Eli Zucker

Mr. Kuszmaul

AP Computer Science

17 October 2015

Hivolts Readme

Introduction

Hivolts is a classic game in which a player moves and tries avoid mhos while slowly killing them off. The player has ten possible moves, the 8 eight squares around him, to sit, or to jump. Jumping will land the player in a random square guaranteed to not be a fence. If they avoid the remaining mhos, it will continue being their turn.

By myself, have coded a recreation of this game in Java, that fulfills the requirements. The game is fully functional and I personally think it can be entertaining.The player is a green circle, while the mhos are yellow. I went for this look because I thought it looked very simple, and similar to “agar.io”, an extemely popular web game.

Specification

My program fulfills the specification in almost all ways. The logic behind the mhos’ moves matches the schoology description perfectly. Not only is the logic behind the program solid, but animations are utilized to make the user experience as enjoyable as possible. Also, the program window can be resized as needed, while maintaining the same aspect ratio. Something I think is unique to my program is that a user can press a key to move, then resize his window. The board will actually continue animating while the window is resizing. Furthermore, my game has a restart button that can be used when they win or lose.

Something that I did not fully complete was signify that it was the user's turn. I could have implemented a scoreboard easily enough, but I thought that the simplicity of the program would be sacrificed if I added another scoreboard. In my opinion, animating signifies that it is not the user's term, while not animating signifies that it is the user's turn.

Current Errors

Currently the only error that I have found is that when the game ends the user is met with a white screen behind the pop-up dialog. I believe that this has something to do with me halting the paint method when the dialog is up, because the paint method causes some strange errors when a JOptionDialog is being used. In order to fix this, I would probably have to either remove the JOptionDialog as a whole, or make my own class extending JOptionDialog and modify it myself. Currently, my solution now makes the program still usable but not completely satisfying.

Code Overview

My code starts in the main class. There, a JFrame is created and a board (JPanel) is added. The board controls displaying every game element. In order to do this, it utilizes Fence, Character, Mho, and Player. Mho and Player extend Player, and they all extend the Unit class. The Unit class has generic code to handle all animations,. Each subclass of unit utilize this information, and each have their own unique paint methods, called in the Gameboard. Although Gameboard does all the graphics, it doesn’t actually do all of the processing for the game. For this reason, a GameProcessor object is made. This object will keep a map of every location (2 dimensional array), and process where mhos and players will move. It also determines whether or not the player wins or loses. An important aspect of my program is animations. I do this by having an infinite loop in the Main class that regularly calls the paint method while increasing the frame. Each element will change its image slightly based on the current frame, resulting in an animation overall. Finally, I also have death/win message code in Main, because it does not fit in Gameboard. The message is in a separate window and is handled separately.

Challenges

By far the hardest part of my code was the animations. It took a lot of time to figure out the idea of constantly looping, and increasing the frame. Once I had the concept figured out, I spent some time modifying code and in the end it worked well. The second hardest challenge was calculating mho movement. I had to think hard, but in the end I managed to get movement working by utilizing several recursive methods in order to move the mhos as they should.

Acknowledgements

I did not do this project completely alone. I would like to thank Devin Ardeshna, for peer-editing my code several times when I was having frustrating errors. I would also like to thank Tyler Packard for giving me the idea of animating the mho and player movement.

Proof of functioning program:

https://i.gyazo.com/6f40186fa9923c505e7d3931219b8910.gif