Animation in JavaME

Warning: This is one of my first projects in JavaME (some years ago). Improvements has to be done, but for while take this "AS IS". At the beginning I did this little program only to understand the API. I hope this help you in some way in your Java ME journey.

In this app you control an actor (left/right), jump (up), change background (down).

Go to my projects and grab Animation project (with all resources and full source code). You can easily open it in NetBeans 6.9

Please don't forget to check my game projects (need feedback): Queops, LunarPatrol & X-Rally.

```
/**
 * by George Roberto Peres.
 * Santos - Sao Paulo - Brazil
 */

import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 *
 * @author George Roberto Peres
 *         e-mail: reflexus@ig.com.br
 *         favor nao exitar em me contactar !
 */

public class Run extends MIDlet implements CommandListener {

 private Display tela;  // objeto Display unico para este MIDlet
  private Animacao animacao;  // instancia do nosso jogo GameCanvas
  private Command exitComando = new Command("Exit", Command.BACK, 1);
  private Command pauseComando = new Command("Pause", Command.SCREEN, 1);
  private Command resumeComando = new Command("Resume", Command.SCREEN, 1);

  public Run() {
    try {
      tela = Display.getDisplay(this);
      animacao = new Animacao(this);
      animacao.addCommand(exitComando);
      animacao.addCommand(pauseComando);
      animacao.setCommandListener(this);
      animacao.start();
    }
    catch (Exception e) { }
  }

  public void startApp() {
    tela.setCurrent(animacao);
  }

  public void pauseApp() {  }

  public void pauseJogo() {
    animacao.removeCommand(pauseComando);
    animacao.addCommand(resumeComando);
    animacao.stop();
  }

  public void resumeJogo() {
    animacao.removeCommand(resumeComando);
    animacao.addCommand(pauseComando);
    animacao.start();
  }

  /*
   * The Application Manager calls this method to give you a chance to do
   * something (such as save state and release resources) before your app
   * is closed.
   * @param unconditional if false you have the option of throwing a
   * MIDletStateChangeException to abort the destruction process.
   */
  public void destroyApp(boolean unconditional) throws MIDletStateChangeException {
    Display.getDisplay(this).setCurrent(null);
    tela = null;
    if (animacao!=null) animacao.stop();
    System.gc();
    notifyDestroyed();
  }

  /** Converte uma excecao numa mensagem e mostra a mensagem..
   *  @param e  -  excecao
   */
  public void erroMsg(Exception e) {
    if (e.getMessage() == null) {
      erroAlerta(e.getClass().getName());
    } else {
      erroAlerta(e.getClass().getName() + ": " + e.getMessage());
    }
  }


  /** Mostra um alerta de mensagem de erro se algo errado acontecer.
   *  @param msg  -  a String proveniente de erroMsg(Exception e)
   */
  public void erroAlerta(String msg) {
    Alert errorAlert = new Alert("error", msg, null, AlertType.ERROR);
    errorAlert.setCommandListener(this);
    errorAlert.setTimeout(Alert.FOREVER);
```


