CSE104 Project Report

Minh Hung Tran, Hoang Nguyen Vu

May 8, 2023

1 Background

Static website is boring. That's what we're thinking about. A good animation and some interactions between the user and the website are not enough to fulfill our dream of making something truly interactive, or at least enjoyable to code. A blog is boring. A store that sells food and clothes is boring. So we chose to do something else, something that we can both code and enjoy our product at the same time. A game was our perfect choice at that moment.

But a game like Tetris or shooting tanks is too common, and the rule is fairly simple. We want something to be logic, something that requires calculations and strategies (actually the result is not that close to our target, but at least it is somehow related to strategy games). And hence, we would love to introduce our project: WeirdRPG.

The game is based on a genre that everybody may recognize. This game includes you, a player, with some enemies for you to fight (some actions, at last!). You can pick up items to strengthen yourself, grab some swords and shields, carry a heavy armor and fight the enemies. You can level up yourself, moving around with further range, find stronger enemies and fight them (again). It is pretty much it, but be careful. Once you are in a battle, you can only move in the enemies' range. You may have to fight multiple enemies at the same time. A trick (if you may not know) if you want to grab a sword really far away but your movement is limited around your enemy range, you can move around, waste your turn, and wait for another enemy to catch up to you. You can jump to the new enemy's range, and you can skip the first battle and move to the other one, and also extend your movement to the new enemy's range. You may waste some HP, so think about your next move precisely.

2 How to play

Use Left - Right - Up - Down arrow to move the cursor. Enter on blank space to move. Enter on enemies and items to attack or interact. Defeat all enemies in the room to move to the next level.

3 Technologies and methods

3.1 Frameworks

As the project was meant to be done with only HTML, CSS, and JavaScript, it was nearly impossible to achieve our goal without the help of an external framework.

The only framework that we use in this project is **PixiJS**, which is a fast, lightweight 2D library that works across all devices. It utilizes the power of hardware acceleration without any prior knowledge of WebGL, which guarantees speed and limitless support (or at least that is what they said on the GitHub page). Thanks to the speed and stability provided by the framework, we can easily skip the setup step and instead, we just grabbed its module JS file and imported it into our code.

3.2 Project structure

JavaScript is truly the sponsor of this project. As we are more familiar with ES modules than the traditional CommonJS, we specify the imported script type with `type="module"`. This helps a lot in structuring the files, as the importing is nearly painless and much more practical than the `require` function for CommonJS.

Apart from the `index.html` and `style.css` files that are put at the front of the project, along with 3 essential folders: `include`, `sprites`, and `src`. `include` contains `pixi.mjs` for PixiJS, `pixi.mjs.map` for PixiJS map, and `fontfaceobserver.js` as font sometimes cannot be loaded properly, so we use `FontFaceObserver` to wait for the

font to be fully loaded before running the game instance. `sprites` contains all sprites in the form of a tilesheet. The sprites are splitted and picked manually, but this is pretty simple as all sprites are 64 x 64 pixels.

The folder `src` contains everything we need to run the game. We have `containers` folder which contains the containers arrangement for the game. The `objects` folders contains the objects, from player to enemies, backgrounds and all other useless things that appear in the game. The `utils` folder contains essential utilities like keyboard receiver and array comparison (we all know that if we compare two idential arrays in two different constants, the return value is always false, and we want this to be done the right way). Other files are `constant.js` for constants management, `loop.js` for loop handler, `sprite_loader.js` to load sprites, and `main.js` to import everything and run the game.

3.3 Logic behind the code

The player can be moved using Left - Right - Up - Down keys on the keyboard, which can be checked using the keyboard utility that we manually created. The game is grid-based, which means that everything has its own cell, and they are stored in a 2D map inside the container.

The fun part is the logic in the enemy movement. We didn't use the naive method and gave them premade movement paths. Instead, we used Breadth first search algorithm to find the shortest path on the containers grid, and based on it to find the most optimal way for the enemy to move to the player. It might sound like something really easy to do, but we have to check tons of conditions, which is costly if there are too many enemies on the map. And to be honest, it would be a waste of time if we try to figure out if there are any easier or faster way to find the shortest path, so we just reduce the enemies' eyesight to somewhat reasonable level so that they cannot move if the enemy is not in their view. Sounds lazy, but it is practical, and it saved us a lot of time.

Although everything seems to be hardcoded and can be optimized, we chose to value our time and try to finish every essential parts. Animation is one thing that we skipped as it could take more time than we expected.

4 Result

Our game is, to be honest, not very perfect. It can be polished and improved, but due to time shortage, we had to leave other ideas in the back and submitted this version. Hope that you can enjoy this small game from us, and thank you for your teaching so far.