Every Man for Himself Technical Guide

Introduction:

Every Man for Himself (EMH)" is a web-based multiplayer game where a number of players are pitted against each other in order to escape a dungeon. One player is the "killer" and his objective is to eliminate the other players who are "prisoners." The prisoners are trying to escape through a procedurally generated dungeon. The killer is attempting to eliminate the prisoners via the use of "traps" that he can place in the various dungeon rooms. The killer will be able to place different kinds of traps in each of the generated dungeon rooms. Every turn, all the prisoners must select a room to enter, and the killer will place a number of traps. If the chosen room is trapped, that prisoner will be killed.

The original developer team consisted of Corey Burns, Vivek Kunnath, Ryan Mackenzie, Adeel Minhas, and Brandon Thorne. The github page for this project can be found at the URL: https://github.com/vkunnath/WebSciGroup6. All developers are welcome to use this project for educational purposes, like for creating their own multiplayer game in NodeJS, using socket.io, or any other technology that was used. The project is licenced under the MIT licence.

If you would like to contribute to this project, please contact the project contact at bthorne3@gmail.com.

Technologies:

NodeJS v6.9.1
AngularJS 1.6.2
Bootstrap 3.3.7
JQuery 3.1.1
MongoDB 3.4.2
Socket.io 1.7.3
Express 4.10.2

Installation:

The site can be accessed by navigating to port 3333 at IP 45.55.197.195. However, a user can also install the application into their own machine to host local games.

To install and run the application, you must have NodeJS, Node Package manager (NPM), and MongoDB installed on your machine. Please follow the guides found here if not already installed:

https://docs.npmjs.com/getting-started/installing-node
https://docs.mongodb.com/manual/installation/

After this is done, follow these steps to install and run the application:

1. Navigate to the folder where you saved the EMFH game files via command line.
2. Run ‘npm install’ to install the proper node modules
3. Start and run MongoDB by:
    a. Navigate to the bin folder in two separate command prompts
    b. Run ‘mongod’ in one command prompt and ‘mongo.exe’ in the other prompt
    c. If this does not work, please consult the MongoDB documentation on how to start it for your operating system: https://docs.mongodb.com/manual/
4. Run ‘npm start’ to start the application
5. Now the application should be running on localhost:3333

File Structure:

Package.json - Needed for NPM. Specifies all of the node modules needed for the project to be downloaded in npm install

Node_modules Folder - contains the node modules needed for this project. Do not edit this folder directly

Public - Contains all of the HTML, CSS, fonts, images and Angularjs/javascript. This entire folder is served by express on port 3333

- CSS - css files for styling of the various views on the page
- Font - downloaded font data used for the logo/general text
- Images - all of the image assets used for the site including backgrounds, the doors, and sprites representing the developers on the about page.
- Js - All the javascript files
    - MainController.js - The angular controller containing all of the logic for the application. Makes/receives socketio messages from the server to implement the game login
    - Gamepage.js - Javascript needed for animations of the door opening/closing
- Index.html - HTML for the single page application. Separated into several different views that is displayed at different times to the user

Write-up/Final Thoughts:

The team enjoyed developing this project and learned many lessons about game design, and web development. For one, game design is hard. Coming up with a fun/interesting game and all the logic is hard by itself, but becomes much more challenging when you need to consider the ways that it needs to be developed, and what can be done in the scheduled time. Originally, the design of the game was going to be a much more elaborate dungeon crawler, however it was scoped down in the proposal to use less assets (no artist on the team), and less complicated to ensure that the entire project could be completed and polished by the end of the semester. The team also

gained valuable experience with client/server relationships, like what data should be shared and sent back between the client and the server, what information should be stored in a client “host” for the session. We also learned the presentation is a very important part of a game, so extra development time was alloted to make the front-end cohesive, stylish, and branded.

As for web development, we learned the importance to stick to a schedule and to follow the chosen methodology closely. At the beginning to the middle of the semester, the team didn’t follow the schedule or methodology well, and quickly the project got backed up. Toward the middle of the semester, we realized our mistakes and began to correct it by dedicating extra development time at the end of the semester, rescheduling, and delegating tasks to particular members. Luckily we were able to fix this problem in enough time to complete the project, but it is a valuable lesson to not ignore a schedule in the future.

In future iterations, we would like to add additional features to this project including items for trappers and prisoners to use, humorous (or scary) game over screens, animations for the end of a round and for displaying the users who died. Also, we would like to add a way to monitor inappropriate names being added to the database and protections to prevent people from cheating by rewriting client side code to sniff/spoof the socket.io messages.

In conclusion, this was a fun project to work on and the team learned much about the basics of game development, and software development in general. It served as a very good educational tool for us, and we hope that it can also serve as an educational tool for other developers and that people can enjoy the game in their free time.