MODULE: CST2120

ASSIGNMENT: CW 1 Game Report

PROFESSOR: Mr. Engie

Student Name: Lebohang Khasipe

Student ID: M00860241

*INTRODUCTION*

First, the report will cover what the game is about, how to play the game, the game implementation and what each class is responsible for doing. It will cover how each game process is done, such as the movement of the player's tank, shooting, collision effect, etc. It will also cover the registration and login process using the local storage session storage and the leaderboard implementation.

**Game Title:** TANKERS

Game description:

Firstly, the game consists of a playable tank controlled with the W and S keys. The game's goal is to survive as long as possible fighting enemy tanks without getting hit. Furthermore, the more enemy tanks you kill, your score will get incremented by 1; the higher the score, the higher you’ll be placed on the leaderboard. You use the mouse cursor to aim and the spacebar to shoot. Depending on where your cursor is on the map, the point where you hit the bullet will impact there. If you get hit by an enemy tank bullet, your lives will go down by one and if you lose all three lives, game over!

HOW TO PLAY?

1. Click the start button to start the round.
2. The game will then start, and enemy tanks will spawn randomly from any side of the screen.
3. Use the W and S to move the tank around and the mouse cursor to aim. It will aim there depending on where your cursor is; there is a space bar to shoot back!
4. To win the game, you must survive as long as possible without dying. You will have three lives; the game ends once all three lives are gone.

