|  |  |
| --- | --- |
| public class Game implements Runnable {  private GameWindow gameWindow;    private GamePanel gamePanel;    private Thread gameThread;    /\*- Tạo ra FPS ổn định cho game - \*/  private final int FPS\_SET = 120;    public Game() {  gamePanel = new GamePanel();  gameWindow = new GameWindow(gamePanel);    /\* -Nhận được sự kiện từ bàn phím -\*/  gamePanel.requestFocus();  /\*- Khởi chạy luồng - \*/  StartGameLoop();  }  private void StartGameLoop() {  gameThread = new Thread(this);  gameThread.start();  }  @Override  public void run() {    /\* - Biến lưu tru thoi luong moi khung hinh sẽ kéo dài - \*/  double timePerFrame = 1000000000.0/ FPS\_SET;    long lastFrame = System.nanoTime();    long now = System.nanoTime();    long lastCheck = System.currentTimeMillis();  int frames =0;  while (true){  now = System.nanoTime();  /\* - Dùng để kiểm tra xem đã đủ thời gian vẽ một frame mới chưa - \*/  if(now - lastFrame >= timePerFrame) {  /\*- Chúng ta cần gọi lại để hiển thi h/a -\*/  gamePanel.repaint();  lastFrame = now;  frames++;  }  /\* - Dùng để kiểm tra xem đã đủ 1s để in ra số frame có trong 1 giây đó - \*/  /\* - Nếu một giây đã trôi qua kể từ lần kiểm tra khung hình/giây cuối cùng, chúng tôi sẽ thực hiện kiểm tra khung hình/giây mới.  Lưu lần kiểm tra FPS mới làm lần kiểm tra FPS cuối cùng và lặp lại- \*/    /\* 1 second (1 giây ) = 1000 mili giây \*/  if(System.currentTimeMillis() - lastCheck >= 1000) {  lastCheck = System.currentTimeMillis();  System.out.println("FPS: " + frames);  frames = 0;  }  }    }      }  63 576 650 059 300 nanosecond | * Bài này liên quan đến việc tạo ra một cái luồng riêng biệt bằng cách implements interface Runable sau đó sẽ ghi đè lên phương thức run   public class Game implements Runable {  private Thread thread;  public Game (){  startGameLoop();  }  private void StartGameLoop (){  gameThread = new Thread(this);  gameThread.start();  }    }   * Tạo một biến hằng số xác định số lượng frame (khung hình) mỗi giây   private final int FPS\_SET =120 ; (final = const = hằng số )   * Trong hàm ghi đè run :   + double timePerFrame = 1000000000/ FPS\_SET (1 tỷ / 120 ) xác định thời lượng  (đơn vị nano giây) cần chờ giữa mỗi frame để đạt được số FPS mong muốn.  (1 nanosecond = 1ns = 10 ^ -9 )  + long lastFrame = System.nanoTime() là biến lastFrame lưu trữ thời điểm cuối cùng một frame đã được vẽ.  (System.nanoTime() là thời điểm hiện tại tính bằng nanosecond  1 nanosecond = 10 ^ -9 (giây) )  + Biến now lưu trữ thời điểm hiện tại tính bằng nanosecond()  + frame lưu trữ số lượng khung hình được vẽ trong 1 giây.  While (true) {   * tạo ra một cái vòng lặp vô hạn   now = thời gian hiện tại tính theo giây \* 10 ^ -9  nếu (Thời gian hiện tại này(10^ -9 ++++ ) – thời điểm mà frame cuối cùng được vẽ >= 1 tỷ / 120 Frames) {  + Tiến hành vẽ hình  + Tăng số lượng frame    }  Nếu ( Thời gian hiện tại tính theo miligiay – Thời gian cuối cùng kiểm  Tra số FPS > 1000) {  + Ỉn ra cái frame đó và cập nhật lại số lượng frame = 0;  }  } |

package main;

import java.awt.Color;

import java.awt.Graphics;

import java.util.ArrayList;

import java.util.Random;

import input.Mouse;

import input.KeyboardInputs;

import javax.swing.JPanel;

public class GamePanel extends JPanel {

private Mouse mouseInputs;

/\* - Theo dõi vị trí HCN- \*/

private float xDelta= 100, yDelta = 100;

/\* - Dùng để điều chỉnh huong HCN - \*/

private float xDir = 1f, yDir =1f;

/\* - Dùng để điều chỉnh màu cảu nó - \*/

private Color color = new Color(10,30,50);

/\* - Biến này dùng để tạo ramdom cho màu sắc \*/

private Random random ;

/\*- Mảng lưu trữ hình chữ nhật \*/

private ArrayList<MyRect> rects = new ArrayList<>();

public GamePanel() {

random = new Random();

mouseInputs = new Mouse(this);

addKeyListener(new KeyboardInputs(this));

addMouseListener(mouseInputs);

addMouseMotionListener(mouseInputs);

}

public void changeXDelta(int value) {

this.xDelta += value;

}

public void changeYDelta(int value) {

this.yDelta += value;

}

public void setRectPos(int x, int y) {

this.xDelta = x;

this.yDelta = y;

}

public void spawnRect(int x, int y) {

rects.add(new MyRect(x, y));

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

for (MyRect rect : rects) {

rect.updateRect();

rect.draw(g);

}

updateRectangle();

g.setColor(color);

g.fillRect((int)xDelta,(int) yDelta, 200, 50);

}

private void updateRectangle() {

xDelta +=xDir;

if(xDelta > 700 || xDelta < 0) {

xDir \*= -1;

color = getRandomColor();

}

yDelta += yDir;

if(yDelta > 400 || yDelta < 0 ) {

yDir \*=-1;

}

}

private Color getRandomColor() {

/\* - Trả về các só trong khoang tu 0 - 255 \*/

int r =random.nextInt(255);

int b = random.nextInt(255);

int g = random.nextInt(255);

return new Color(r,b,g);

}

// Temp

public class MyRect {

int x, y, w, h;

int xDir = 1, yDir = 1;

Color color;

public MyRect(int x, int y) {

this.x = x;

this.y = y;

w = random.nextInt(50);

h = w;

color = newColor();

}

public void updateRect() {

this.x += xDir;

this.y += yDir;

if ((x + w) > 700 || x < 0) {

xDir \*= -1;

color = newColor();

}

if ((y + h) > 400 || y < 0) {

yDir \*= -1;

color = newColor();

}

}

private Color newColor() {

return new Color(random.nextInt(255), random.nextInt(255), random.nextInt(255));

}

public void draw(Graphics g) {

g.setColor(color);

g.fillRect(x, y, w, h);

}

}

}