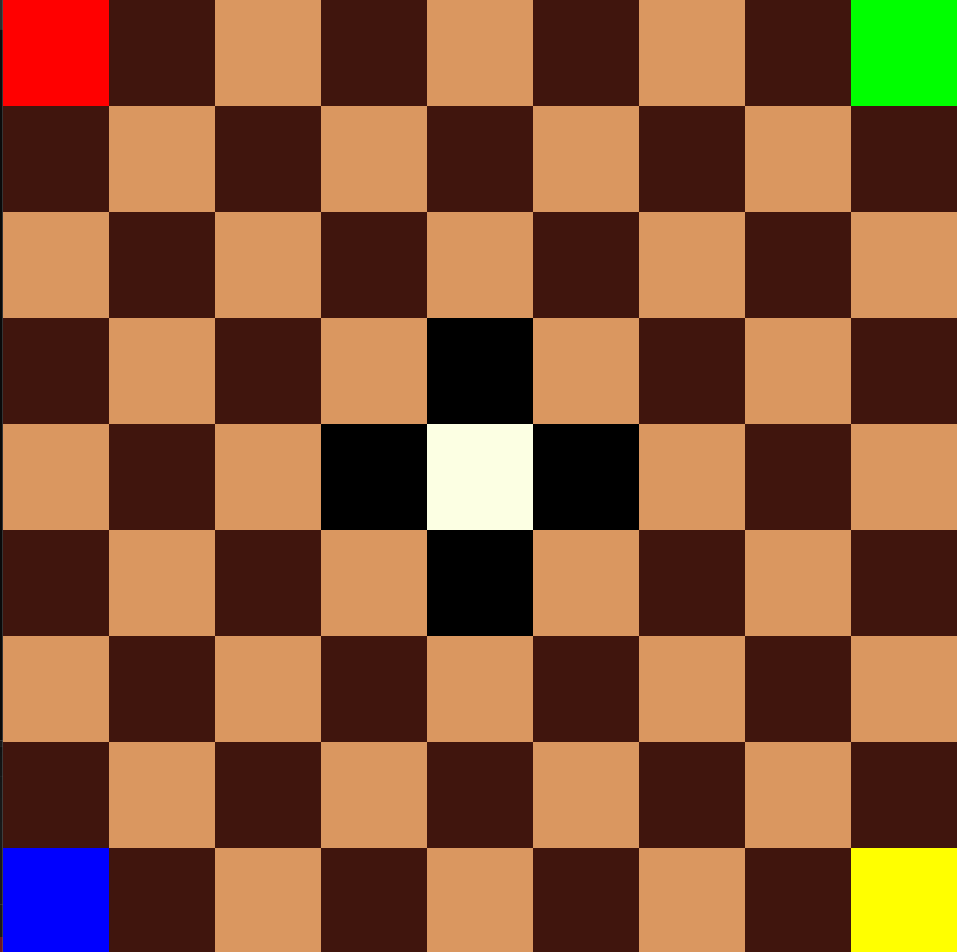
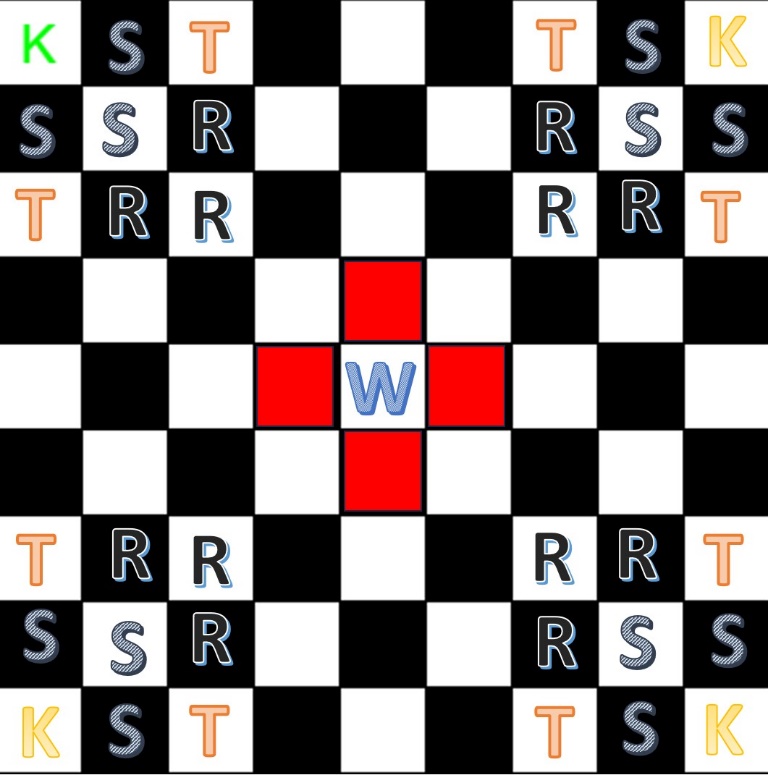
**Project Name- QuattroRealm**

