Team 13 Final Design Document

Contact Info

Team Members:

- Haaris Chaudhry – hhchaudh@ku.edu
- Hunter Crisp – h714c637@ku.edu
- Navpreet Singh – navpreet.singh@ku.edu
- Nicholas Robless – n906r527@ku.edu
- Robert Cheruiyot – r855c264@ku.edu

Meeting Time: Monday, 1:00 PM

Contact: Haaris Chaudhry

Github Link: https://github.com/nrobless/team-13-project

Project Description

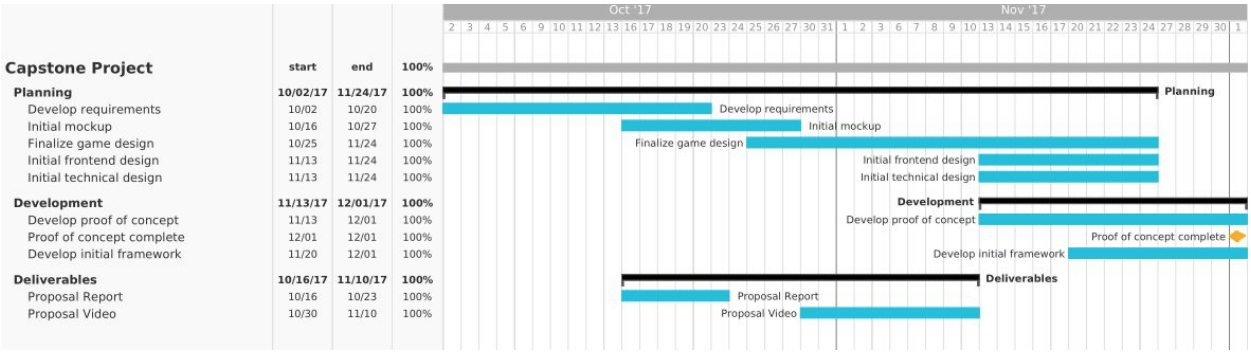
For our project, we will be implementing a real-time multiplayer game. Seeing the success of casual multiplayer web games inspired us to create something in the same vein, but with fresh new mechanics. Our goal for the game is to make it so that it is easy to learn, easy to master, and requires almost no commitment to be competitive; a game that a user can have fun with in the span of only a few minutes. The current idea for the game is a maze traversal game with a competitive multiplayer element.

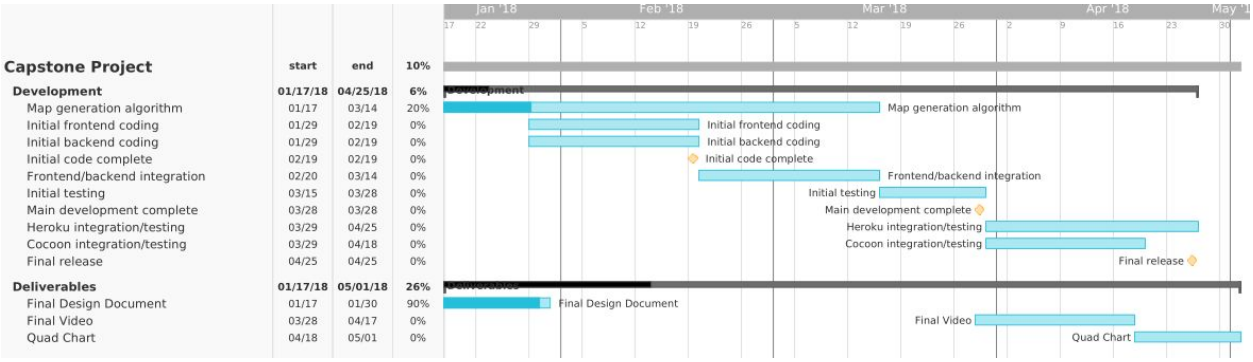
We want to undertake this project because we would like to familiarize ourselves with general application level networking concepts. The game will be using a minimal UI and Phaser on the frontend and the node.js language on the backend with web sockets. The concept of handling a real-time game with multiple players (perhaps over 100 players) in one server while maintaining low latency presents a challenge that we feel will better prepare us for future software engineering projects in the real world.

