CP476 Final Project

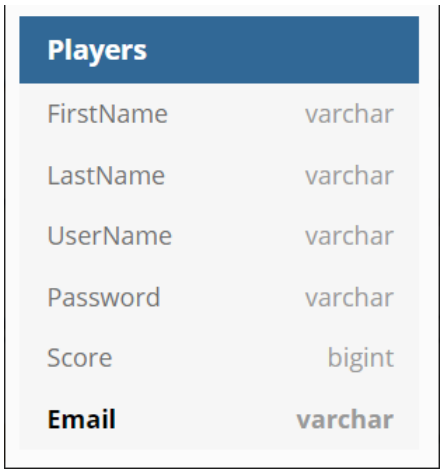
Tyler Allen - 140725940
Vishrut Sutaria - 139008636

April 2019

Contents

1 Database

1.1 E-R Diagram



2 Functionalities

2.1 Server

The server handles the connection to the database, allowing the creation of users and their login attempts. In addition, it lets the user move between pages using the side navigator. The game script is also handled by the server, where it listens for new users, and disconnects players who leave the page.

2.2 Database

Users can create an account using the 'signup.html' page, and can login to their account through the 'login.html' page. The passwords are hashed and stored in the database to increase account security.

Unfortunately the leaderboard section was unable to be completed due to time restrictions. User scores are not updated in the database, nor can they be retrieved by the leaderboard page.

2.3 Game

2.3.1 List

- Multiple users can connect to the server's game session
- Users' guns follow their mouse movements around on the canvas
- Players can move around using the WASD keys
- Each user has a camera which follows their player around the map
- Cameras and players cannot leave the boundary of the map itself
- Players collide with the house walls on the map
- Users can fire their gun by clicking on the canvas
- Bullets fired will be deleted if they connect with a house wall, boundary wall, or another player
- Players have a health bar displayed above their head
- Players take damage when hit with a bullet (health bar also decreases)
- If a player "dies" their client will be redirected to the play menu
- "dead" players will disappear on other client's canvas
- Players are spawned in a random location on the map
- Connected players are assigned one of 6 preset colors depending on how many players are currently connected
- Variables such as player speed/size/health, bullet speed/size/damage, map size, etc.. can all be changed through the server and object files

2.3.2 Explanation

We chose to go with an object-oriented approach to developing the game to keep everything modular and independent. Moving objects such as Player, Camera, Gun, and Bullet each have their own 'update' function which is called 60 times per second (our set fps is 60). Each client sends their keyboard presses and mouse coordinates/clicks to the server at the same fps so that the server knows what to pass the update functions. Upon each iteration of updating, the objects are compared against all other objects for conflicting positions. If there is a confliction with Player or Camera objects, the update does not change anything; If not, it will update the object's X and Y positions. If there is a confliction between a Bullet and Player object however, the server updates the players 'health' attribute by decreasing it by the 'damage' attribute of the bullet object.
Each client receives the necessary objects (players, guns, bullets, camera) from the server so that it can redraw everything with respect to the current state of the game.

3 Technologies Used

3.1 Server

We decided to go with node.js for our server because it seemed to be a popular choice in the modern industry. W3Schools has a helpful section to teach users how to work with it, which is where we learned how to implement various functions.
Node also allows users to install libraries (also called modules or APIs) to make tasks easier. Modules we install include:

- express: For routing between pages
- socket.io: Provides real-time bidirectional event-based communication
- express-session: Helps track the logged-in user across sessions
- cookie-parser: Accessing the cookies stored in the browser
- body-parser: Allows us to grab the input from users from the forms

- password-hash: Hashes users plain text password
- mysql: For connecting and running operations on a database

3.2 Database

Database was made using MySQL, which was taught in class. So, creating and accessing the database was simple and straight forward.

