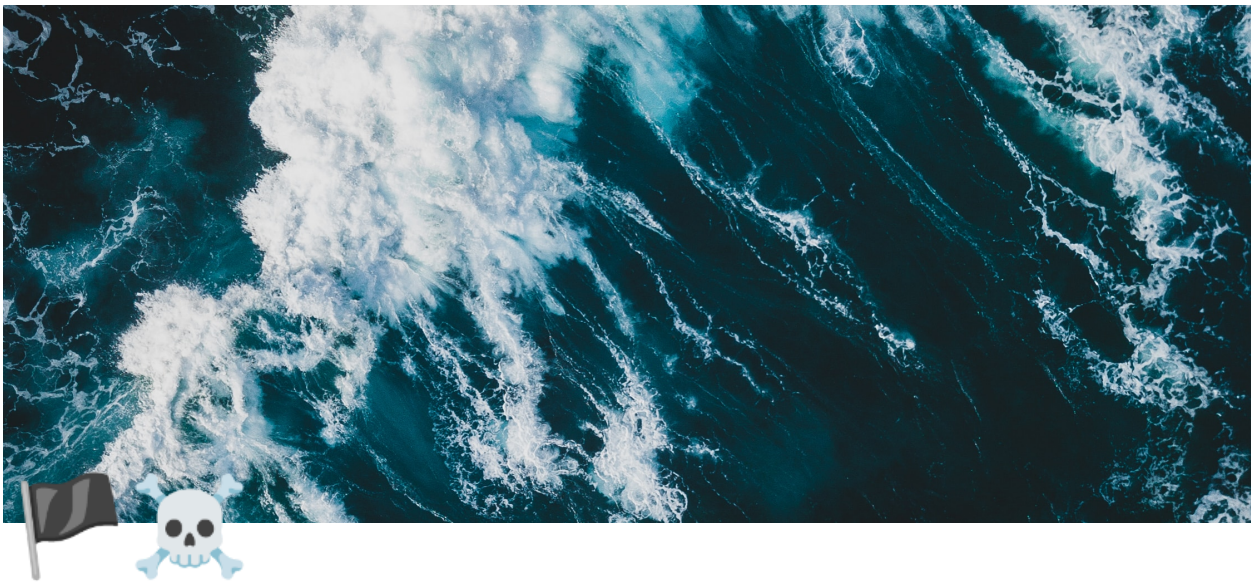

Project 2 — Pirate Game

Brainstorm

1. Video Game using p5.js — with scoreboard using socket.io
    a. Related to music — hit the key to the beat (on time) → Having a scoreboard to keep track of other players' scores.

> Socket.IO software gallery | Devpost
>
> See every software project on Devpost built with Socket.IO.
>
> https://devpost.com/software/built-with/socket-io

User Testing

For user testing, I can't necessarily say that it helped me with my actual project, and that is on me since I did not have much to user test to begin with, I still found that getting an outside perspective was really useful. Once I began to explain the concept of my game, it was really nice to receive their feedback and input as well as some tips and tricks that they thought would help

me. I discuss below a bit more about my process and how the idea that I had user tested with is different to the one I ended up going with, I wanted to share a few of the ideas that I got:

1. Using 2D grid with 1s and 0s. I had initially wanted to create a capture the flag game *(details below under Progress)*, and was trying to create the map. Somebody suggested to use a method of 1s and 0s to create a border of where the player could and could not go. I believe the Professor even shared an example of that as well, so I found that to be really intriguing, and it worked great for me before I decided to change my idea.
2. The second recommendation was to watch the Coding Train's videos. This was something that I typically usually do anyways, since I find Daniel Shiffman's videos extremely useful, but I though I should mention it anyways.
3. The last thing that was mentioned was the theme of the game — what the aesthetic should be and a lot of my classmates agreed that a minimalistic look would be great for the concept of the game, and I was more than happy to consider that when coding.