Project Milestones

1. Preliminary design completed – October
2. Establish basic development framework – October
3. Formal design completed – November
4. Proof of concept completed – December
5. Game logic implemented on backend – February
6. UI implemented on frontend – February
7. Frontend and backend integrated – January
8. Cocoon version for mobile devices – March
9. Deployed to Heroku – March
10. Testing complete – April

Gantt Chart





Project Budget

- HTML5 Canvas (https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API)
- Phaser (https://phaser.io)
- CocoonJS (https://cocoon.io)
- Node.js (https://nodejs.org/en/)
- Github (https://github.com/)
- Web Hosting
    - $5-30/month, depending on platform – For spring semester

Work Plan

We are planning to float between roles in addition to our primary roles. Our roles are subject to change based on proficiency and aptitude through the course of the project.

- Game Graphics and Interface – Robert Cheruiyot, Nicholas Robless
- Integration – Haaris Chaudhry, Robert Cheruiyot, Nicholas Robless
- Map Generation – Robert Cheruiyot, Navpreet Singh
- Backend/Game Logic – Haaris Chaudhry, Nicholas Robless
- Documentation – Nicholas Robless, Navpreet Singh

Final Project Design

This section walks through, step-by-step, how the client will interact with the game as well as what the server sees. Dot Bot, as we have called our maze-traversal game, primarily consists of four states: login, lobby, gameplay (divided into two phases), and game resolution. Below is an overview of these states, which will be explained in the following subsections.

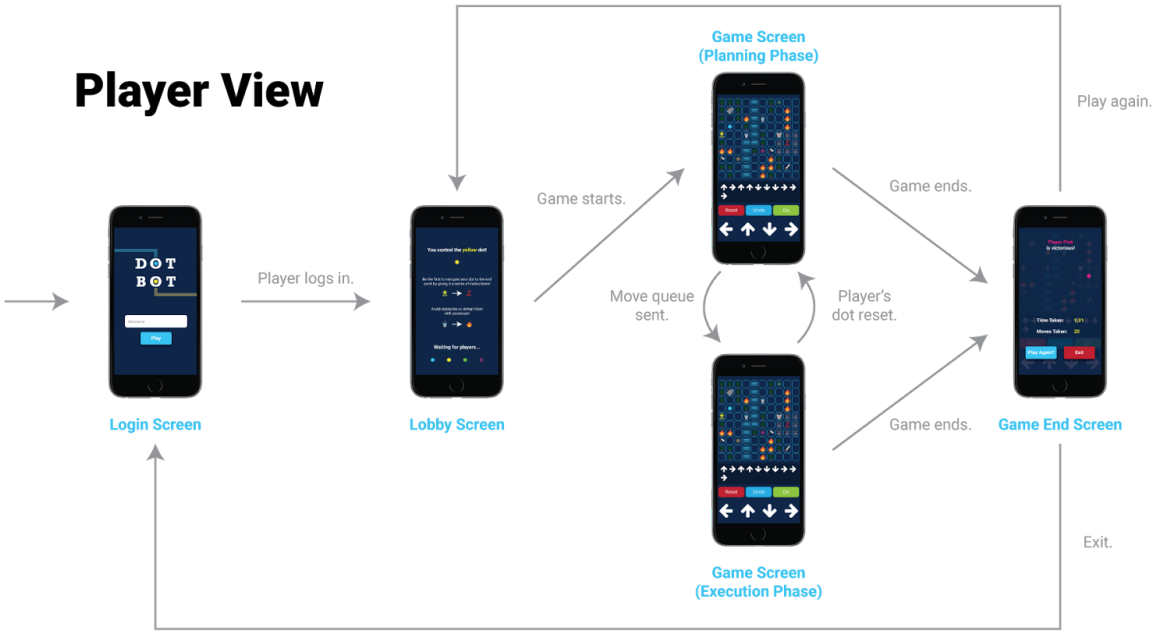


Fig. 1: Game states.

Game Setup

Players will first enter a nickname which will be used to temporarily track the user throughout the session and potentially be entered into a leaderboard down the road. Nicknames are not unique, and will not be saved permanently.

Afterwards, players will hit 'Play' and enter a game queue, or lobby, for the next game, which will take up to four players at a time and start a game process on the server when all four are present. There may be multiple lobbies on the server to accommodate more players and reduce wait times. Players can join the queue from the login screen or from the game end screen.

