<u>Project Title:</u>

Tetris Warriors

<u>Group Members:</u>

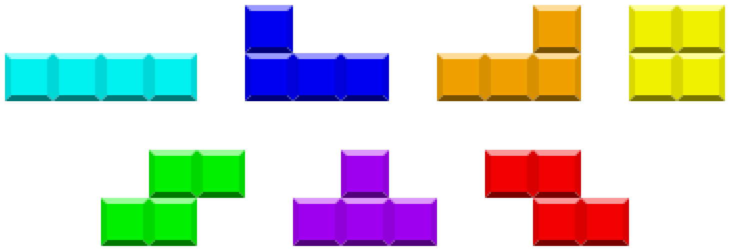
Yousuf Aijaz (ya09222)

Taha Zahid (tz09220)

<u>CS-224</u>

Project Overview

We will be creating a Tetris game using the Simple and Fast Multimedia Library (SFML) for graphics, player input. In this game players arrange and rotate shapes (called tetrominoes). There are seven types of tetrominoes (four blocks each). Players can rotate the shapes as they fall to fit them into place. Player’s aim is to make sure that they keep making solid rows without any gaps. When a player makes a solid row, it disappears, his line score gets incremented and the game continues. The Level increases once the player reaches a certain line score. The game ends if a new tetromino can’t spawn into the game as the player fails to create solid rows in time.



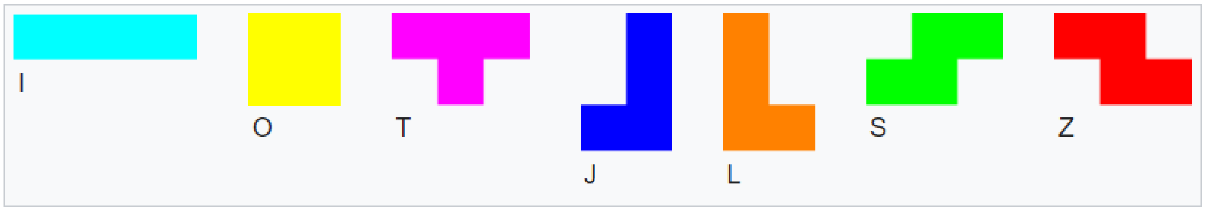
Game Structure

The game will have three main screens:

1. **Start Screen**: Shows options to start the game or quit and displays the games name.
2. **Gameplay Screen**: Shows the Tetris grid, the player's current level, number of lines completed, and the upcoming tetrominos. Players can rotate, move, and drop the tetrominoes as they try to form complete lines and increase their level.
3. **Game Over Screen**: Shows "Game Over" along with the player’s number of lines completed and their level and an option to restart or quit the game.

Class Design (UML Overview)

1. **InitialScreen**: Initiates the game flow, such as starting the game, and switching between screens
2. **Playfield**: Represents the Tetris grid, manages where blocks go, checks for completed lines, and adjusts the level as needed. Keeps track of number of lines completed, level, and upcoming shapes.
3. **Tetromino (Main Class)**: Represents the common properties and behaviors between the blocks. Handles the block's rotation, movement, and position.



