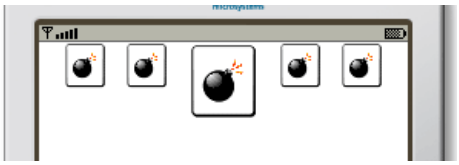
J2ME Fisheye Menu with JSR 226

A simple Fisheye menu can be created with JSR 226.



Following is the actual implementation. Some details:

- `paint()` method is used to actually paint the menu on a Graphics instance
- `keyPressed()` method is used to handle keypress events, so, to scroll menu
- `mustRepaint()` is useful to know if menu requires a repaint or not

The rest is quite all size/coords calculations :)

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Graphics;
import javax.microedition.m2g.ScalableGraphics;
import javax.microedition.m2g.ScalableImage;

public class SvgFisheye
{
 ScalableGraphics sg = null;

 ScalableImage[] icons = null;
 int itemsNum = 0;

 int bgColor = 0xffffff;

 int currentItem = 0;
 int previousItem = 0;

 int width = 0;
 int height = 0;

 int iconPadding = 10;

 static int baseIconSide = 30;
 static int currentIconSide = 50;


 static final int TRANSITION_STEPS = 5;
 int currentStep = TRANSITION_STEPS;
 int direction = 0;

 boolean mustRepaint = true;

 public SvgFisheye(ScalableImage[] icons, int width, int height)
 {
  sg = ScalableGraphics.createInstance();

  this.width = width;
  this.height = height;

  this.icons = icons;
```


(C) Copyright Nokia 2014. All rights reserved.

```
  this.itemsNum = icons.length;
 }

 public void paint(Graphics g)
 {
  int cx, cy, cw, ch;
  cx = g.getClipX();
  cy = g.getClipY();
  cw = g.getClipWidth();
  ch = g.getClipHeight();

  g.setClip(0, 0, width, height);

  int currentSide = currentIconSide - (TRANSITION_STEPS - currentStep) *
(currentIconSide - baseIconSide) / TRANSITION_STEPS;
  int previousSide = baseIconSide + (TRANSITION_STEPS - currentStep) * (currentIconSide
- baseIconSide) / TRANSITION_STEPS;

  int center = 0;

  if(direction == 0)
  {
   center = (currentItem) * (iconPadding + baseIconSide) + currentSide / 2;
  }
  else if(direction == Canvas.RIGHT)
  {
   int prevCenter = (currentItem - 1) * (iconPadding + baseIconSide) +
    previousSide / 2;

   int currentCenter = prevCenter + previousSide / 2 + iconPadding +
    currentSide / 2;

   center = prevCenter + (currentCenter - prevCenter) * currentStep / TRANSITION_STEPS;
  }
  else if(direction == Canvas.LEFT)
  {
   int currentCenter = currentItem * (iconPadding + baseIconSide) +
    currentSide / 2;

   int prevCenter = currentCenter + currentSide / 2 + iconPadding +
    previousSide / 2;

   center = prevCenter + (currentCenter - prevCenter) * currentStep / TRANSITION_STEPS;
  }

  g.setColor(bgColor);
  g.fillRect(0, 0, width, height);

  int left = width / 2 - center;

  sg.bindTarget(g);

  for(int i = 0; i < itemsNum; i++)
  {
   int iconSide = i == currentItem ? currentSide :
    (i == previousItem ? previousSide : baseIconSide);
```


(C) Copyright Nokia 2014. All rights reserved.

```
    icons[i].setViewportWidth(iconSide);
    icons[i].setViewportHeight(iconSide);

    sg.render(left, 0, icons[i]);

    left += iconSide + iconPadding;
   }

   sg.releaseTarget();

   move();

   g.setClip(cx, cy, cw, ch);
  }

  public boolean mustRepaint()
  {
   return mustRepaint;
  }
  private boolean isMoving()
  {
   return currentStep < TRANSITION_STEPS;
  }
  private void move()
  {
   if(currentStep < TRANSITION_STEPS)
   {
    mustRepaint = true;

    currentStep++;

    if(currentStep == TRANSITION_STEPS)
    {
     direction = 0;
    }
   }
   else
   {
    mustRepaint = false;
   }
  }
  private void startMove()
  {
   currentStep = 0;

   mustRepaint = true;
  }

  public void keyPressed(int keyCode)
  {
   int delta = keyCode == Canvas.RIGHT ? 1 : (keyCode == Canvas.LEFT ? -1 : 0);

   if(delta != 0 && !isMoving() && currentItem + delta >= 0 && currentItem + delta <
itemsNum)
   {
    previousItem = currentItem;
```


(C) Copyright Nokia 2014. All rights reserved.

```
    currentItem = currentItem + delta;

    startMove();

    direction = keyCode;
   }
  }
 }
```


(C) Copyright Nokia 2014. All rights reserved.