**Team 13 Initial Project Description**

**Team Members**

* Haaris Chaudhry – [hhchaudh@ku.edu](mailto:hhchaudh@ku.edu)
* Hunter Crisp – [h714c637@ku.edu](mailto:h714c637@ku.edu)
* Navpreet Singh – [navpreet.singh@ku.edu](mailto:navpreet.singh@ku.edu)
* Nicholas Robless – [n906r527@ku.edu](mailto:n906r527@ku.edu)
* Robert Cheruiyot – [r855c264@ku.edu](mailto:r855c264@ku.edu)

**Team Meeting Time:** Wednesdays 11:00 AM

**Lab Meeting Time:** Mondays 11:15 AM

**Contact:** Haaris Chaudhry

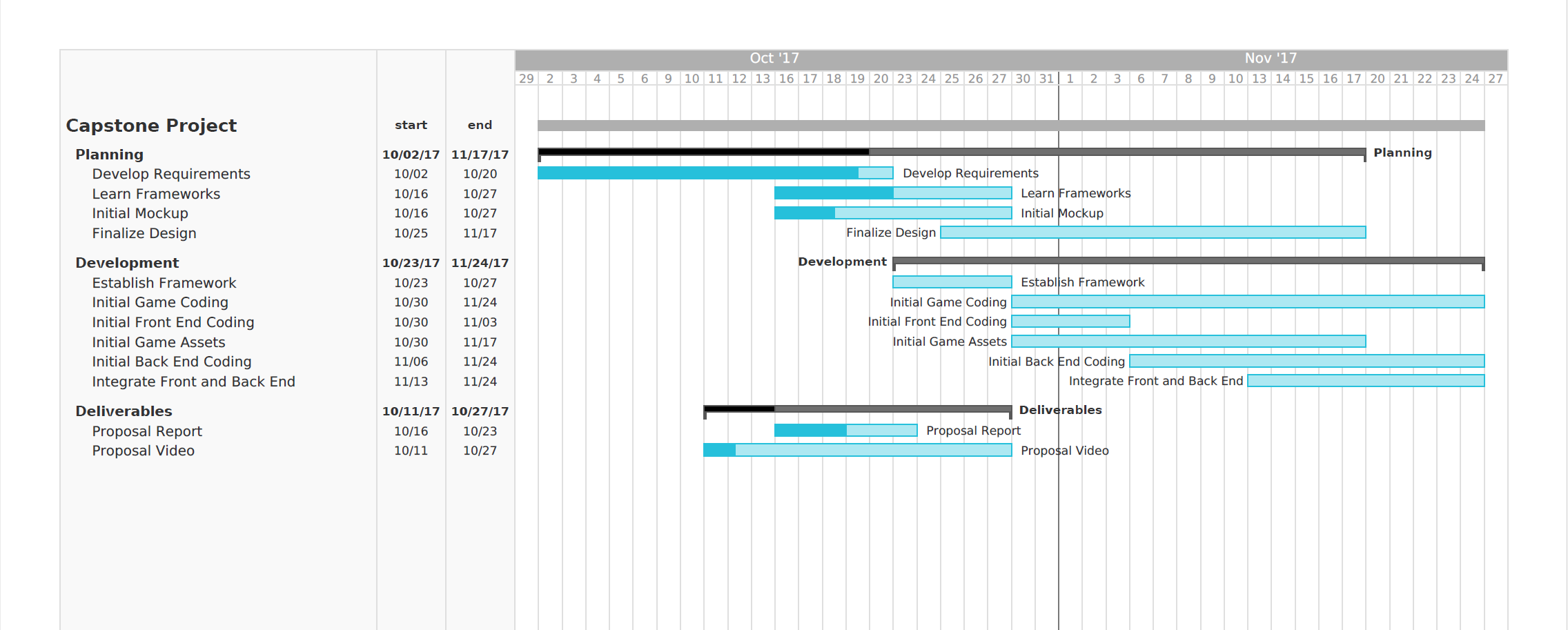
**Project Description**

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 Canvas 2D 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. Formalize a design for all components of the game – October
2. Establish basic development framework – October
3. Basic movement and controls developed – October
4. Establish connection between frontend and backend – November
5. Game logic implemented on backend – November
6. Two players able to join a game at once – December
7. Set up official game server and hosting – January
8. Create interface for many players to join seamlessly – February
9. Testing for scalability – March
10. Development documentation complete – April

Gantt Chart:

**Project Budget**

* HTML5 Canvas (<https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API>)
* Phaser ([https://phaser.io](https://phaser.io/))
* CocoonJS ([https://cocoon.io](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 – Hunter Crisp, Nicholas Robless
* Frontend Integration – Hunter Crisp, Robert Cheruiyot
* Backend Integration – Haaris Chaudhry, Robert Cheruiyot
* Game Logic – Navpreet Singh, Nicholas Robless
* Documentation – Haaris Chaudhry, Navpreet Singh

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

**Preliminary Project Design**

Our project is a maze-traversal game. We will have a log in screen where players will log in anonymously (i.e. they can create a username, but none of their information is saved). A game session requires 4 players to start, so once 4 players have logged in then they will be moved to a game session on the server. We will have a log in queue that will contain players waiting to join a game. Assuming our game creation service is fast enough (and that we have enough resources on the server to meet peak demand), then there should be no reason that this queue ever grows to more than 10 players. Our server will contain multiple game sessions with each containing 4 players. A game session on the server is simply a process where 4 players will continuously play against each other in one maze-traversal game after the other. If a player drops out, then the server will bring in another player who is currently waiting in the queue.

