
  * From MIDlet.
  * Called to signal the MIDlet to terminate.
  * @param unconditional whether the MIDlet has to be unconditionally
  * terminated
  */
 public void destroyApp(boolean unconditional) {
 }

 /**
  * Inner class for handling the canvas.
  */
 class ScaleCanvas extends Canvas implements CommandListener{

  // The original image to use for resize so that quality will stay the same when resizing
  private Image originalImage;

  // The scaled and the image to be drawn
  private Image scaledImage;

  /**
```


(C) Copyright Nokia 2013. All rights reserved.

```
 * exit midlet command
 */
private Command exitCommand;
private Command zoomInCommand;
private Command zoomOutCommand;


private float scaleIndex;

private int screenWidth;
private int screenHeight;

/**
 * Construct the Displayable.
 */
public ScaleCanvas() {
 initialize();
}

/**
 * Initialize self.
 */
void initialize() {

 // screen width and height
 screenWidth = getWidth();
 screenHeight = getHeight();

 scaleIndex = 1.0f;

 try {

  scaledImage = originalImage = Image.createImage("/ball.png");

 } catch (IOException e) {
  //TODO: Handle exception.
 }


 zoomInCommand = new Command("Zoom In", Command.OK, 1);
 zoomOutCommand = new Command("Zoom Out", Command.OK, 2);
 exitCommand = new Command("Exit", Command.EXIT, 3);

 addCommand(zoomInCommand);
 addCommand(zoomOutCommand);
 addCommand(exitCommand);

 // set up this Displayable to listen to command events
 setCommandListener(this);

}

/**
 * From Canvas.
 * Rendering the scene
 */
protected void paint(Graphics g) {
 // fill background
 g.setColor(0xfefefe);
 g.fillRect(0, 0, screenWidth,screenHeight);

 // draw scaled image in middle of screen
 g.drawImage(scaledImage, screenWidth>>1, screenHeight>>1, Graphics.HCENTER | Graphics.VCENTER);

}


/**
 * From CommandListener.
 * Called by the system to indicate that a command has been invoked on a
 * particular displayable.
 * @param command the command that was invoked
 * @param displayable the displayable where the command was invoked
 */

public void commandAction(Command command, Displayable displayable) {
 if (command == exitCommand) {
  instance.notifyDestroyed();
 } else if (command == zoomOutCommand) {
  scaleIndex *= 0.9f;
  if(scaleIndex<0.1f)scaleIndex = 0.1f;
  scaledImage = scaleImage(originalImage,(int) (originalImage.getWidth()*scaleIndex), (int)(originalImage.getHeight()*scaleIndex))
  repaint();
 } else if (command == zoomInCommand) {
  scaleIndex *= 1.1f;
  if(scaleIndex>2.0f)scaleIndex = 2.0f;
  scaledImage = scaleImage(originalImage,(int) (originalImage.getWidth()*scaleIndex), (int)(originalImage.getHeight()*scaleIndex))
  repaint();
 }
}

/**
 * Changes size of image
 * @param sourceImage to be scaled
 * @param newImageWidth is the new width of the image
 * @param newImageHeight is the new height of the image
 */
private Image scaleImage(Image sourceImage, int newImageWidth, int newImageHeight){

 // width of original source image
 int srcWidth = sourceImage.getWidth();
 // height of original source image
 int srcHeight = sourceImage.getHeight();

 // An array for RGB, with the size of original image)
 int rgbSource[] = new int[srcWidth*srcHeight];
 // An array for RGB, with the size of scaled image)
 int rgb2Scaled[] = new int[newImageWidth*newImageHeight];

 // Get the RGB array of source image
 sourceImage.getRGB(rgbSource,0,srcWidth,0,0,srcWidth,srcHeight);

 // calculations and bit shift operations to optimize the for loop
 int tempScaleRatioWidth = ((srcWidth<<16) / newImageWidth);
 int tempScaleRatioHeight = ((srcHeight<<16) / newImageHeight);

 int i = 0;

 for (int y = 0; y < newImageHeight; y++) {
  for (int x = 0; x < newImageWidth; x++) {
```


(C) Copyright Nokia 2013. All rights reserved.

```
      rgb2Scaled[i++]=rgbSource[(srcWidth*((y * tempScaleRatioHeight)>>16))+((x * tempScaleRatioWidth)>>16)];
     }
    }

    //Create an RGB image from rgbScaled array
    return Image.createRGBImage(rgb2Scaled,newImageWidth,newImageHeight,true);
   }

  }

 }
```

Resources

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

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

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.