

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 g;

    ScalableImage = null;
    int itemsNum = 0;

    int bgColor = 0xffffffff;

    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)
    {
        g = ScalableGraphics.createInstance();

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

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

J2ME_Fisheye_Menu_with_JSR_226

```

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

    setClip(0, 0, width, height);

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

    int center = 0;

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

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

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

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

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

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

    int left = width / 2 - center;

    bindTarget(g);

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

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

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

        left += iconSide + iconPadding;
    }

    releaseTarget();
}

```

J2ME_Fisheye_Menu_with_JSR_226

```

        ();          move

    setClip(cx, cyg.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)
    {
        currentItem = currentItem + delta;
        previousItem = currentItem - 1;
        nextItem = currentItem + 1;

        startMove();

        direction = keyCode;
    }
}
}
}

```