Tecnológico de Monterrey

Juan Pablo Aboytes Novoa A01701249

Horse Racing Simulation.

Programming Languages

21/11/19

Index

Context of the problem

In an environment where the majority are students, a means of distraction is necessary and there are many ways, such as playing a musical instrument, watching series or playing video games, which is what I personally prefer.

Sometimes we also need to make decisions and we don't know which decision to make, so what better way to decide than with a random game!

Solution

As I said before I have a taste for simulation games, and I wanted to make a project which interests me and I like it and therefore I decided to make a "mini game" and also because java is the best option for the management of a project thought of objects for the ease of handling them and modifying their properties. among the ones we had .

The reason why I chose the theme of a horse race simulated by my taste for simulation games, games that each game is different, I will speak again about the reason why do this project in the next section but oriented to knowledge acquired in this semester in the subject of programming languages

Horse Racing Simulation (non-technical description).

It is a representation of a horse race in a section of a racecourse, having as participants two runners (players) who will have to run a maximum of three times to find a winner in case a player gets two points or a draw, this It happens when each player gets two points.

But ... How do you get the points? As it is a simulation game, it only remains to observe! A point is obtained when a runner reaches the finish line or when both runners reach the finish line at the same time.

Organization.

From here begins a more oriented language for people with programming skills.

The project is divided into four classes:

1.- **The Start class:** It is our start of the game, it is a window (later it will be explained more accurately) where you must click on the "Play" button to be able to go to the game screen.

2.- **The Race_s class:** It is our class that will function as our threads, it is the one with the classic "infinite cycle" and the logistics of the game.

3.- **The RaceFrame2 class:** It is the window in which the game takes place where they contain all the graphic elements that we will also talk more about later and where the two Race_s type threads are created.

4.- **The End class:** Having the result (the victory of a player or a draw) a window will be displayed for the purpose of the game.

Explanation of the code

In this part of the report I want to better explain the components and code of each class, I also want to explain the reason for using what I used.

Start class:

Library or interface used**:**

- **Swing**: This is a graphical library, I used this library in order to create the JFrame that contains components that at the same time contains methods that make possible the transition between JFrames.

Components used:

- **JLabel**: I used 1 JLabel and I modified the text property in order to not show text and added the racing_1.png to the icon property.
- **JButton**: I used a button for the transition between the JFrame of the Start class and the JFrame of the Race_s class, I modified the show and icon property but I added the PlayButtonRed.png.

Functions used:

First I declared the JButton and JLabel variables.

- **Start(void) Function**: It calls the initComponents function.
- **initComponents(void) Function**: Creation and definition of the JButton and JLabel, more specifically is where I defined the properties and layout of the components.
- **jButton1ActionPerformed(void) Function**: This is a function assigned to the JButton, here is where I created an instance of RaceFrame2 and made the RaceFrame2 JFrame visible and made invisible the Start JFrame
- **main(void) Function**: It calls the Start function and make it visible (the Start class JFrame)

Image:

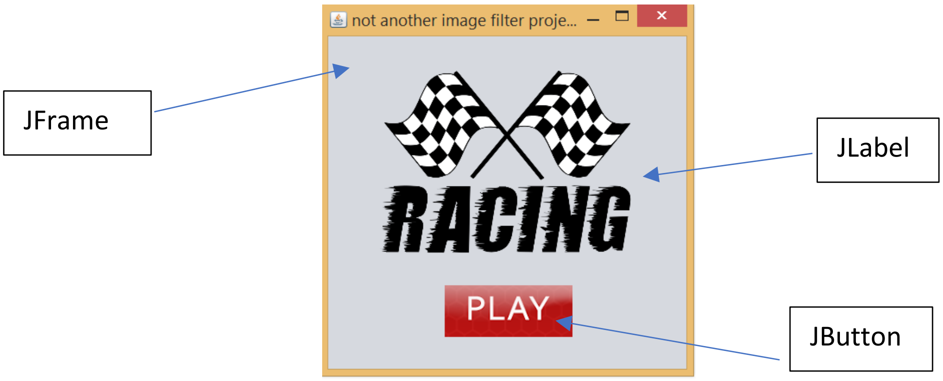


Image 1.0: Start JFrame

RaceFrame2 class:

Library or interface used**:**

- **Swing**: This is a graphical library, I used this library in order to create the JFrame that contains components that at the same time contains methods that make possible the transition between JFrames.
- **AudioClip**: Interface that is a simple abstraction for playing a sound clip.

Components used:

- **JLabel**: To simulate the race logs, it will show who is the winner of the round(explained in the Race_class run() function), to simulate the score board, to simulate the runners, finish line and the fences of the racecourse. I used the playeers.png, finishline_4.png and bardatam.png in the icon property of the last three JLabel mentioned.
- **JButton**: I used a button for creating two threads, it start the song and the transition between the JFrame of the Start class and the JFrame of the Race_s class, I modified the show and icon property but I added the PlayButtonRed.png.