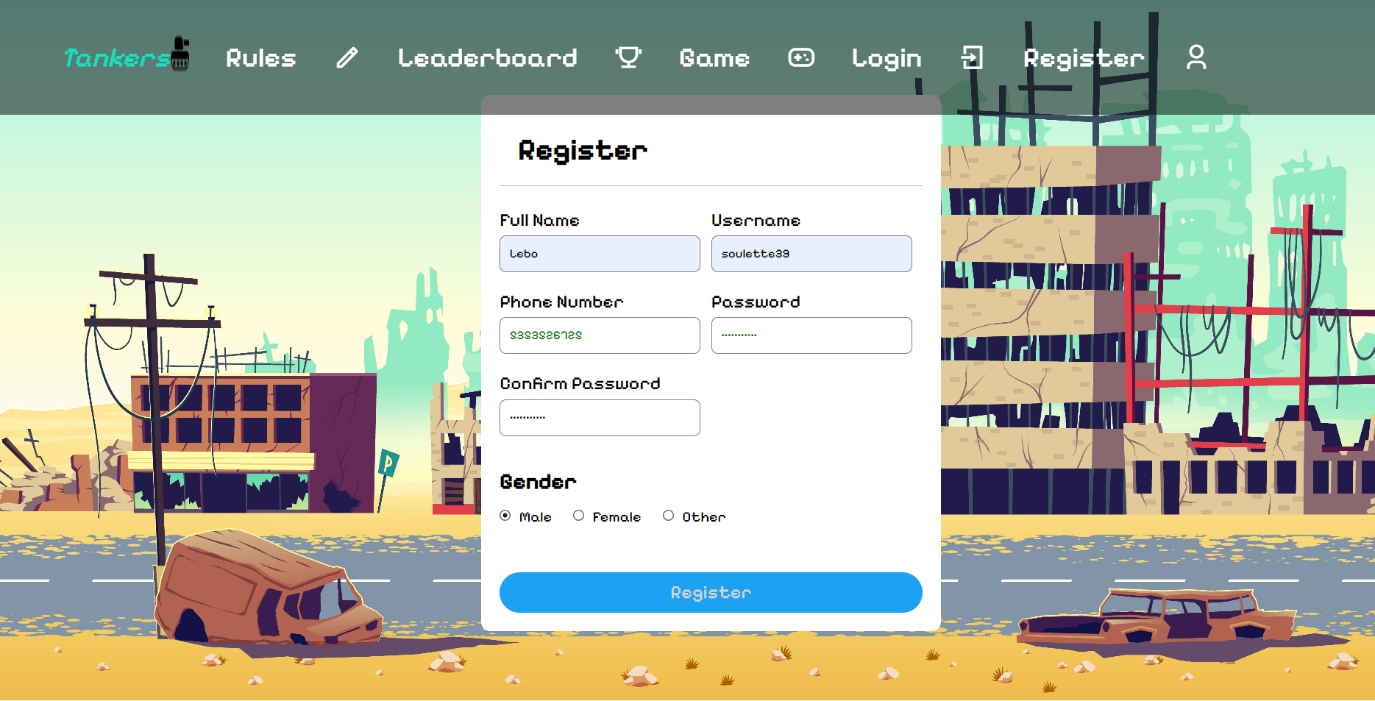
GAME WEBSITE

* Rules page
* Game page
* Login page
* Register page.
* Leaderboard page

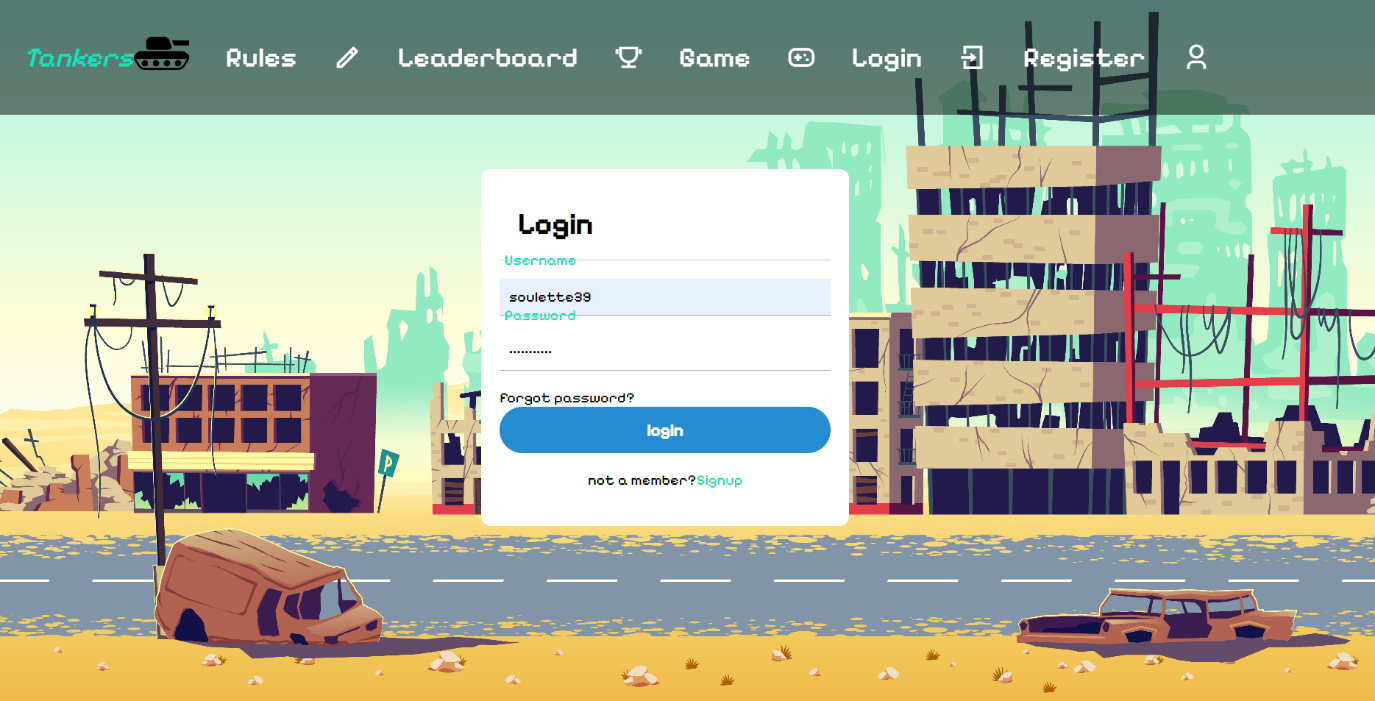
Gameplay



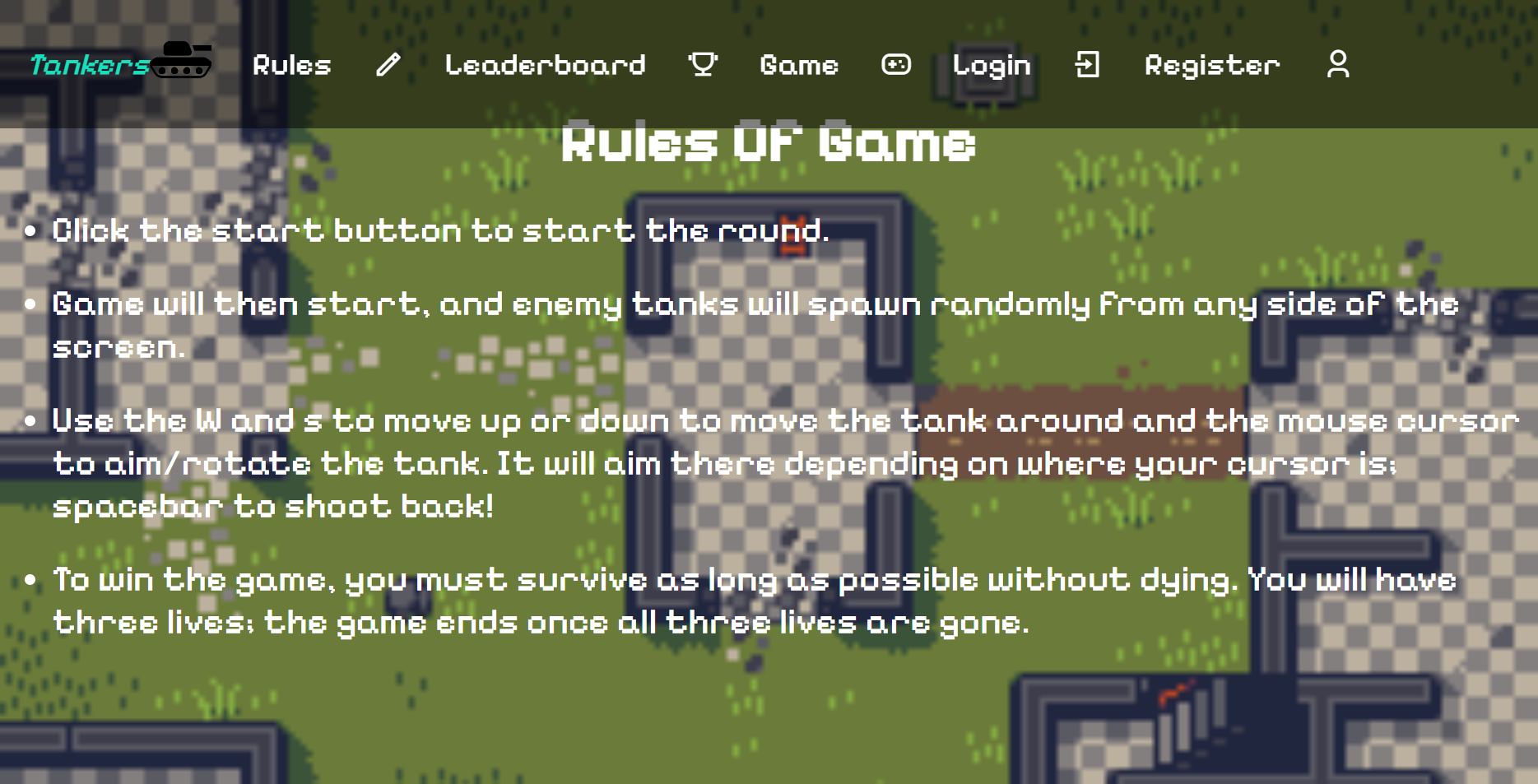
Registration



Login



Rules



Game implementation

CLASSES

***`Game`*** class:

First, the `***game`*** class is responsible for managing the game's state, including the player and enemy tank, the player and enemy tank bullets, the score of the game, the lives of the player tank, and the game logic, such as game over, resting after lost, looping to update game state and so on. For instance:

* Initializing variables
* Enemy tank generation
* Shooting mechanism for player Tank and enemy tank
* Movement of player tank based on keypress events.
* Collision detection between bullets and tanks
* Keeping score and lives
* Handling game-over
* Handling reset of game after game over
* Hiding buttons and starting the game
* Create a game loop using the requestAnimationFrame method for smoother animations.

***`PlayerTank`*** class:

Moreover, the ***`PlayerTank`*** class represents the playable tank the user uses to play; it handles the following:

* Movement of a player tank moving forward and backward.
* Rotation to update player tank rotation based on the cursor position.
* Event handling: listen for key-down events to handle the tank movement.
* Drawing: draw the tank on the canvas.

***`EnemyTank`*** class:

The ***`EnemyTank`*** class extends the ***`PlayerTank`*** class and represents the enemy tank. It includes methods for shooting bullets at the player tank and overrides the *`draw*` methods to handle the enemy tank behaviour.

***`Bullet`*** class:

The ***`Bullet***` class defines the properties and behaviour of bullets, such as the bullet's movement on the canvas and the drawing of the bullet. A new instance of a bullet is created when a tank shoots.

Local Storage and leaderboard (form validating and error handling):

Furthermore, I created functions to update the leaderboard using the local storage. Firstly, the user registers and information are stored locally. Then, the user logs in and stores the user in the session storage. It retrieves user data, sorts it based on the score from highest to lowest, and then displays only the top 4 scores on the leaderboard table.