(C) Copyright Nokia 2013. All rights reserved.

```
    Display.getDisplay(this).setCurrent(errorAlert);
  }


  /**
   * O metodo da interface de CommandListener chamado quando usuario executa
   * um comando, neste caso 3 comandos.
   * @param command
   * @param displayable
   */
  public void commandAction(Command c, Displayable d) {
    if (c == exitComando) {
      try { destroyApp(false); }
      catch (MIDletStateChangeException e) {
        erroMsg(e);
      }
    }
    if (c == pauseComando) {
      pauseJogo();
    }

    if (c == resumeComando) {
      resumeJogo();
    }
  }
}
```

```
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.game.Sprite;

/**
 *
 * @author George Roberto Peres
 *         reflexus@ig.com.br
 */
public class Actor {

  public int x;
  public int y;
  public Sprite spriteActor;
  private static Image actor;

  // Construtor e inicializacao
  public Actor() throws Exception {
    actor = Image.createImage("/actor.png"); // Carrega arquivo de imagem
    spriteActor = new Sprite (actor, Tools.actorLarg, Tools.actorAlt);
    spriteActor.setFrameSequence(Tools.ACTORSEQ);
  }

  /** coloca actor no plano cartesiano */
  public void setXY(boolean dir) {
    if (dir) {
      spriteActor.setTransform(Sprite.TRANS_NONE);
      spriteActor.setPosition(x,y);
    }
    else {
      spriteActor.setTransform(Sprite.TRANS_MIRROR);
      spriteActor.setPosition(x,y);
    }
  }

  public void pula(boolean dir, int passo) {
    if (dir) {
      spriteActor.setPosition(x + Tools.puloX[passo], y + Tools.puloY[passo]);
    }
    else {
      spriteActor.setPosition(x - Tools.puloX[passo], y + Tools.puloY[passo]);
    }
  }
}
```

```
/*
 * Esta classe define os metodos que praticamente todas as classes usam.
 * telaLarg()  e telaAlt() definem a resolucao em que o jogo vai rodar,
 * Para alterar esses numeros estaticos alterar somente aqui, pois as outras
 * classes envolvidas ja contem codigos para lidar com esses numeros.
 *
 * Obs: todas as medidas em pixel.
 * Obrigatorio estar em tira horizontal e cada frame espacado
 * uniformemente (a largura do png tem que ser multiplo do sprite)
 *
 * Estao aqui para facil portabilidade para outras resolucoes.
 *
 */

import javax.microedition.lcdui.Graphics;

public final class Tools {

  public static final int GRAPHICS_TOP_LEFT = Graphics.LEFT | Graphics.TOP;

  // Dimensoes da tela Game Canvas
  public static final int telaLarg = 240;   // largura da tela (x)
  public static final int telaAlt = 291;   // altura da tela (y) 309,320,291

  // Dimensoes do personagema  (Actor)
  public static final int actorLarg = 32;   // largura do personagem (x)
  public static final int actorAlt = 48;   // altura do personagem (y)
  public static final int ACTORSEQ[] = { 0,1,2,3,4,5,6,7 };   // Actor frameSeq

  // valores pre-definidos para a parabola do pulo
  public static final int puloX[] = { 0,0,0,0,1,1,2,3,4,5,6,7,8,9,10,11,12,13,14,
  15,16,17,18,19,20,20,21,21,21,21 };
  public static final int puloY[] = { 0,-1,-2,-3,-4,-5,-6,-7,-8,-9,-9,-10,-10,
  -10,-11,-11,-10,-10,-10,-9,-9,-8,-7,-6,-5,-4,-3,-2,-1,0 };

}
```

```
/**
 * Animacao. (c) GRP Software. ALL RIGHTS RESERVED!
 * by George Roberto Peres.
```


(C) Copyright Nokia 2013. All rights reserved.

```
 * Santos - Sao Paulo - Brazil
 *
 */

import java.io.IOException;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.game.GameCanvas;
import javax.microedition.lcdui.game.LayerManager;
import javax.microedition.lcdui.game.Sprite;

/**
 * @author George Roberto Peres
 *         e-mail: reflexus@ig.com.br
 *         favor nao exitar em me contactar !
 */
public class Animacao extends GameCanvas implements Runnable {

  private Graphics g = getGraphics();
  private Run run;
  private byte c;   // contador interno de ciclos usado para timing
  private int fundo;   // qual numero de fundo utilizaremos
  private int passo;
  private boolean isPlay;   // true = estamos jogando
  private static final int QUANT_FUNDOS = 4;   // total de backGrounds (comeca em 1)
  private Actor actor;
  private boolean f_actor;
  private boolean dir;
  private boolean move;
  private boolean pulo;
  private boolean auxFrame;
  private Image imageFundo;   // a imagem de fundo = background#.png
  private Sprite spriteFundo;   // sprite da tela de fundo
  private LayerManager gerenteDeCamadas;   // usado para remover/trocar o fundo

  /**
   * Construtor e inicializacao
   */
  public Animacao(Run midlet) {
    super(false);
    run = midlet;
    fundo = 1;
    gerenteDeCamadas = new LayerManager();
    gerenteDeCamadas.setViewWindow(0, 0, Tools.telaLarg, Tools.telaAlt);
    carregaFundo(fundo);
    atualizaVars();
    g.setFont(Font.getFont(Font.FACE_SYSTEM, Font.STYLE_BOLD, Font.SIZE_SMALL));
  }


  /** atualiza as principais variaveis do jogo */
  private void atualizaVars() {
    c = 0;
    dir = true;
    if (actor == null) {
      try { actor = new Actor(); }
      catch (Exception e) { }
    }
    f_actor = true;
    actor.x = Tools.telaLarg/2 - Tools.actorLarg/2;
    actor.y = 140;
    actor.setXY(dir);
  }


  /** Inicia a thread = loop do jogo */
  public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
  }


  /** Pausa esta thread */
  public void stop() {
    isPlay = false;
    g.setColor(255, 255, 0); // amarelo
    g.drawString("P A U S E", Tools.telaLarg/2 - 22, Tools.telaAlt/2, Tools.GRAPHICS_TOP_LEFT);
    repaint();
  }


  /** limpa keystates */
  public void limpaTeclado() {
    getKeyStates();
  }

  /** carrega nova imagem imageFundo = background#.png
   *  @param i numero do fundo do arquivo png
   */
  private void carregaFundo(int i) {
    try {
      imageFundo = Image.createImage("/background" + i + ".png");
      spriteFundo = new Sprite(imageFundo);
      gerenteDeCamadas.append(spriteFundo);
    }
    catch (IOException e) { }
  }


  /** troca o plano de fundo - backGround */
  private void trocaFundo() {
    fundo += 1;
    if (fundo==QUANT_FUNDOS+1) fundo=1;
    gerenteDeCamadas.remove(spriteFundo);
    // estamos trocando de fundo entao zere tudo, liberando os
    // preciosos recursos = memoria usada (garbage collector)
    spriteFundo = null;
    imageFundo = null;
    carregaFundo(fundo);
  }


  /** If the game is hidden by another app (or a menu) */
  public void hideNotify() {
    stop();
    run.pauseJogo();
  }
```