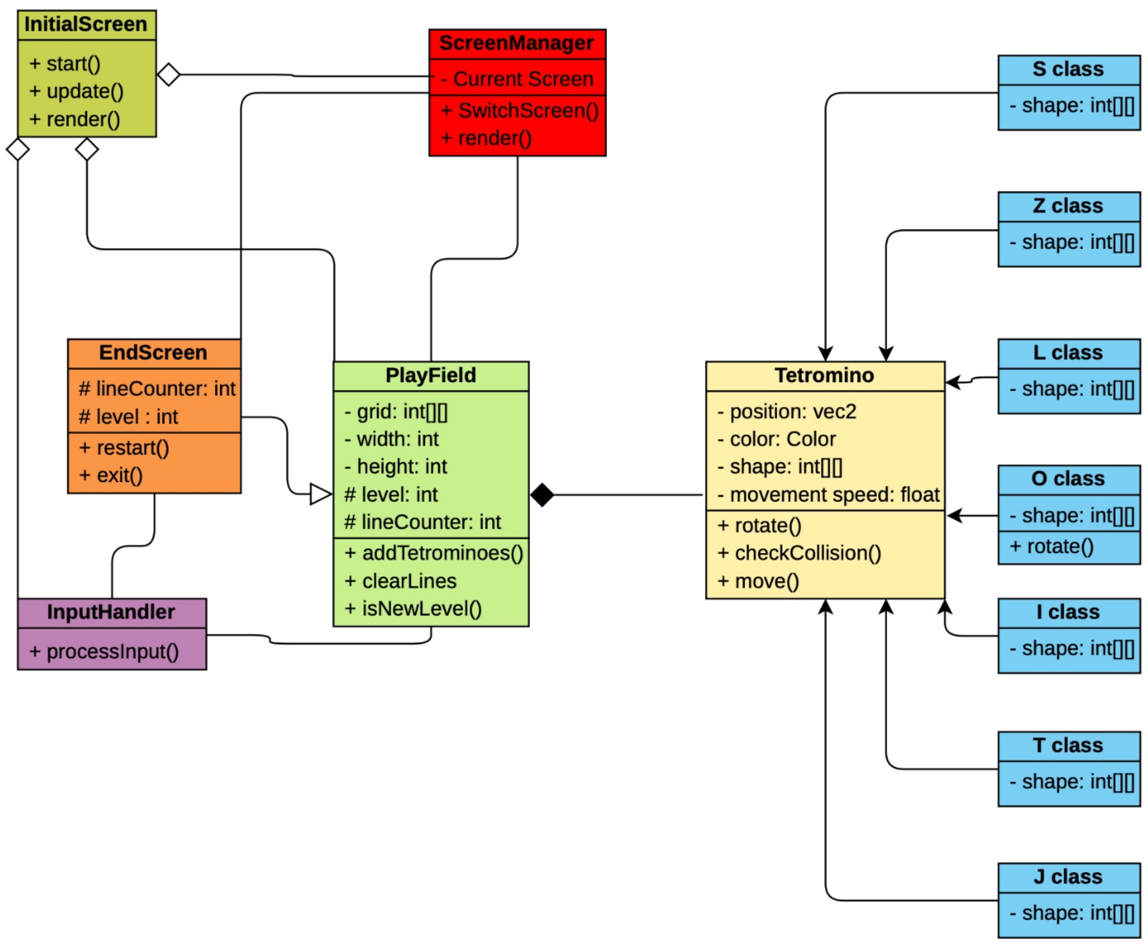
4. **I, O, T, J, L, S, Z (7 Sub Classes Of Tetromino):** Represents matrix for the shape and override any properties of rotation, movement, and position if needed. For instance, **O** doesn't need to be rotated as there won't be a visible change so we can override the rotate function.
5. **InputHandler**: Manages keyboard inputs for player actions (move left/right, rotate, drop).
6. **ScreenManager**: Controls the different screens (start, gameplay, game-over) and switches between them as needed.

Each class will be set up with OOP principles. This design will make it easy to understand and modify each part of the game independently.

UML

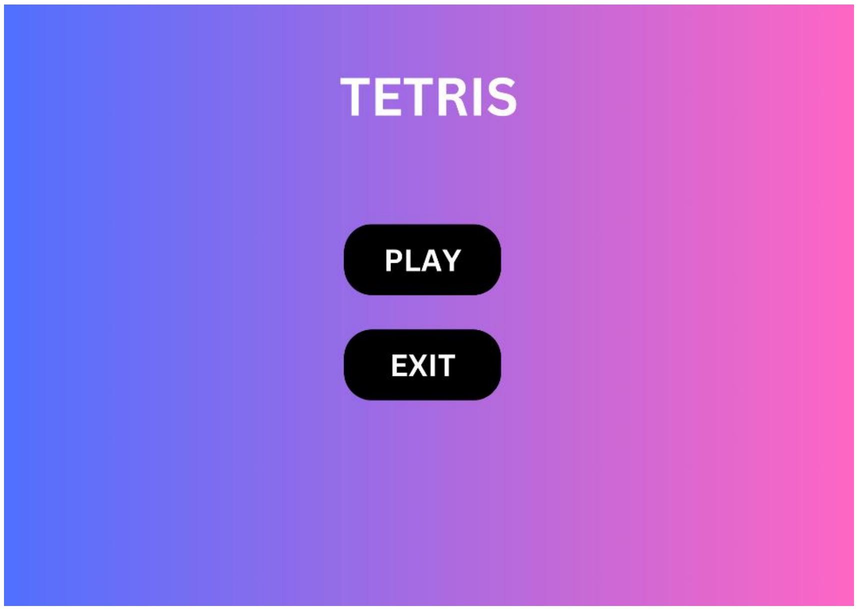
The UML diagram will include the class names, their main parts, and how they interact. This structure keeps each part of the game simple and easy to understand, with each part having a clear job.

