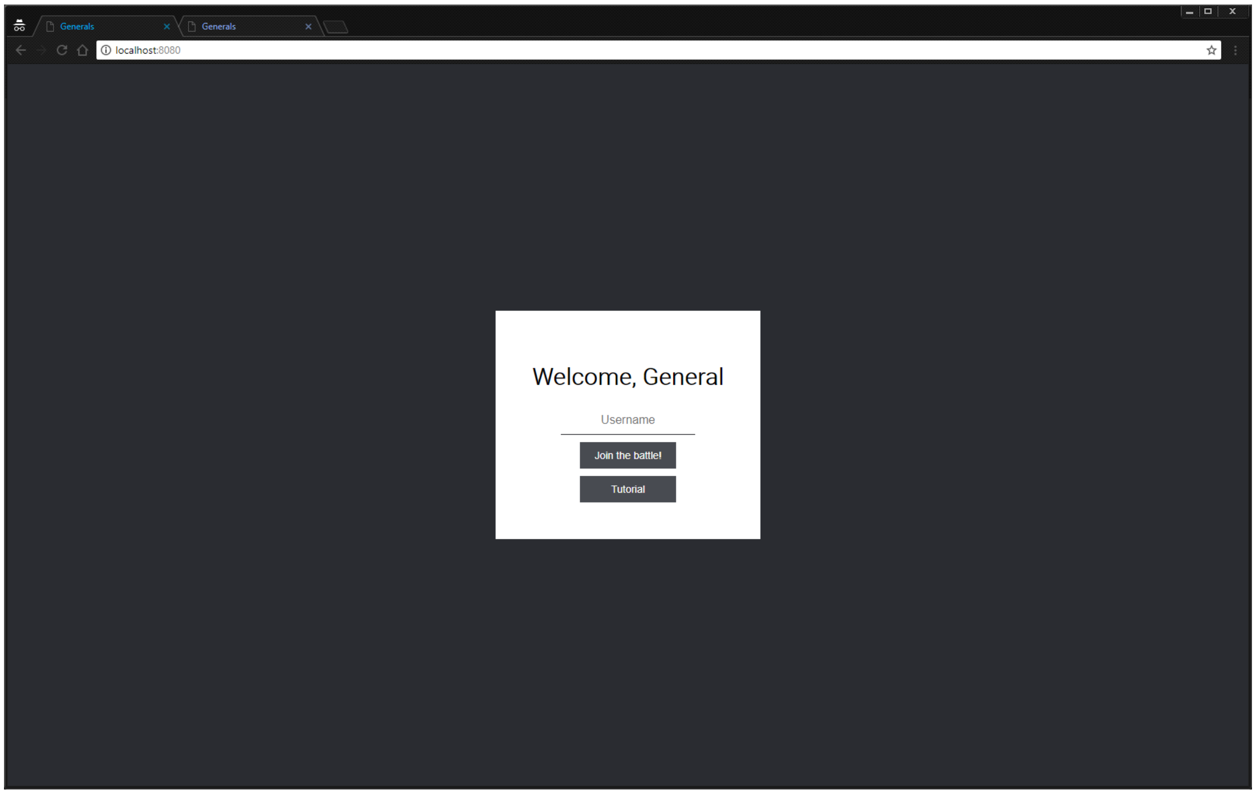
Project – Generals Che Jung Lee | 301249826



Overview

Generals is a multiplayer turn-based tactical board game written in Node.js platform. It is a cooperative game involving competitions between two opposing teams. Players are expected to communicate with teammates and form strategies to win the game.

Design

Generals is composed of two main elements: chat and game. Chat has global and team rooms to switch between, and players can communicate with teammates or all clients connected to the server at anytime.

