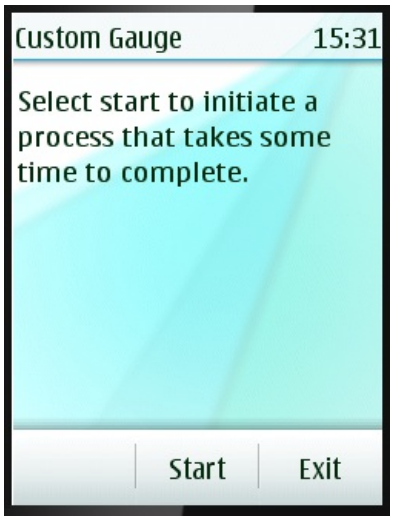
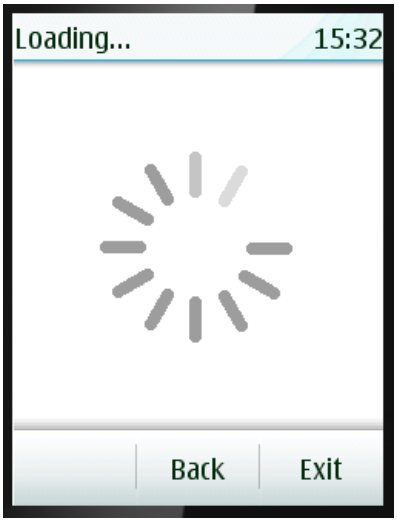
How to create a custom gauge as loading screen with Java ME

This article describes how to create a custom Gauge as loading screen. The standard LCDUI Gauge provides an easy way to append a Gauge to a high level Form at the expense of having to go with a predefined look and animation. It is therefore worth investigating how to use low level UI components, such as a Canvas subclass in order to achieve a different look. In this example we demonstrate how to create a loading screen that looks like a loading bar.

Introduction





This example consists of two main components

- 12 frames that can be found within the Media:GaugeLoaderSource.zip
- The logic to display the frames one after the other within a short period, so that the animation effect is achieved

Note: You will need to extract the 12 frames found in the zip file inside the resource directory of your working project in order for this example to properly work

Animation as a sequence of static images

In this example we used static images that we modified in Microsoft's Paint so that each frame has at least one spike that is entirely blank, and the two preceding ones, are painted in a lighter color compared to the others that have the same solid grey color:

We used the bucket tool (Fill with Color) for this purpose.

The logic behind the animation

In the Canvas subclass, we load the frames as an array of Image instances from the resource directory of the working project. We have named our files consecutively frame1.png, frame2.png etc, so that it gets easier to instantiate the array within a for loop:

```
public CustomGauge() {
     frames = new Image[12];
        nextFrame = 0;
        // The array of frames is filled up from the resource png files
        for(int i = 0; i < 12; i++) {
         try {
```


(C) Copyright Nokia 2014. All rights reserved.

```
    frames[i] = Image.createImage("/frame" + (i+1) + ".png");
   } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
   }
        }
        //The thread moves from one frame to the next
        thread = new Thread(this);
        thread.start();
    }
```

The animation is implemented within a thread, so that an integer variable keeps track of the active frame in a range from 0 to 11 (12 in total). If the active frame becomes the 11th, the frame that should follow is the one with index 0. Inside the thread we define the animation speed by setting the time for which the thread should sleep. A value of 70 milliseconds is used here. By modifying this value a faster or slower animation can be achieved:

```
 public void run() {

  while (true) {
   try {
    //start over if the last frame is active
    if(nextFrame == 11) {
     nextFrame = 0;
    }
    //otherwise increse frame number
    else {
     nextFrame++;
    }
    repaint();
    //animation speed
    Thread.sleep(70);
   }
   catch (InterruptedException e) {
    break;
   }
  }
 }
```

After setting the active frame, the repaint method draws the chosen image in the middle of the screen. Give that the images used in this example have dimension of 133 x 135 pixels, in order to be displayed properly in the middle of the device's screen, we use as anchor point the top left corner of the image, and subtract half of the image's width and height from the screen's center in order to calculate the X and Y coordinates for placing the image's anchor point on the Canvas:

```
   protected void paint(Graphics g) {
     //draws the next frame at the center of screen. The images have width 133 and
height 135 pixels
     g.drawImage(frames[nextFrame], (getWidth() / 2) - 67, getHeight()/2 - 67,
Graphics.TOP|Graphics.LEFT);
 }
```

The MIDlet's code

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
```


(C) Copyright Nokia 2014. All rights reserved.

```
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

public class GaugeMIDlet
 extends MIDlet
 implements CommandListener {

 Display display;

 Command startCommand = new Command("Start", Command.OK, 0);
 Command exitCommand = new Command("Exit",Command.BACK, 0);
 Command backCommand = new Command("Back", Command.ITEM, 0);

 CustomGauge customGauge; //the loading Gauge
 Form mainForm;  //the main control screen

 public void commandAction(Command c, Displayable d) {
  //exits the MIDlet
  if(c == exitCommand){
   notifyDestroyed();
  }
  //starts the custom loading Gauge
  if(c == startCommand) {
   display.setCurrent(customGauge);

  }
  //Interrupts the loading screen
  if(c == backCommand) {
   display.setCurrent(mainForm);
  }
 }
 protected void startApp() throws MIDletStateChangeException {
  display = Display.getDisplay(this);

  //The main control screen's components
  mainForm = new Form("Custom Gauge");
  mainForm.append("Select start to initiate a process that takes some time to
complete.");
  mainForm.addCommand(startCommand);
  mainForm.addCommand(exitCommand);
  mainForm.setCommandListener(this);
  display.setCurrent(mainForm);

  //The loading Gauge's commands
  customGauge = new CustomGauge();
  customGauge.setTitle("Loading...");
  customGauge.addCommand(exitCommand);
  customGauge.addCommand(backCommand);
  customGauge.setCommandListener(this);
 }

 protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
  //To do
```


(C) Copyright Nokia 2014. All rights reserved.

```
 }

 protected void pauseApp() {
  //To do
 }
}
```

The Canvas subclass

```
import java.io.IOException;
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;

class CustomGauge
 extends Canvas
 implements Runnable {


    Thread thread;
    Image[] frames;  //the array of frames
    int nextFrame;  //the running thread updates the next Frame

    public CustomGauge() {
     frames = new Image[12];
        nextFrame = 0;
        // The array of frames is filled up from the resource png files
        for(int i = 0; i < 12; i++) {
         try {
    frames[i] = Image.createImage("/frame" + (i+1) + ".png");
   } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
   }
        }
        //The thread moves from one frame to the next
        thread = new Thread(this);
        thread.start();
    }
 public void run() {

  while (true) {
   try {
    //start over if the last frame is active
    if(nextFrame == 11) {
     nextFrame = 0;
    }
    //otherwise increse frame number
    else {
     nextFrame++;
    }
    repaint();
    //animation speed
    Thread.sleep(70);
   }
```


(C) Copyright Nokia 2014. All rights reserved.

```
    catch (InterruptedException e) {
     break;
    }
   }
  }

  protected void paint(Graphics g) {
      //draws the next frame at the center of screen. The images have width 133 and
height 135 pixels
      g.drawImage(frames[nextFrame], (getWidth() / 2) - 67, getHeight()/2 - 67,
Graphics.TOP|Graphics.LEFT);
  }
}
```

Resources

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

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

See also

- The LCDUI Gauge
- JavaDocs for the Gauge Class

(C) Copyright Nokia 2014. All rights reserved.