ECE Capstone Project
Real Life JRPG

Jacob
jacob.wilson6997@gmail.com

4/15/2021

Table of Contents

1. Introduction

Real Life JRPG is a game that involves taking the battle system found in JRPGs and bringing it into real life. It is a turn based system where one or more players battle a Dragon. players will select actions they wish to perform using a controller, called a sword, then will physically swing the sword to confirm their decision. After which the Dragon will attack, and they will be able to select another action.

Both the players and the Dragon have Health points, called HP, once a player loses all their HP they can't perform any more actions, if all players HP reaches 0 they lose, while if the Dragon loses all it's HP the players win.

The game uses up to three controllers wirelessly connected to each other.

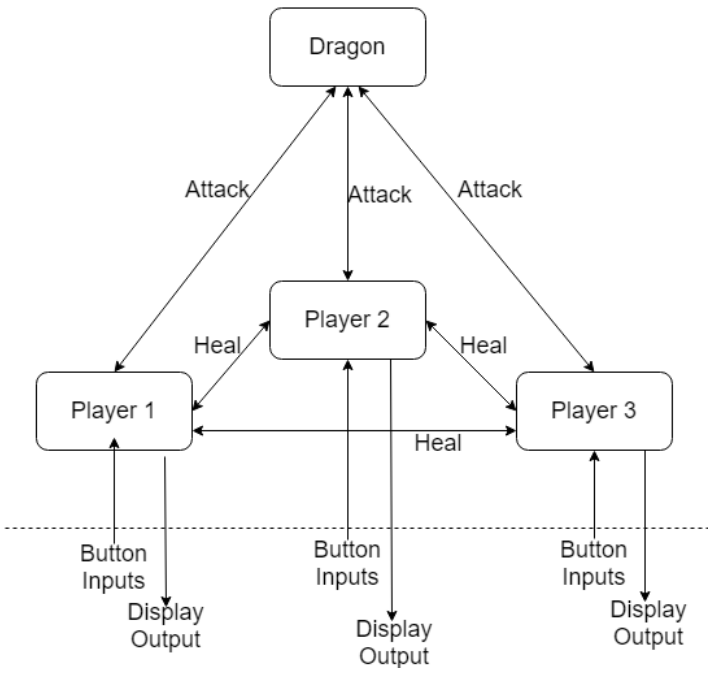
Each controller has a screen to display a menu, have a way to navigate the menu, and be able to detect motion.

The game is a turn based battle system where the players have 2 actions they can perform, Attack and Heal. Attack will reduce the HP of the Dragon by a set amount, while Potion will allow the player to increase the HP of another player. Each player can only heal once.

The game is meant to emphasize teamwork.

2. Methods

This project requires several devices to communicate wirelessly. Due to previous knowledge in the subject, it was decided to do this by the dragon device creating a wireless network that the sword devices will connect to. Information would be communicated through WebSockets.



The diagram to the left is the system diagram. The Dragon receives and sends attacks and the players receive and send heals from each other, the inputs are buttons and the outputs are the display.

In order to make things compact the swords were to be powered by a Raspberry Pi Zero W, while the dragon, which could be much bigger, was to be powered by a Raspberry Pi 3

Each Sword would need a screen for choosing an action, 3 buttons to navigate the actions, and an accelerometer to detect the swinging of the sword. They would all be connected to the Raspberry Pi Zero W's pins.

Due to the familiarity of the language, Nodejs was chosen to run on the devices.

In order for the turn based system to function, the dragon would wait until all players had sent an message, this message would contain whether it is an attack or a heal, as well as many other pieces of information required for the program to function, it would then reply to the swords to let the players choose a next attack. The reply would sometimes contain an attack from the dragon, depending on what attack the dragon uses.

After sending an attack, a player will be unable to make any more actions until the dragon sends a reply.

A player will only know their own status, communication is required to know if another player is low HP. In order to find the best strategy to defeat the Dragon players must work together.

3. Results

Determining that my objectives were met was done by looking at the screen as you play.

The game doesn't start until all players have pressed the ready button, therefore I know that all players are connected.

The inputs all work, a menu displays on the screen and when I press a button it is registered, and each button alters the data being sent. Finally when the screen asks for you to swing the sword it will not continue until you do, so I know it can detect motion.

I know I can attack the Dragon because I've won, meaning it's health reached zero, I also used command line outputs which displayed his health each turn to double check, and I know I can heal by looking at the health bar of my ally and noticing that it rose, and also from command line outputs.

The fact that my ally's health bar isn't on my screen is what I'm looking for when it comes to teamwork. Communication is needed for efficiency.

4. Discussion

In this project I overestimated the power of Nodejs on the Raspberry Pi Zero W, although in the past it was unable to run certain things, because I worked with an Arduino with a screen in the past, I assume I could do the same with nodejs on the Raspberry pi Zero W.

But The Raspberry Pi Zero W can't control a screen or accelerometer with nodejs, and I can only assume it's the fault of the language considering it works just fine with Python. In order to avoid

rewriting the whole thing in Python I found a way to have Nodejs send data to a python program and back, it slowed down the process but it still worked.

5. Conclusion

Although in the end the project met all requirements, it isn't without its flaws. This project isn't sellable by any means, it's far too expensive and slow.

To fix this problem the whole thing would have to be redone from the ground up. I would reprogram the whole thing in python, and bluetooth should be used rather than WebSockets. This will not only increase the speed, but also increase the reliability of the connection.

I am also not a game designer, as such the fight is unbalanced. The battle system is also very simplistic, you can only implement very basic strategies. This isn't helped by the lack of visuals on the dragon's part, which holds the players back from having an engaging experience.

However, despite how much it can be improved, I still succeeded in what I set out to do, all features I wanted were implemented. In doing so my knowledge of Nodejs, Raspberry Pis, and GPIO pins increased substantially, and I plan on pursuing a career in that subject.