Game Description: QuattroRealm

QuattroRealm is a strategic board game designed for four players, each commanding their own army consisting of nine pieces.

**Board-**

**Board.cpp-**

#include <SFML/Graphics.hpp>

class **Board** {

*private***:**

*int* *size***;**

*int* *windowWidth***;**

*int* *windowHeight***;**

*int* *cellSize***;**

    std**::**vector**<**std**::**vector**<**sf**::**Color**>>** cells**;**

*public***:**

**Board**(*int* **size,***int*  **windowWidth,***int* **windowHeight**) **:** *size*(**size**)**,** *windowWidth*(**windowWidth**)**,** *windowHeight*(**windowHeight**) {

*cellSize* **=** **windowWidth** **/** **size;**

        cells**.**resize(**size,** std**::**vector<sf**::Color**>(**size,** sf**::**Color**::**White))**;**

**for** (*int* i **=** 0**;** i **<** **size;** **++**i) {

**for** (*int* j **=** 0**;** j **<** **size;** **++**j) {

**if** ((i **==** 4 **&&** j **==** 3) **||** (i **==** 4 **&&** j **==** 5) **||** (i **==** 5 **&&** j **==** 4) **||** (i **==** 3 **&&** j **==** 4)) {

                    cells[i][j]**=** sf**::**Color**::**Black**;**

                }

**else** **if** ((i **+** j) **%** 2 **==** 0) {

                    cells[i][j] **=** sf**::**Color(218**,** 151**,** 96)**;**

                }

**else** {

                    cells[i][j] **=** sf**::**Color(64**,** 21**,** 13)**;**

                }

            }

        }

        cells[0][0] **=** sf**::**Color**::**Red**;**

        cells[0][**size** **-** 1] **=** sf**::**Color**::**Blue**;**

        cells[**size** **-** 1][0] **=** sf**::**Color**::**Green**;**

        cells[**size** **-** 1][**size** **-** 1] **=** sf**::**Color**::**Yellow**;**

        cells[**size** **/** 2][**size** **/** 2] **=** sf**::**Color(252**,** 255**,** 227)**;**

    }

*void* draw(sf**::RenderWindow***&* **window**) {

**for** (*int* i **=** 0**;** i **<** *size***;** **++**i) {

**for** (*int* j **=** 0**;** j **<** *size***;** **++**j) {

                sf**::**RectangleShape cell(*sf***::**Vector2f(cellSize**,** cellSize))**;**

                cell**.**setPosition(i **\*** *cellSize***,** j **\*** *cellSize*)**;**

                cell**.**setFillColor(cells[i][j])**;**

**window.**draw(cell)**;**

            }

        }

    }

*void* handleMouseClick(*const* sf**::Vector2i***&* **mousePos**) {

*int* row **=** **mousePos.***y* **/** *cellSize***;**

*int* col **=** **mousePos.***x* **/** *cellSize***;**

*// Handle move based on the selected cell*

*// handleMove(row, col);*

    }

}**;**

*int* main() {

    sf**::**RenderWindow window(*sf***::**VideoMode(960**,** 960)**,** "QuattroRealm")**;**