The maze traversal game itself is very simple. There is a start-point on the maze and an end-point.

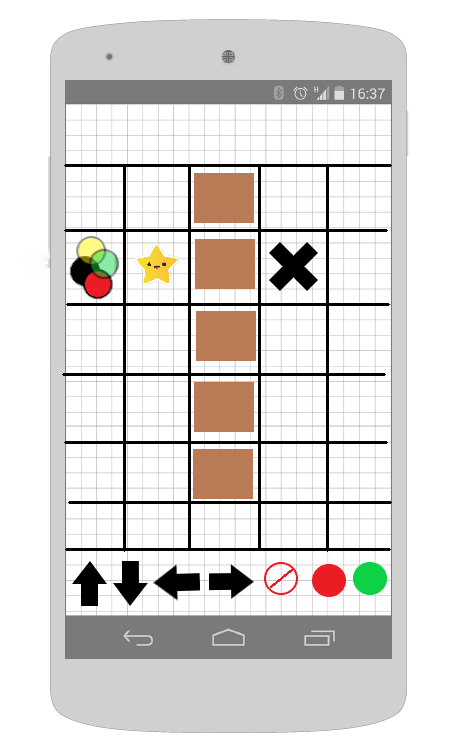


Figure 1

In *Figure 1*, we can see that the start-point is the cell that contains 4 discs: one yellow, one green, one black, and one red disc. The end-point is the cell that contains a large “X”. On the grid we can see brown blocks that represent barriers and a star that represents a power-up that can destroy barriers. The player’s goal is to queue up moves that will result in his disc getting to the “X” before the other players. The player will not be able to see, on the grid, where his queued-up moves will move his disc. However, there will be a section on top of the grid that will show queued-up moves. Let’s look at an example:

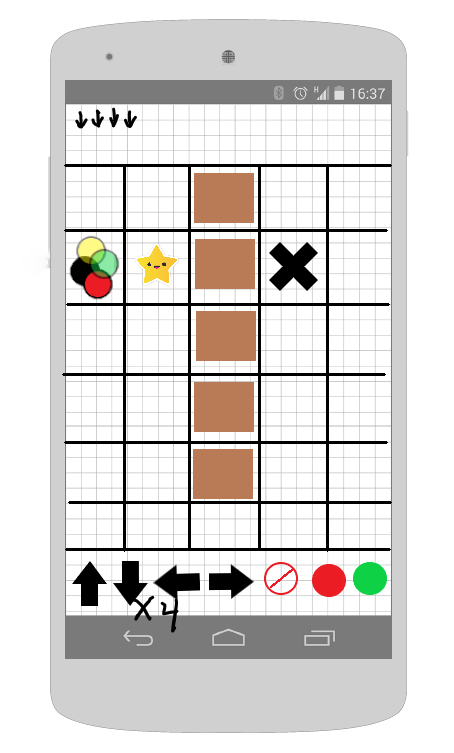


Figure 2

We can see in *Figure 2*, that the player has queued-up 4 “down” moves by tapping the down arrow button 4 times. The current move-queue at the top of the grid displays the moves that have been queued-up. If the player would like to remove the last queued move, then he will tap the cancel button. Tapping the all-cancel button will result in the removal of all moves in the move-queue. Tapping the start   button will move the player’s disc using the moves in the move-queue.