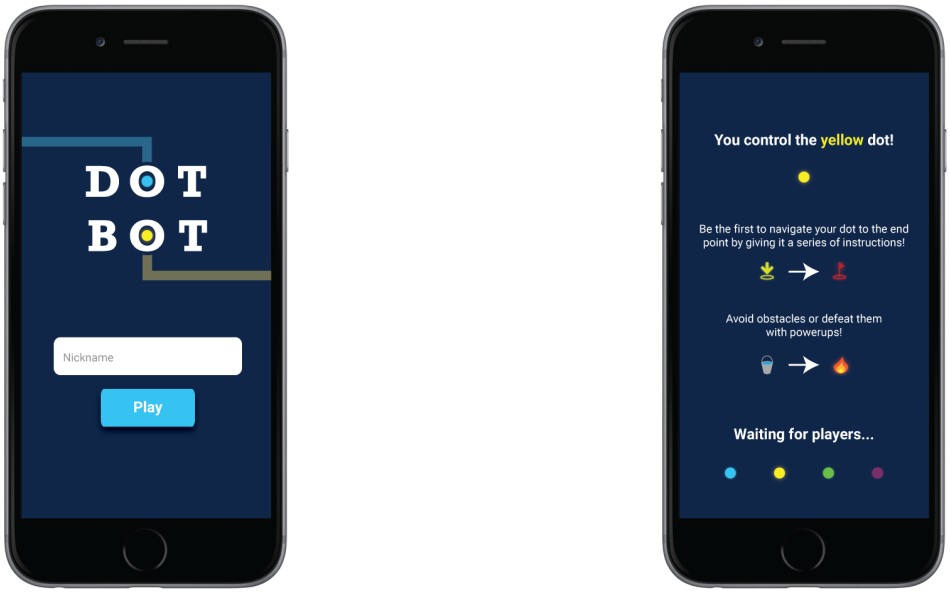
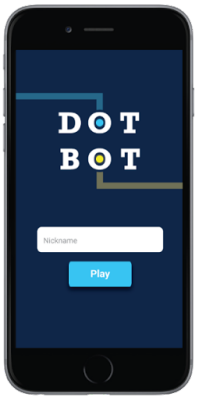


Fig. 2a: Login and lobby screens.

Client-Server Interactions

Login



Login Screen

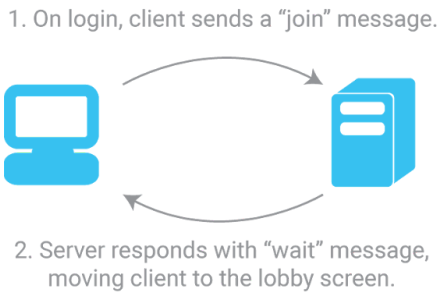


Fig. 2b: Login logic.

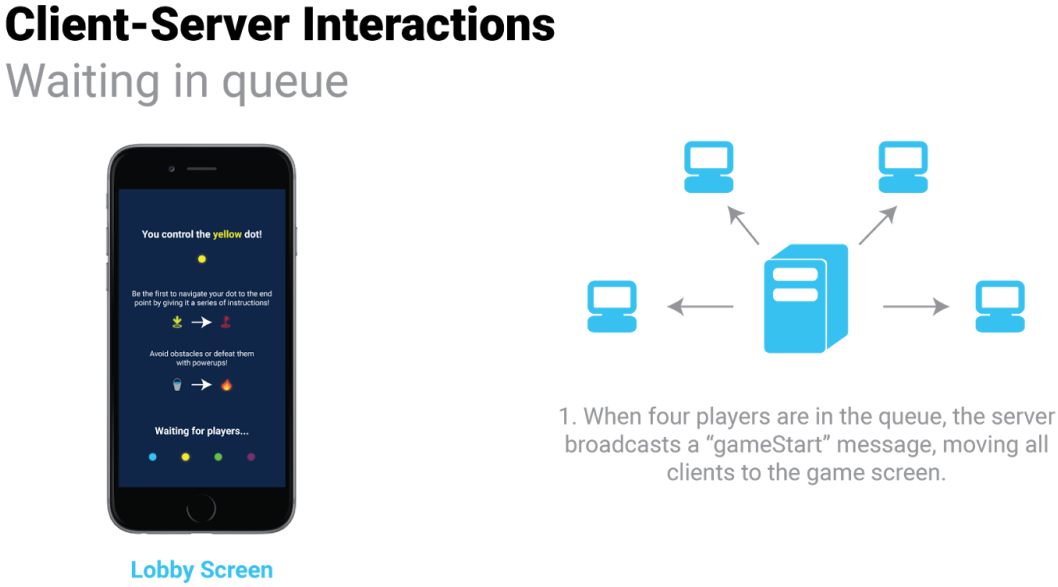


Fig. 2c: Lobby logic.