Progress & Difficulties

1. At first, I started with a game of capture the Flag. The premise is simple, steal your enemy's flag and bring back to your base to win a point. If you are intercepted on the way to the flag, or on the way back to your base, you do not get the point...
2. While that was an idea I was extremely keen on, I realized that there were lots of other elements that needed to happen for this project to be complete and I did not think that I would personally be able to complete it within the deadline.
3. I decided to change the game: my new idea was a Pirate Game!
    a. The whole concept is to collect as amny coins as possible in order to be the richest pirate in the seas.
4. While I was researching, I found a youtube video that went over creating a game with socket.io and found it extremely useful. I was able to grasp a lot of concepts from it. The concepts used were beneficial for me, but overall, the way that my game worked in comaprison to his was completely different, so it was all about applying the concepts to me.

Multiplayer Snake Game | JavaScript & Socket.io

In this video we will create a multiplayer snake game using socket.ioHungry Turtle Code YouTube Channel:https://www.youtube.com/channel/UC7Vxnf06GP6w42Lg3

https://www.youtube.com/watch?v=ppcBIHv_ZPs

5. Started with server side this time → Made sure that my server as well as my sockets were running — did the basics of console.log the users that connect as well as the ones that disconnect
6. First, I began by creating my html elements → this involved creating different divs, as well as a canvas that I would access in my code to create the game
7. I started defining some variables as well as accessing them by ID — with my canvas, i initialized it as well as set a fixed height and width for it.
    a. Later when accessing the initialized canvas size, I made sure to downscale the size. At first I had done that just to make the game move smoother, but also realized that I accidentally made something cool out of it →the fact that the players now felt like they were moving on water.
8. I wanted to test whether I could share keyboard information. One thing that I was worried about doing was controlling the characters on the screen and how the relaying of data would be achievable
    a. I decided to break it down into smaller steps. First I began by using a keydown function to test whether the information would be relayed when a certain key was pressed on my keyboard
        i. I found this to be rather simple — I used the following code, that way when a key was pressed, the key code would be displayed in the console

```
function keydown(e){
  console.log(e.keyCode);
}
```

    b. This was all and well seeing as this was currently a single player type of situation, and I was worried about how I would be able to transition it into a multiplayer setting.

        i. Regardless, I decided to pause on that for a second and focus on continuing to set up my game before I delved into the more detailed and elaborate parts of the project

9. I decided to initialize my gameState — all the relevant information needed for my grade (acceleration, position, color, score, etc...)
10. Next, I imported my socket and started with a simple console.log in my “connection” to check if it works (this is in my backend)
    a. In my frontend, I defined my socket as well as my http, and called on the emit message that I created to check if it works, and the connection between my frontend & backend was valid
11. Next, I created a game state that receives all the updates within the game
    a. this is a concept that I struggled to understand, but after watching a few videos, I was able to get it down
    b. What I did was created a function called “updateGameState” that used the gameState parameter that I initialized
        i. the whole purpose is that every time the server sends a game state message with the new game state object, the browser will receive it, and draw the game using that data
12. Watching this video really helped me understand some concepts — that is how I knew that the next steps would be to focus on the server
    a. Here, I focused on the game mechanics and how they would interact with the front-end
13. I began by figuring out the controls - how the player would move, how quick they would accelerate, how they would interact with other players as well as other components within the game
    a. To being with movement, i began with the key codes. Using the keydown function (built-in js function), I made it so that when, for example, the left arrow is pressed, a socket.emit with the name “left” would be emitted. In addition I created an “accelPlayer” function that took 3 arguments: the socket.id so that the correct player would be moved, as well as the x and y values → for example, for the left direction: x = -1 & y = 0

    i. This accel player function would later be called on with the socket name that was emitted to cause the player to move on screen

    ii. I had some issue with figuring out how to emit and where to emit, but once I figured it out, it was pretty straightforward since using key-codes in combination with the acceleration I found to be rather simple

14. Next, I added a score element. I thought it would be a cool concept if the game was just endlessly running and whoever joined the game would see people's previous high scores. There would not necessarily be an end to the game.

    a. snake.io: This is a phone app that I used to play that has a similar concept. Once. user join the game does not reset, but rather the new player starts fresh while the other players carry on with the progress that they've already created.

    b. I did not have time to figure out how to implement a database so I just made it that when the timer runs out, the game is over...

    c. To be completely honest, since this was a very last minute solution, I did not even have time to transition into a game over page since everything was finished so last minute → this is not the result I wanted but, the way that I coded it was so when the timer would run our, you would get an alert telling you that the game is over and that you would have to refresh to play again

        i. Not an ideal solution, but one I had to implement within the limited time that I had.