Functions used:

First I declared the JButton and JLabel variables.

- **RaceFrame2(void) Function**: It calls the initComponents function.
- **initComponents(void) Function**: Creation and definition of the JButton and JLabels, more specifically is where I defined the properties and layout of the components.
- **jButton1ActionPerformed(void) Function**: This is a function assigned to the JButton, assigned the x-position to both runners, creates two threads (Race_s class) and start them, it also play the song.

- **main(void) Function**: It calls the RaceFrame2 function and make it visible (the RaceFrame2 class JFrame)

Make nonsense to explain the getters defined in this class.

Image:

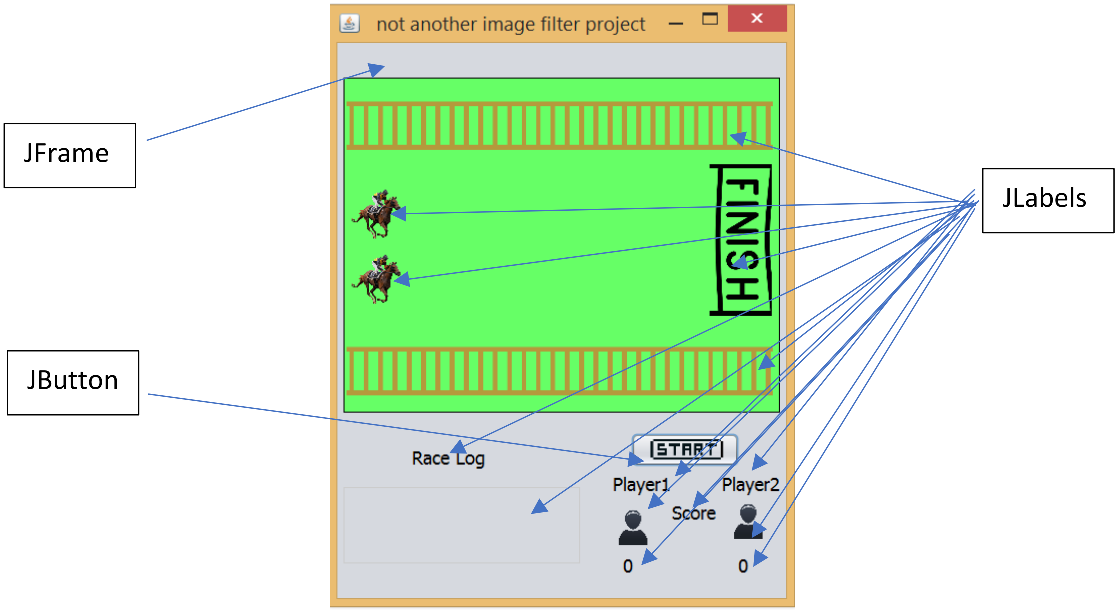
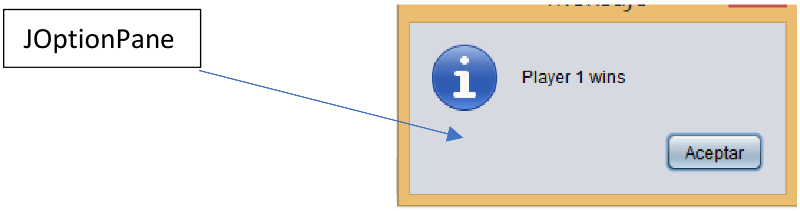


Image 2.0: RaceFrame2 JFrame



Player 1 wins

Image 2.1: JOptionPane

Race_s class:

Library or interface used**:**

- **Swing**: This is a graphical library, I used this library in order to create the JFrame that contains components that at the same time contains methods that make possible the transition between JFrames and also I used to show JOptionPane to show the result of the race.

- **Random**: Used to generate a stream of pseudorandom numbers.
- **Lang**: Provides classes that are fundamental to the design of the Java programming language such as String, Math, and basic runtime support for threads and processes.

Components used:

- **JOptionPane**: This is the only graphical component used in this class, it is used to show the result of the game and make the transition between the RaceFrame2 JFrame to End JFrame.

Function used:

First I declared the JButton and JLabel variables.

- **Run(void) Function**: Here is where I defined a while(true) statement, inside the while I generate two pseudo-random numbers (between 1 and 10) with the function Math.random, one of those random numbers it is used to sleep the threads and the other it is used to know in which if statement is going to enter.

On each while cycle it and check if the finish line minus ten units is overstep by any runner, this with the help of the get_Final_Line () and getLocation (). x method to know the x-position of the JLabel used to simulate a finish line.