Gameplay

When a game process starts, players will be given a short countdown and then be moved to the game screen, shown below. The objective of the game is to proceed from a common start point to a common end point. **The first player to reach the end point without getting reset wins the game, which ends the game for all players and displays the game end screen.**

The game itself is divided into two phases: planning and execution. The planning phase consists of players placing moves in their individual move queues using the arrows and other buttons. The arrows place moves in the move queue for that player, which we may also implement as swiping once we have full native support. Reset will clear the move queue, and Undo will remove the last move entered in the queue. Go will send the move queue to the server and begin the execution phase.

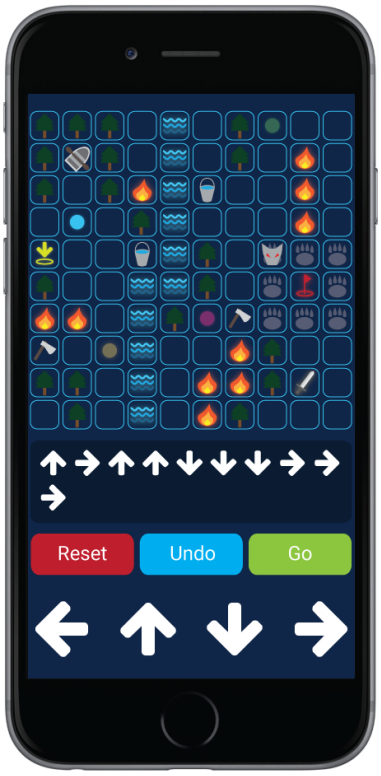
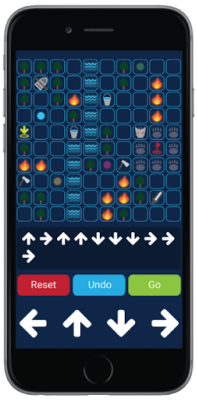


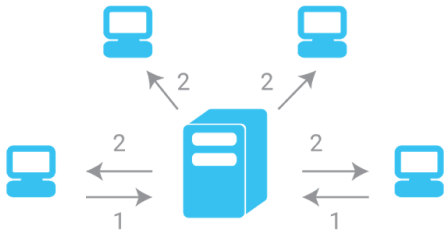
Fig. 3a: Game screen.

Client-Server Interactions

Gameplay



Game Screen



1. When an individual client presses Go, the client sends a “moveQueue” message to the server.

2. Whenever a move is executed in any move queue, which happens every 1 to 1.5 seconds in separate queues, the server broadcasts a “mapUpdate” message, telling clients to redraw the map.

Fig. 3b: Game logic.

Execution phase will show the player the progress of their dot, displaying the dot's position on the map as it executes each move at 1.5 second intervals. The interface will change as well, highlighting the next move to be executed in the move queue, disabling and greying out the controls, and showing what powerups are affecting the player's dot. **The player will not be able to interfere with the execution phase until the dot is either reset or reaches the end point.**

A dot can be reset by running out of moves when it has not reached the end point, running into the map edge, or by running into a hazard, both of which return the dot to the start point and reset the player to the planning phase. **Three types of obstacles can hinder the player: barriers, hazards, and monsters.** Barriers prevent movement into squares containing them, potentially wasting moves or forcing the player to choose a longer path. Hazards immediately reset the player's dot when moved into, and monsters are hazards which move along a fixed path at a specified time interval.

