Matthew Fong

Arcade-Jam Documentation

**App purpose:**

For my second project, I took inspiration from a game called Disco Jam and implemented a small 4-player team-based game. Two players are on the red team and the other two are on the blue team. Each player can shoot projectiles and score for their team if their projectile reaches the other side of the canvas screen. Players can also block the other team’s projectiles by activating a shield and when the projectile collides with the shield, the projectile will be deleted. The first team to get 3 points wins the game.

**Websockets use:**

For websockets, I am using socket.io. The server listens for join updates that sets up the connected socket. In terms of movement, the server listens for movement updates. Each player (client) sends movement, fire projectile, and shield updates to the server. The server listens for these events and handles these requests respectively. For movement updates, the server just sends back data to all clients connected in the room. When projectiles are fired, the server sets the projectile’s direction and updates the movements on the server-side. Furthermore, when a projectile is fired and the server is notified, the server will check if it collides with any of the other team’s player’s active shield. Similarly, when shield updates are sent, the server immediately sends them back to all clients in the room. The server also listens for updating and displaying both teams’ points. Additionally, the server listens for a reload request from each connected client and a game over state. The reload request will be explained later in the documentation. The server listens for a game over state and when a team gets 3 points, the winning team is declared and the server sends a response to each client in the room to create and display a ‘Play Again’ button in which the clients can click to set up a new game to play, instead of having to reconnect to a new websocket room.

**What went right:**

Overall, implementing the core gameplay and idea of my game went well. I felt I was able to implement all the necessary game mechanics to create a very simple, yet engaging game for the players. The look and feel was also very simple and I feel the overall design and colorway was aesthetically thematic. Implementing my above and beyond features also went smoothly. Additionally, before a new game starts, all clients connected are placed in sort of a skirmish game state. Players are still free to move and shoot projectiles and shield. However, they will not gain points in this “skirmish” mode. My decision to implement this in the game is to hopefully allow players to get a feel of the game, movement-wise, shooting-wise, etc.

**What went wrong:**

One of the most time-consuming game implementations that I struggled quite a bit with was structuring data, passing information around on the server-side (physics.js and sockets.js) and sending updated data back to the client, that all related to collision detection between a projectile and another player’s shield. Rather than checking for these collisions every time the server received a movement update from any of the clients in the room, instead I set an interval on the server-side to check for these collisions. During development, I ran into several issues, one of which was that the projectile was being deleted on the server-side, but was not sending the update bullet’s array back to the clients. As a result, all clients continued to draw the projectile traveling across the canvas screen, even though collision was detected. One approach I took in handling this issue was to try to pass the io variable to physics.js and use it to emit to all clients when detection occurred. However, this did not work because the io variable was still undefined when it was exported. Another approach I took was making a single projectile variable in sockets.js and passing that to physics.js to later be updated when collision was detected. This too did not bode well as the value of the variable in sockets.js was not being updated. Finally, I opted to create a helper function that will be called in physics.js when collision occurs. This helper function will then splice the projectile from the proper client’s bullet array, and will emit the updated data to all clients.

**Future Improvements:**

If I were to continue working on the game, there a several features and improvements I would like to make. Firstly, as the game stands as of now, when a player (client) disconnects, the 3 remaining players cannot start a new game. Since a new game requires 2 full teams (or 4 players), there is no way for the 3 remaining players to remain in their room and score for their teams. Therefore, I would like to implement the feature that allows a new user to connect to a 3 player room even after their fourth player has disconnected. Another feature I would like to implement is perhaps different projectile directions. The projectiles only fire linearly; I think it would be a more engaging and exciting game experience if the players can shoot their projectiles in different angles and have the projectiles bounce off the top and bottom of the canvas screen. Another implementation I would consider adding to the game is to add an event for when a projectile collides with an actual player on the opposing team. As of now, nothing happens, because I wanted the players to actually use their shields to block projectiles instead of just body-blocking them with their player avatars. However, I could add a feature for the case that a projectile collides with the player that penalizes the player for not using their shield. This could be either reducing the team’s points by 1 or even temporarily slowing down the player’s movement.

**Above and beyond:**

One of my above and beyond features is the ability for each player to select which team to play on. On top of that, players can also select which “position” they would like to play on their team; player 1 for both red and blue teams is positioned in the “back”, or rather closer along the sides of the canvas screen, and player 2 is positioned in the “front”, or rather closer to the center of the canvas screen. Additionally, another feature, but rather small one is if a player tries to select a position and team that is already taken, the server will respond with an “error joining” message, informing the player that that position is already selected. The player will then have to select another position that is still open. Finally, in terms of the player shooting mechanic, rather than allowing each player to spam projectiles at one another, which I imagine would be very chaotic and possibly result in a very dull game experience, I decided to implement a reload feature. Each player may shoot 3 projectiles, however, after firing all 3, the player must wait a small delay time to reload, or restock on projectiles before firing more.

**References:**

* Avatar image: <http://1.bp.blogspot.com/-ffa8lLD5LUY/UvRZhHfzbkI/AAAAAAAABXg/IbAquczH61Q/s1600/st1.png>