17.11.2024

1:00 AM

The rectangle is moving

The errors I encountered:

1.

package Display;  
  
import javax.swing.\*;  
  
public class Main {  
 public static void main(String[] args) {  
  
 //allows the window to appear on screen  
 JFrame window= new JFrame();  
 window.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);//if this were not here, program would run even after closing the window  
 window.setResizable(false);  
 window.setTitle("Maze explorer");  
  
 Panel panel = new Panel();//the panel is what is displayed in the window we oppened  
 window.add(panel);//we add it to a JFrame  
  
 window.setLocationRelativeTo(null);  
 window.setVisible(true);  
  
  
 panel.requestFocusInWindow();  
  
 window.pack();//this is done to actually see the panel  
 //we can display 16 48\*48 tiles  
  
 panel.startTheThread();  
 }  
  
}

In the original code, panel was declared after the setLocationRelativeTo and the set Visible, so I couldn’t grant keyHandler the focus it needed to check the input.

2.

Yes, I am embarrassed. Yes, I copy and pasted the code from keypress to keyReleased and forgot to change true to false. No, I will not say anything about that.

Stuff to talk about:

1. To make the game loop work: 2 methods-> thread sleep ->checking the time

* Next day: check with delta.

//but the game doesn't wait for user input, it keeps running  
//we need to set an FPS to see how fast we update  
  
//but where does the player sit? If we do the logical, seemingly realistic action of loading the map and moving across  
//it, we'd run into memory issues along the road, since a lot of games (Elden Ring, my beloved, are very detailed)  
//In 3D it's a bit more complicated, but in 2D, we can simply not move the character, but the map itself  
  
//i add this for the future. if we add a 2\* speed feature : Rafa from future: nah, i'd skip that

//for our game loop, due to how the methods are used, the timem is in nanoseconds  
//nano = 10^(-9), so the draw interval is 1sec/ FPS, so every one 60th of a second, we want to do the update and redrawing  
  
//double nextDrawTime = System.nanoTime() + drawInterval;///System.nanoTime-> current timem  
//our loop starts, let's say, at a timem t, so we start measuring from that onward.  
//the next cycle must be after a drawInterval has passed=> t + drawInterval  
  
//second method: use a delta timer

*///to ensure the 1/FPS game loop cycle, we have 2 methods: making the thread sleep until the necessary timem has passed:  
/// problem: not fully stable, can have some delays of a couple milliseconds  
 ///calculating the timem in which we should start the process: aka the DELTA method*