UML Digram

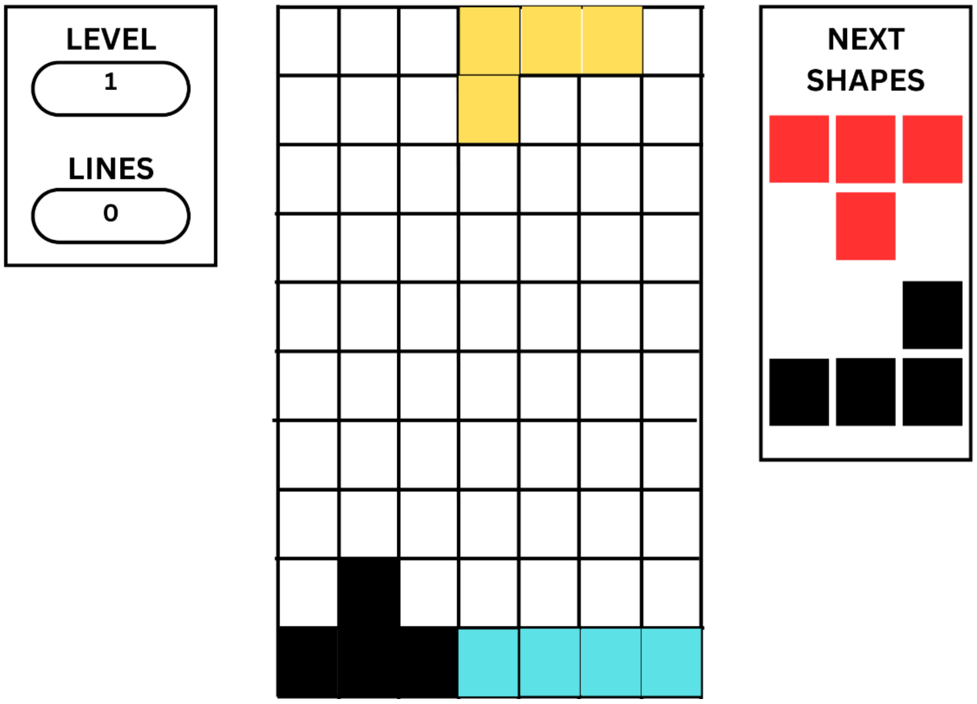


Sample Screens

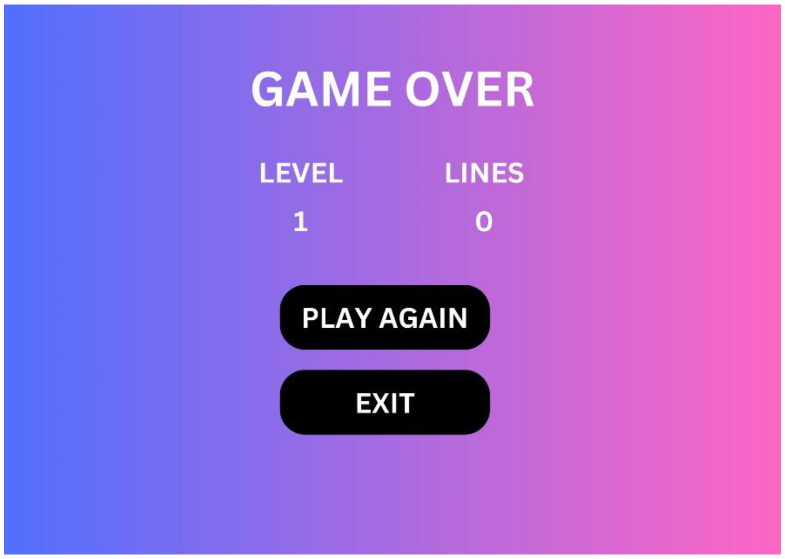
1. **Start Screen:** Shows the title "Tetris" and two button "PLAY" and "EXIT" so the user can either start the game or exit the program.



2. **Gameplay Screen:** Shows the Tetris grid with active and upcoming tetrominoes, the current lines, and level.



3. **Game Over Screen:** Displays "Game Over" along with the fina level and lines score, and options to play again or quit the game.



Conclusion

This project will bring Tetris to life using SFML and apply important object-oriented programming concepts like encapsulation and class relationships. The design, screens, and features are planned to create a complete, engaging game that meets the project requirements