3.3 Game

Due to a lack of understanding of many JavaScript frameworks, we chose to program our game using pure JavaScript. As mentioned previously, we took an object-oriented approach to programming the game. Each object is contained in its own file which starts with 'module.exports = class Object' where Object is the name of the class. This is a recent development in the JavaScript industry, which is slowly replacing the use of object prototypes.

4 How To Use

4.1 Starting The Server

1. Install node.js and xampp on your pc
2. Create a mysql database called 'pew_pew' and import the player.sql file into it
3. Open the command prompt (cmd) or PowerShell to the project folder
4. Type 'node server.js' to run the server

To avoid the hassle of configuring everything, we will be hosting the server on our home network which can be connected to at '99.250.188.188:5000'. If there is an issue connecting, please email me at alle5940@mylaurier.ca and we will try to have it working as soon as possible.

4.2 Creating An Account

To create an account, navigate to the 'Sign Up' page and fill out the provided form. After the form is completed, press the blue 'Sign Up' button to add the account to the database. If

the account got added to the database then the user will be redirected to the front page.

Note: if you attempt to sign up with non-matching passwords, the page will hang forever. To correct this, simply check to make sure your passwords match and press the 'Sign Up' button again.

4.3 Logging In

To login to your account, navigate to the 'Login' page and enter your username and password in the form. Once your information is entered, press the blue 'Login' button to be connected to your account. If the account exist in the database then the user will be redirected to the front page.

Note: if you attempt to login with an account not previously created, the page will hang forever. To correct this, simply check to make sure your username and password has been submitted using the 'Sign Up' page before trying to login to the account.

4.4 Playing The Game

To play the game, navigate to the 'Play' page and select 'Play solo' or 'Play in teams'. You will be taken to the 'game.html' page where you will see a canvas with a player in the middle. This is your player. Move around using the WASD keys and control the gun by moving the mouse on the canvas. You can also fire your gun by clicking on the canvas.

The object of the game is to eliminate other players in the battlefield. If no other players are present, you can open a new tab and direct it to the game page. Avoid other players' bullets and try to kill them by shooting them.

If you are hit with a bullet, the red bar above your player will decrease. This is your health bar and indicates your remaining health points. Once this bar is fully depleted, your client will be redirected to the 'Play' page.

5 Lessons Learned and Challenges

The most time consuming/largest challenge from the beginning was learning how each of our technologies worked, and how they worked together. Having no previous background in JavaScript, or node.js felt like we were being pushed in a pool and told to learn how to swim. The html *canvas* element was also new to us so we had to learn how to draw images and shapes on it to simulate movement. To expand on this, lots of trigonometry was used to calculate the angle of the gun to point towards the cursor. The bullets also used this to be able to be spawned at the end of the gun, and to have a trajectory. To achieve the drawing aspect, the canvas performs multiple transitions and transformations before drawing each object, and is then returned to the default state.

Connecting multiple, concurrent users was also found to be a difficult hurdle to overcome. The server would repeatedly crash as we attempted to pass information between itself and the clients. There happened to be a race condition each time a new user connected which had to be resolved by delaying the client by 1000ms.

Learning node.js proved to be a larger task than initially anticipated, on top of which was discovering the right modules to use for our needs. Even when we thought we had the right module for a task, we sometimes ended up spending hours learning it only to find a better module to fit the role.

6 Next Steps

Going forward we would like to implement:

- Leaderboard functionality
- Player usernames being displayed over their player
- Splitting the game server into multiple independent servers (10 players per lobby)
- Team mode where users are placed into 1 of 2 teams
- Dedicated hosting

7 Similar Online Game

Name: Surviv.io Battle Royale

URL: http://surviv.io/

Similarities	Differences
Last-man-standing game	We have less game modes
Sign Up and Login	They track how much damage each player has given/taken, and how many kills that person has
Simple coloured shapes	Our game only has 1 weapon
Bullets can be seen	Their game has more map objects
House/wall collision	Their sign up can be done using different social media
	Can be played on a phone