(C) Copyright Nokia 2013. All rights reserved.

```
  /** When it comes back into view */
  public void showNotify() { }

  /** Loop Principal do jogo. Roda quando isPlay = true */
  public void run() {
    while (isPlay) {
      c++;   //  timing
      if (c==60) c = 0;
      input();
      // se actor nao estiver movendo entao setFrame(3)
      if (!move & !auxFrame) { auxFrame = true;  actor.spriteActor.setFrame(3); }
      if (!pulo) {
        if (f_actor) {
          actor.setXY(dir);
          if (c%4==0 & move) {
            actor.spriteActor.nextFrame();
          }
          move = false;
        }
      } else {
        actor.spriteActor.setFrame(6);  // mudamos para o frame do pulo
        actor.pula(dir, passo);
        passo += 1;
        if (passo == Tools.puloX.length) {
          pulo = false;
          if (dir) actor.x += Tools.puloX[passo-1]; else actor.x -= Tools.puloX[passo-1];
          passo = 0;
          actor.spriteActor.setFrame(3);   //  ao aterrisar do pulo mude frame
        }
      }
      // desenha
      desenhaTela(g);
      // da uma pausa na execucao: para o jogo nao ficar muito rapido e tambem
      // para correto funcionamento da leitura do teclado em input()
      try { Thread.sleep(1); }
      catch (InterruptedException e) { }
    }
  }

 /**
  * Metodo para lidar com entrada do teclado.
  */
  private void input() {
    int keyStates = getKeyStates();
    // LEFT
    if ((keyStates & LEFT_PRESSED) != 0 ) {
      if (actor.x>0 & !pulo) {
        dir = false;
        if (!pulo) move = true; else move = false;
        if (f_actor) actor.x -= 1;
        auxFrame = false;
      }
    }
    // RIGHT
    if ((keyStates & RIGHT_PRESSED) !=0 ) {
      if (actor.x < Tools.telaLarg - Tools.actorLarg & !pulo) {
        dir = true;
        if (!pulo) move = true; else move = false;
        if (f_actor) actor.x += 1;
        auxFrame = false;
      }
    }
    // UP
    if ((keyStates & UP_PRESSED) != 0) {
      if (!pulo) {
        if (actor.x>0 & actor.x<Tools.telaLarg-Tools.actorLarg) {
          pulo = true;
          //actor.spriteActor.setFrame(2);
        }
      }
    }
    // DOWN
    if ((keyStates & DOWN_PRESSED) !=0) {
      trocaFundo();
    }
    // FIRE
    if ((keyStates & FIRE_PRESSED) !=0) {
      //
    }
  }

  // desse jeito ta desenhando tudo de uma so vez: mudar para desenhar
  // apenas o que mudou e precisa ser desenhado (mais rapido).
  /** Metodo usado para desenhar e mostrar graficos
   *  @param g a tela que e' representado pela nossa GameCanvas
   */
  private void desenhaTela(Graphics g) {
    // desenha tela de fundo
    gerenteDeCamadas.paint(g,0,0);
    g.setColor(0, 0, 0);
    g.fillRect(0, 200, Tools.telaLarg, Tools.telaAlt - 200);\
    // verificar se ativo antes de desenhar
    if (f_actor) actor.spriteActor.paint(g);
    repaint(0, 0, Tools.telaLarg, 200);
  }
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.