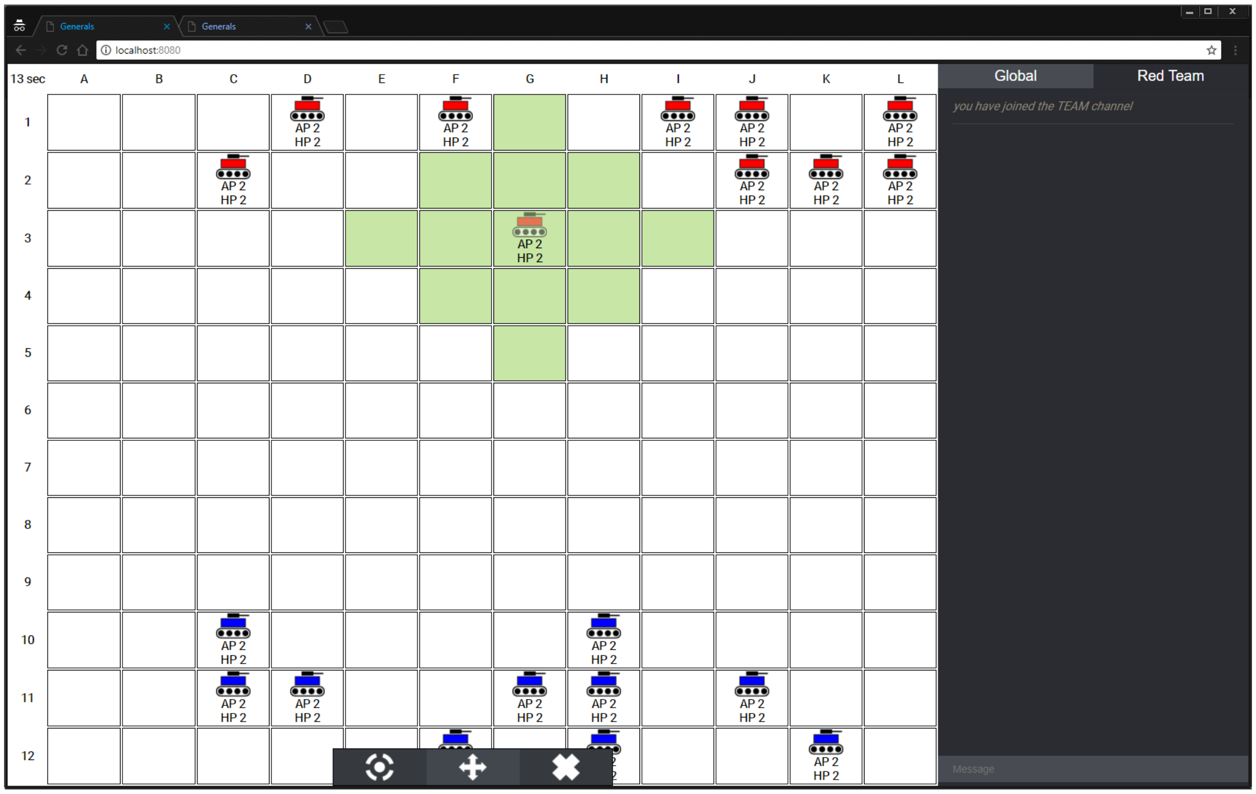
Each team has ten units, all of which perform two actions: attack and move. One action costs 1 AP, and each unit has 2 AP per turn. Therefore, one unit can either attack twice, move twice, or do a combination of both in one turn. All team members can control any of their team's units except the ones currently selected by other teammates.

Generals has two game phases in one turn: planning phase and execution phase. In the planning phase, both teams can control their own units to perform actions in which only the teammates can see. In the execution phase, players are not allowed to anything, and all actions done during the planning phase are revealed to both teams, including the health points and the movements of all units.

Due to this unique turn mechanism, where you can only see your teammates' actions during the planning phase, some certain situations may appear, so the following are the rules for those special cases:

- In the planning phase, if you attack a location occupied by an enemy unit, and that unit moves, the attack will miss. Likewise, if you attack an empty location, and an enemy unit moves to that location, the attack will hit.

- In the planning phase, if you move a unit to a location, and an enemy unit also moves to that location, both units die.



you have joined the TEAM channel

Implementation

Front-end

- HTML, CSS, and JavaScript
- Nginx

The front-end of Generals is a single web page built with HTML, CSS, and JavaScript. It performs basic validations such as user information and game status, but for the most part, it is receiving information from the server to display user interface, chat messages, and game visuals. We use Nginx to deploy our application and serve static content.

Back-end

- Node.js that uses Express.js and Socket.IO
- PostgreSQL database

The back-end of Generals is built in Node.js platform that uses Express.js framework and Socket.IO library. This combination allows real-time, asynchronous communication between the server and clients, which is essential for this project as we are constantly updating the chat and game status. The communication is done by creating and parsing JSON objects. PostgreSQL is used to record game stats for end game evaluation and status, though it is not necessary in this project; we did it merely for the requirement. All click events on the game board are sent to the back-end to process and validate. After evaluation, relevant information gets sent back to the client side to handle and display.

Features

- Nginx front end server
- Generator for units
- Turn cycle
- Game completion
- Game restarting
- Database for tracking gameplay stats
- Chat
- Chat room
- Movement
- Attack
- Action validation
- Unit interaction

Distribution

Ankit Dassor

- Project poster
- Unit sprites
- HTML and CSS for game board and general user interface
- Some Vagrant and Nginx configuration

Rohm Laxton

- Game design and project proposal
- JavaScript (both server and client sides) for game turn/start/end mechanism
- Some Vagrant and Nginx configuration

Nicole Thomas

- PostgreSQL implementation
- Some Vagrant and Nginx configuration

Che Jung Lee (Me)

- Everything about chat (HTML, CSS, JavaScript – both server and client sides)
- JavaScript (both server and client sides) for unit action and interaction

Alternative

There are too many multiplayer board games similar to Generals, so I only compare it with the games created using the same technology.

Unlike other Node.js board games that use modules, libraries or API to control game logic or display game visuals, such Phaser, Quintus, and WebGL, our game uses nothing but socket.IO, JavaScript, and HTML table element to handle game rules and game board. This is a unique feature that no other game implements, though this is quite challenging and inefficient. The table cell position is the only information available for calculation and validation, which can be insufficient for some special cases.

Conclusion

We originally planned to add different types of units and many small yet nice details to the game, such as listening to keypress events to control units. Unfortunately, we did not have enough time to do the additional features. This project is the base form of our plan, though I would personally love to do more than just meeting the project proposal.