1 – Estágio Um montagem básica do projeto:

Código inicial com controle de frames:

**package** me.rlbpc.main;

**import** java.awt.Canvas;

**import** java.awt.Color;

**import** java.awt.Dimension;

**import** java.awt.Graphics;

**import** java.awt.image.BufferStrategy;

**import** java.awt.image.BufferedImage;

**import** javax.swing.JFrame;

**public** **class** Game **extends** Canvas **implements** Runnable {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** **static** JFrame *frame*;

**private** Thread thread;

**private** **boolean** isRunning = **true**;

**private** **final** **int** WIDTH = 180;

**private** **final** **int** HEIGHT = 160;

**private** **final** **int** SCALE = 4; //Usar o scale para aumentar ou diminuir a janela

**private** BufferedImage image;

**public** Game() {

**this**.setPreferredSize(**new** Dimension(getWIDTH() \* getSCALE(), getHEIGHT() \* getSCALE()));

initFrame();

image = **new** BufferedImage(getWIDTH(),getHEIGHT(),BufferedImage.***TYPE\_INT\_BGR***);

}

**public** **void** initFrame() {

*setFrame*(**new** JFrame("Primeira Janela do Jogo"));

*getFrame*().add(**this**);

*getFrame*().setResizable(**false**);

*getFrame*().pack(); //calibra as dimensões do frame usando o canvas

*getFrame*().setLocationRelativeTo(**null**);

*getFrame*().setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

*getFrame*().setVisible(**true**);

}

**public** **synchronized** **void** start() {

thread = **new** Thread(**this**);

isRunning = **true**;

thread.start();

}

**public** **synchronized** **void** stop() {

isRunning = **false**;

**try** {

thread.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

**public** **static** **void** main (String args[]) {

Game game = **new** Game();

game.start();

}

**public** **static** JFrame getFrame() {

**return** *frame*;

}

**public** **static** **void** setFrame(JFrame frame) {

Game.*frame* = frame;

}

**public** **void** tick() {

}

**public** **void** render() {

BufferStrategy bs = **this**.getBufferStrategy();

**if** (bs == **null**) {

**this**.createBufferStrategy(3);//usar de 2 a 3

**return**;

}

Graphics g = image.getGraphics();

//PREPARA As IMAGENS PARA SEREM APRESENTADAS

g.setColor(**new** Color(0,0,0));

g.fillRect(0,0,getWIDTH(),getHEIGHT());

//Renderização do jogo

//Graphics2D g2 = (Graphics2D) g; //casting do g para gráficos 2D

//APRESENTA A IMAGEM NO FRAME

g.dispose(); //limpar dados de imagem otimiza a performance

g = bs.getDrawGraphics();

g.drawImage(image,0,0,getWIDTH()\*SCALE,getHEIGHT()\*SCALE,**null**);

bs.show();

}

**public** **void** run() {

**long** lastTime = System.*nanoTime*();

**double** amountOfTicks=60.0; //FPS rate

**double** ns = 1000000000 / amountOfTicks;

**double** delta = 0;

**int** frames = 0;

**double** timer = System.*currentTimeMillis*();

//Looping principal do jogo

**while** (isRunning) {

**long** now = System.*nanoTime*();

delta += (now - lastTime) / ns;

lastTime=now;

**if**(delta>=1) {

tick();

render();

frames++;

delta--;

}

**if**(System.*currentTimeMillis*() - timer >= 1000) {

System.***out***.println("FPS: "+ frames);

frames = 0;

timer = System.*currentTimeMillis*();

}

}

stop(); //segurança para que as threads parem se ocorrer algum problema e liberem os recursos do computador

}

**public** **int** getWIDTH() {

**return** WIDTH;

}

**public** **int** getHEIGHT() {

**return** HEIGHT;

}

**public** **int** getSCALE() {

**return** SCALE;

}

}