In addition, for form validation and error handling, I use a regex to force a password requirement and a phone number regex to ensure you have at least ten numbers as your phone number. Also implemented where the password is constantly checked, it updates the input colour based on whether it meets the criteria. It is implemented using the *`checkPassword`* function and the same with the *`checkPhoneNumber`.* Also, check if the passwords match; if not, an error message is displayed; if the password meets the criteria, also check if the username already exists in the stored user data and if the username is unique. Then, a new user object is created and added to the `users` array and stored in the local storage, implemented using the *`register`* function.

Furthermore, error messages are constantly displayed if an error occurs, for instance, if password criteria are not met, and the username is not unique. In addition, it gives successfully registered and logged-in messages, respectively, showing the user dynamic feedback accordingly.

CHALLENGES FACED:

Lastly, regarding this web game project, I faced many issues in the implementation process. For instance, I failed to implement the enemy tank moving towards the player tank at a certain distance for it to move backwards if the player tank tries to get closer. I also struggled immensely regarding the bullet shooting logic but eventually solved the issue at hand, fortunately enough.

Moreover, the issue was figuring out how to shoot the bullet at the x and y positions of the enemy tank or player tank, respectively, and calculating the angle at which the bullet would travel in the canvas. However, I was able to use the rotation angle to solve this issue and fetch the player tank and enemy tank x and y positions and set it as the bullet position as well. Still, regarding the logic of the enemy tank shooting at the player tank rather than at the cursor's rotation angle, I could not alter the code in this regard.