15. Ideally, I would have had a main screen where users join a game → once they join the same game, meaning they've created a unique id, then the game would start, but that is another thing that I did not get around to doing...

16. I also created a list for my pirates that are randomly generated here: https://www.imagineforest.com/blog/pirate-name-generator/

    a. Used a math function to randomly generate whole numbers that refer back to the array

    b. To make sure it worked, I console logged it with the socket.on connection → that way every time a new user joined, a new name was being assigned

        i. The problem with that is that it is limited in many ways. The first is that there are a set amount of names so the more the users, the more likely the names would be repeated. Second, the random function did not guarantee that two names would not be repeated even if the number of players was small.

ii. Thinking back, I feel it would have been nicer to have the users create their own user name and then have some sort of error checking to make sure that no 2 players on the server have the same name.

17. In order to play the game, I needed to make sure that each player looked unique. To keep it simple, I made it so when a user logged onto the screen, a rectangle with a unique color would appear.
    a. Since I was not using p5.js, making the rectangle appear was a bit different than what I was used to. Regardless, I found this on Stack Overflow and what it does is create a color based on a string. I though this would be extremely beneficial since a lot of my game logic focuses on the socket.id so it only made sense to have that extend even onto the color. I did not write this, but it was extremely easy to implement and had some amazing results!

```
var stringToColour = function(str) {
  var hash = 0;
  for (var i = 0; i < str.length; i++) {
    hash = str.charCodeAt(i) + ((hash << 5) - hash);
  }
  var colour = '#';
  for (var i = 0; i < 3; i++) {
    var value = (hash >> (i * 8)) & 0xFF;
    colour += ('00' + value.toString(16)).substr(-2);
  }
  return colour;
}

// https://stackoverflow.com/questions/3426404/create-a-hexadecimal-colour-based-on-a-stri
ng-with-javascript
```

18. Now for the most part, the way that I had created the code was catered towards a single player game. There were aspects of it I had used with sockets that added a more multi-player dynamic to it, but I had yet to implement the code that would allow for the game to be properly played as a multiplayer game
    a. Initially in my index.js, I had created 1 player object that contained all the attributes: position, velocity, size etc... but in order for this game to be multiplayer, I had to convert that singular object into an array of objects, that way it can be accessed and a new player can be generated whenever they log into the server

    i. Again, I struggled a bit with converting this into multiplayer and every time I would try to access an array I would run into an error and my game would come up blank → I quickly realized that a good solution was to pass the socket.id of each player as an argument, that way a unique object will be created based on the user's id

19. Next, I had to take into consideration all the other logistics of the game: how to keep the player within the border, how to handle collision based of user's position, etc...
    a. The video I used was a bit tricky in this sense because the game that the person was creating strictly used 2 players, so I had to try my best to apply the concepts to my own game
        i. Again, I found the solution to be in the user id. In the video, the man explicitly calls on the two players using their array index:

```
const playerOne = state.players[0]
const playerTwo = state.players[1]
```

        This was a bit different than what I had to do, because I did not want to have a cap on the number of players that would join a game so defining each and every single player would not be possible → instead of using index values, I used the socket.id → I found this to be an easy & efficient solution and that way I could ensure that each player had unique attributes since i knew that their id's would already be unique

20. Since the mechanics of the game were already figured out, the most difficult, as well as the part that took the most time, was the multiplayer aspect of it → implementing the sockets correctly
    a. Once I figured that out, it was smooth sailing ☠️

Final Thoughts and Reflections

As a final note, I wanted to say that using this video as a point of reference really benefitted me with this project, so I feel like I have to give credit where credit is due. While the game concept was completely different than mine, the key concepts remained the same. Especially towards the end when the guy was debugging his game because it was not going the way it was supposed to,

I found it very refreshing to see that process and seeing how his game came along as he progressed. In a way, it motivated me to implement those concepts in my own game and work to figure out my own issues within the game. While this project was extremely stressful to work on and I spent countless late nights trying to figure out the silliest of mistakes, I am glad with the result that came out, even if it is not as complete as I had hoped it would be.