*const* *int* windowWidth **=** 960**;**

*const* *int* windowHeight **=** 960**;**

*const* *int* boardSize **=** 9**;**

**Board** board(boardSize**,** windowWidth**,** windowHeight)**;**

**while** (window**.**isOpen()) {

        sf**::**Event event**;**

**while** (window**.**pollEvent(event)) {

**if** (event**.***type* **==** sf**::**Event**::**Closed) {

                window**.**close()**;**

            }

**if** (event**.***type* **==** sf**::**Event**::**MouseButtonPressed) {

                sf**::**Vector2i mousePos **=** sf**::***Mouse***::**getPosition(window)**;**

                board**.**handleMouseClick(mousePos)**;**

            }

        }

        window**.**clear()**;**

        board**.**draw(window)**;**

        window**.**display()**;**

    }

**return** 0**;**

}

The game features four distinct types of pieces, each with unique abilities and characteristics:

1. Commander(1)

2. Rifleman(3)

3. Sniper(3)

4. Tank(2)

**1. Commander (1 per player):** Represented by a distinctive PNG image, the Commander is the main piece in each player's army. If the Commander is eliminated, the player controlling it is removed from the game, and the remaining pieces of that player are assimilated into the army of the player who eliminated them. The Commander possesses the ability to move one step at a time, exclusively diagonally, but has the flexibility to capture enemy pieces in any direction. Any piece of other players can kill the commander.

**Class Pieces and Class commander**

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Class Piece\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Piece is an abstract base class representing a generic game piece.*

*//It contains common attributes and methods shared by all types of pieces.*

class **Piece** { *//Khaas aapke liye banai gayi abstract class*

*protected***:**

    sf**::**Texture *texture***;**

    sf**::**Sprite *sprite***;**

*//sprite kya hoti... Isko jananne ke liye you should read the the word document... Aise hi nahi bheja hai...*

*int* *playerID***;**

*public***:**

**Piece**(*int* **playerID**) **:** *playerID*(**playerID**) {}

*// Humara pyare pure virtual functions*

*virtual* *bool* isValidMove(*int* **newRow,** *int* **newCol**) *const* **=** 0**;**

*virtual* *bool* canCapture(*int* **newRow,** *int* **newCol**) *const* **=** 0**;**

*void* setPosition(*float* **x,** *float* **y**) {

*sprite***.**setPosition(**x,** **y**)**;**

    }

*void* draw(sf**::RenderWindow***&* **window**) *const* {

**window.**draw(*sprite*)**;**

    }

}**;**

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Class Commander\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Commander is a derived class representing the Commander piece.*

*// We have overridden the isValidMove and canCapture methods to implement*

*// the specific movement and capturing rules as we have described.*

class **Commander** **:** *public* **Piece** {

*public***:**

**Commander**(*int* **playerID**) **:** **Piece**(**playerID**) {

*// Loading commander texture from file*

*texture***.**loadFromFile("commander.png")**;**

*// Mr. Best in the Biz, please share the the commander image also so that it can be placed here.*

*sprite***.**setTexture(*texture*)**;**

    }

*//Overriding is sometimes life threatning but...*

*//Let's overriding isValidMove for Commander*

*bool* isValidMove(*int* **newRow,** *int* **newCol**) *const* *override* {

*// Commander should move only one step diagonally.*

*int* rowDiff **=** abs(**newRow** **-** *sprite***.**getPosition()**.***y*)**;**

*int* colDiff **=** abs(**newCol** **-** *sprite***.**getPosition()**.***x*)**;**

**return** (rowDiff **==** 1 **&&** colDiff **==** 1)**;**

    }

*// Overriding canCapture for Commander*

*bool* canCapture(*int* **newRow,** *int* **newCol**) *const* *override* {

*// Checking if the target position is within one step in any direction.*

*int* rowDiff **=** abs(**newRow** **-** *sprite***.**getPosition()**.***y*)**;**

*int* colDiff **=** abs(**newCol** **-** *sprite***.**getPosition()**.***x*)**;**

**return** (rowDiff **<=** 1 **&&** colDiff **<=** 1)**;**

    }

}**;**

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

Some SFML documentation that we are using-  
1. **Sprite and texture:**

In SFML (Simple and Fast Multimedia Library), a **Sprite** is a graphical entity used for rendering images onto the screen. It represents a 2D texture that can be manipulated, transformed, and drawn within an SFML window.