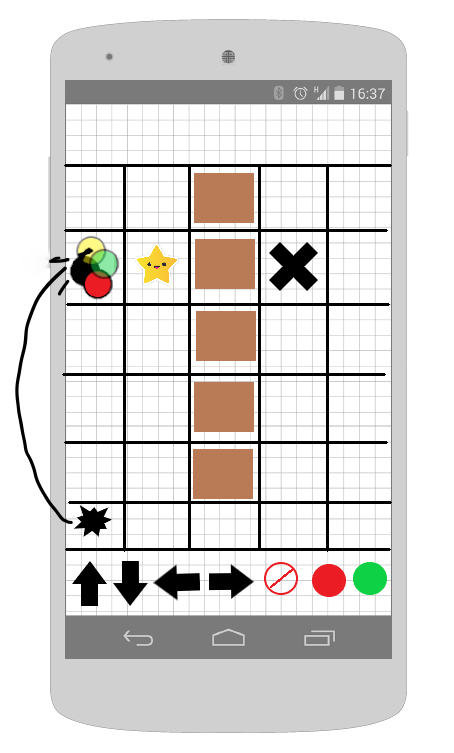
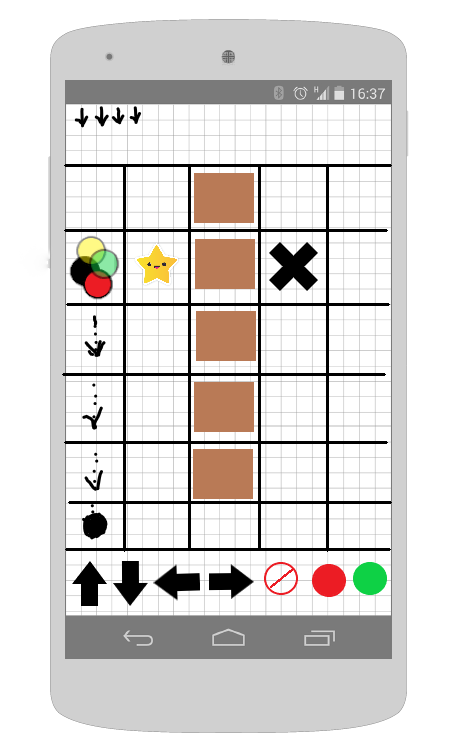
****

Figure 4

Figure 3

Let’s assume that the player controls the black disc. As shown in *Figure 3* when the player hits the start button, his disc will traverse the grid according to what is inside the move-queue. The drawn dotted arrows simply show where the player’s disc will have been on the grid during the move-queue traversal. As shown in *Figure 4* if the player gets to the end of the queue without having reached end-point on the maze (the “X”), then his disc will explode and will be reset to the start position. This is a critical aspect of the game, once the player hits start *his move queue must result in reaching the end-point or else he starts over*. Another way that a player may end up having to start over is if his move-queue results in attempting to move to a cell that is occupied by a barrier. A disc move animation will take approximately .5 seconds. While the playing the game, the player may also observe the discs of other players moving according to their respective move-queues.

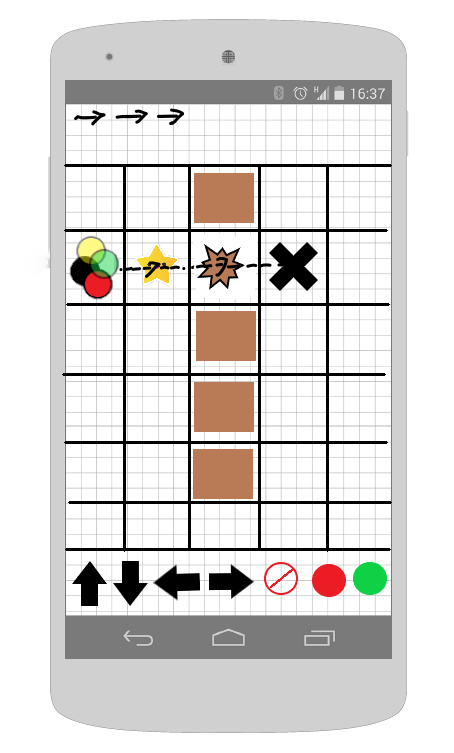


Figure 5

One of the strategic elements that we are going to add to our game is the use of power-ups. Power-ups will enable the player to overcome certain obstacles which may allow the player to reach the end-point of the maze using fewer moves. Let’s look at the simple example provided in *Figure 5.* Here, we can see what happens after the player hits the start button with 3 right arrows queued-up (right arrows indicate that the disc will move to the right). Once the player reaches the star, his disc will gain the ability to destroy barriers, therefore, the 2nd right arrow move in the move-queue will result in a valid move because the player’s disc is allowed to move to grid cells that are occupied by barriers. We can see that the minimum number of moves required to reach the end-point using a star is 3 while the minimum amount required while not using a star is 11.

The examples given here are very simple using a fairly small grid. Our end product will contain games with larger grids and more complex mazes. We would like to include different types of power-ups that can be used to overcome different types of hazards.

Our game will use a traditional client-server model that will simulate a real-time experience using web sockets. All of the game logic will be contained on the server and the client will simply be required to send moves and receive updates on the current status of the game. For each game session, the server will take any incoming move-queues for a player and use a timeout sequence of .5 seconds between each move before moving the player and then updating the new coordinates to all players in the game. The client is expected to perform the correct animations after each update from the server.

One of the things that we want to avoid are dead games. A dead game is a game that cannot be solved or one that is not actively trying to be solved. We will have to be careful when creating our mazes to prevent any unsolvable mazes from occurring. One of the extra features that we would like to implement, given enough time, would be to create a random map generator. However, randomly creating a map with various hazards and powerups that is both

a) non-trivial to solve AND

b) still solvable

is a task that may require the completion of an algorithms class. Therefore, for now we will be creating maps “by hand”. Another feature that we will implement is a timeout feature where a player that is inactive for a certain period is removed. Lastly, we’ll make sure each game must be solved in a specific amount of time so that the game is guaranteed to end.

**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. Project Description:
   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).