Each type of obstacle can be overcome with a different powerup, which is consumed on pickup for the rest of the current execution phase. When an obstacle is defeated, it disappears for the player until the current execution phase is over. **A powerup can be either used up upon defeating the first obstacle in the dot's path, or can be used to defeat an infinite number of obstacles, remaining in effect until the end of the current execution phase.** In the example above, the bucket and axe powerups are one-time use, while the boat and sword are persistent. If the execution phase finishes, all consumed powerups and defeated obstacles return to their original places. If the dot reaches the end point, the game end screen will be displayed, showing the player as the winner.

Throughout both phases, players will be able to see the positions of other players as transparent dots. **Players will not see the status of powerups and obstacles of other players, so no player can interfere with another player's execution phase.**

Game Resolution

After a game is finished, the gameplay screen will pause and become darkened, and the game end screen will be displayed over the top. The game end screen will show the winning player's nickname, and highlight their dot on the map. Other information, such as the length of the game and number of moves taken may be taken into consideration later. **A 'Play Again' button will also be displayed, and will place the player back into the game queue when tapped.**

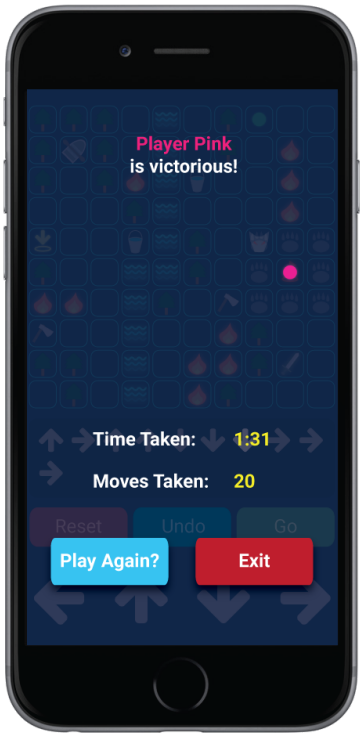
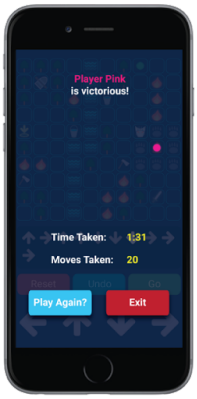


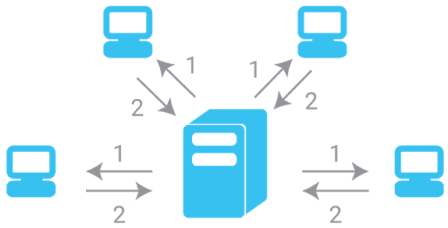
Fig. 4a: Game end screen.

Client-Server Interactions

Game Resolution



Game End Screen



1. When a player wins, the server broadcasts a “gameEnd” message, moving all clients to the game end screen.

2. On player acknowledgement, the clients individually send a “join” or “leave” message, being moved to Login or Waiting states, respectively. Timing out registers as a “leave”.

Fig. 4b: Game end logic.

Ethical and Intellectual Property Issues

Ethical

Our main ethical concern is developing a product that meets the requirements that we have set. We need to make sure that we deliver a viable product with no bugs or glitches with a user interface that is engaging. We also need to make sure that the product that we deliver is large enough in scale that it merits 6 hours' worth of upper level EECS classes. There is a vague line that separates a project that is trivial and a product that is non-trivial because the software engineering skills of each student may vary considerably. Considering the tools that we have selected to use, we feel that the project that we are undertaking is deserving of two semesters because of the amount of research that will be required to become proficient with these tools.

Intellectual Property

Clearly, we will need to make sure that we establish the authorship of all source code that we use. We have already established that we are using Node.js, Socket.io, CocoonJS, and Phaser. These pieces of software are all open source and can be used for both freeware and also for software that is intended for monetary benefit. Source code that is written by each of our team members will contain documentation that attributes the author. Using Github will help prevent the accidental change of authorship when updating code (i.e. change of authorship will require approval from all team members).

Art and sound for our game will need to be correctly attributed. We intend to use a lot of freely available sound (and perhaps art). We will strive to use art and sound assets that have the least stringent licensing requirements.

Change Log

1. Changed description to reflect current game idea. Instead of being a slither.io clone, the game is now a competitive maze traversal game (Oct 4, 2017).
2. Added new documentation and graphics to project design, revised Gantt chart and milestones to reflect current status (Jan 26, 2018).
3. Formatting changes, split Gantt chart into two charts for both semesters (Jan 29, 2018).