Then it checks if the thread is going to modify the x-position of the JLabel that simulate the runner (backwards or forward) or just sleep more milliseconds and then it call the repaint() method that it is used for update all the components modified that have been modified.

Next, it checks if the x-position each JLabel that simulates the runner is the same as the x-position of the JLabel that simulates the race finish line and depending on how many players (if both) have crossed the finish line or if any one specifically modifies the value of a JLabel that simulates a log race where it tells us the result of the round

The last part is a verification to know if a player has 2 points or if they both have 2 points to be able to show a JOptionPane that will let us know the winner of the game and take us to the final Jframe.

- **setNumber_r(void) Function**: Updates the number of rounds.

End class:

Library or interface used:

- **Swing**: This is a graphical library, I used this library in order to create the JFrame that contains components that at the same time contains methods that make possible the transition between JFrames.

Components used:

- **JLabel**: I used 1 JLabel and I modified the text property in order to not show text and added the gameover.gif to the icon property.

Functions used:

First I declared the JButton and JLabel variables.

- **End(void) Function**: It calls the initComponents function.
- **initComponents(void) Function**: Creation and definition of the JLabel, more specifically is where I defined the properties and layout of the components.

- **main(void) Function**: It calls the End function and make it visible (the End class JFrame)

Image:

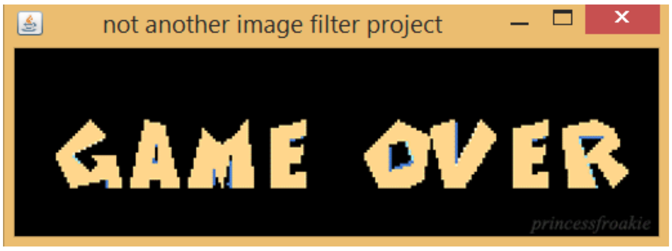


Image 3.0: End JFrame

Setup Instructions

The code can be found at the following link:

https://github.com/JuanPabloAboytes/Final_Project_HorseRace

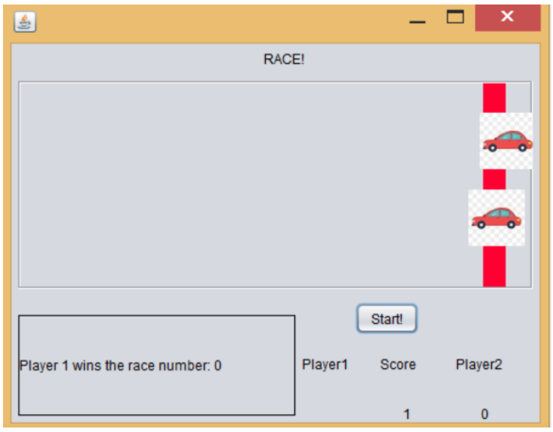
I reccomend to use NetBeans to run the game by click the “race” package and then click the “Run Project” button

Results and evidence.

First of all I consider it a mistake to use the JLabel to manage the graphic part of the game, since it limited the scope of the project a bit, the way java behaves with the change of property values such as text and icon, but Not all of them are disadvantages, using JLabels for the graphic part allowed me to save time in the definition of methods for managing the position of the components (using the getLocation () and setLocation () method).

Perhaps I can consider an error the topic chosen for the game, since a simulation game requires a much more complex structure than the one presented.

Horse Racing Simulation was tested a couple of times for several days, each time improving the game based on the user's opinions. Some of the opinions and comments said for those users will be written below.



Player 1 wins the race number: 0

Image 4.0: First Version of Horse Racing Simulation

Omar de Alba tested the first version of the game (image 4.0), at the beginning I was thinking of being a car race simulator but Omar recommended several things.

1.- A better aspect of the game (use pngs).

2.- The topic of a horse race simulation.

3.- Insert background music.

Romel tested the last version of the project said that it could be great add random graphic elements that modify the course of the race in a more visual way.

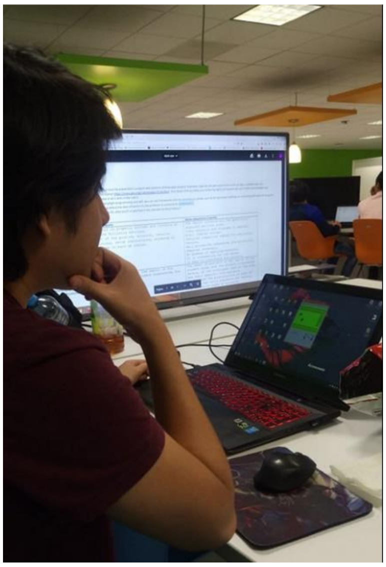


Image 4.1: Romel Martinez

Alfonso agreed with Romel about adding random graphic elements and he also said that he do not like simulation games.

Image 4.1: Alfonso Diaz

Author comment: Sorry about the JFrames titles, hopefully no one gets mad about that.