MINI PROJECT

(2020-21)

# SFML Graphics based chess game

## MID-TERM REPORT

**Institute of Engineering & Technology**

**Submitted by-**

Harshvardhan Singh (181500264)

Priyanshu Rajpoot (181500514)

**Supervised By:-**

**Mr. Mandeep Singh**

Technical Trainer

**Department of Computer Engineering & Applications**

## Contents

## Abstract

In recent years, computer games have become a common form of entertainment. Fast advancement in computer technology and internet speed have helped entertainment software developers to create graphical games that keep a variety of players' interest. The emergence of artificial intelligence systems has evolved computer gaming technology in new and profound ways. Artificial intelligence provides the illusion of intelligence in the behavior of NPCs (Non-Playable-Characters). NPCs are able to use the increased CPU, GPU, RAM, Storage and other bandwidth related capabilities, resulting in very difficult game play for the end user. In many cases, computer abilities must be toned down in order to give the human player a competitive chance in the game. This improves the human player's perception of fair game play and allows for continued interest in playing.

# Introduction

## 1.1 General Introduction to the topic

This project implements a classic version with a Graphical User interface. The Chess game follows the basic rules of chess, and all the chess, and all the chess pieces only move according to valid moves for that piece. The aim of the project is to provide both a chess as well as chat facility in one application such that two users can play the chess against each other.

Our implementation of Chess is for two players(no Artificial Intelligence). It is played on an 8x8 check board, witha square in each player's lower left corner.

We successfully created a GUI based version, inheritance and templates, as specified. Despite several unusable bugs in the GUI, our chess program is a great, user-friendly game for two players.

As chess game is generally popular to make the brain more able in thinking strategically, additionally like our project provides better view of graphics enhancing the fun and making the game more interactive. Our project play an important role in making strategy, enhances brain's strategy making ability with very good interaction and graphics presentation.

## Area of Computer Science

### Why chess game using SFML ?

**Simple and Fast Multimedia Library** (**SFML**) is a cross-platform software development library designed to provide a simple application programming interface (API) to various multimedia components in computers. It is written in C++ with bindings available for C, Crystal, D, Euphoria, Go, Java, Julia, .NET, Nim, OCaml, Python, Ruby, and Rust.Experimental mobile ports were made available for Android and iOS with the release of SFML

SFML handles creating and input to windows, and creating and managing OpenGL contexts. It also provides a graphics module for simple hardware acceleration of 2D computer graphics which includes text rendering using FreeType, an audio module that uses OpenAL and a networking module for basic Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) communication.

SFML is free and open-source software provided under the terms of the zlib/png license.

It is available on Linux, macOS, Windows and FreeBSD. The first version v1.0 was released on 9 August 2007, the latest version v2.5.1 was released on 15 Oct 2018.

SFML provides a simple interface to the various components of your PC, to ease the development of games and multimedia applications. It is composed of five modules: system, window, graphics, audio and network.

## **1.3 Hardware Requirements**

- Memory [4GB RAM (or higher)]
- Intel core Pentium 64-bit Processor (or higher)

## **1.3 Software requirements:**

- OS:  Windows 7(or higher)

## Objective

In order to bring the people back in such games where they could enhance their brain's strategy making ability with fun. We have focused more on interaction, to make the user not feeling of less than a real chess. It provides a user-friendly interactive environment to the users of the application that helps them to play. The care has been taken that the application has less CPU usage, so that other applications can also be performed, if required.

Chess is for two players. It is played on an 8x8 checked board, with a square in each player's lower left corner.

## Implementation Details

We are going to use sfml library here, we are also attaching a respective png's for each character in the game for a better GUI.

We are using the concept of OOPS, by defining each entity's behaviour, attributes by splittingly defining. Each character has its own header file in which we are going to define their respective png image, their legal moves etc... which is making the project more easy in case bugs found .

## Progress

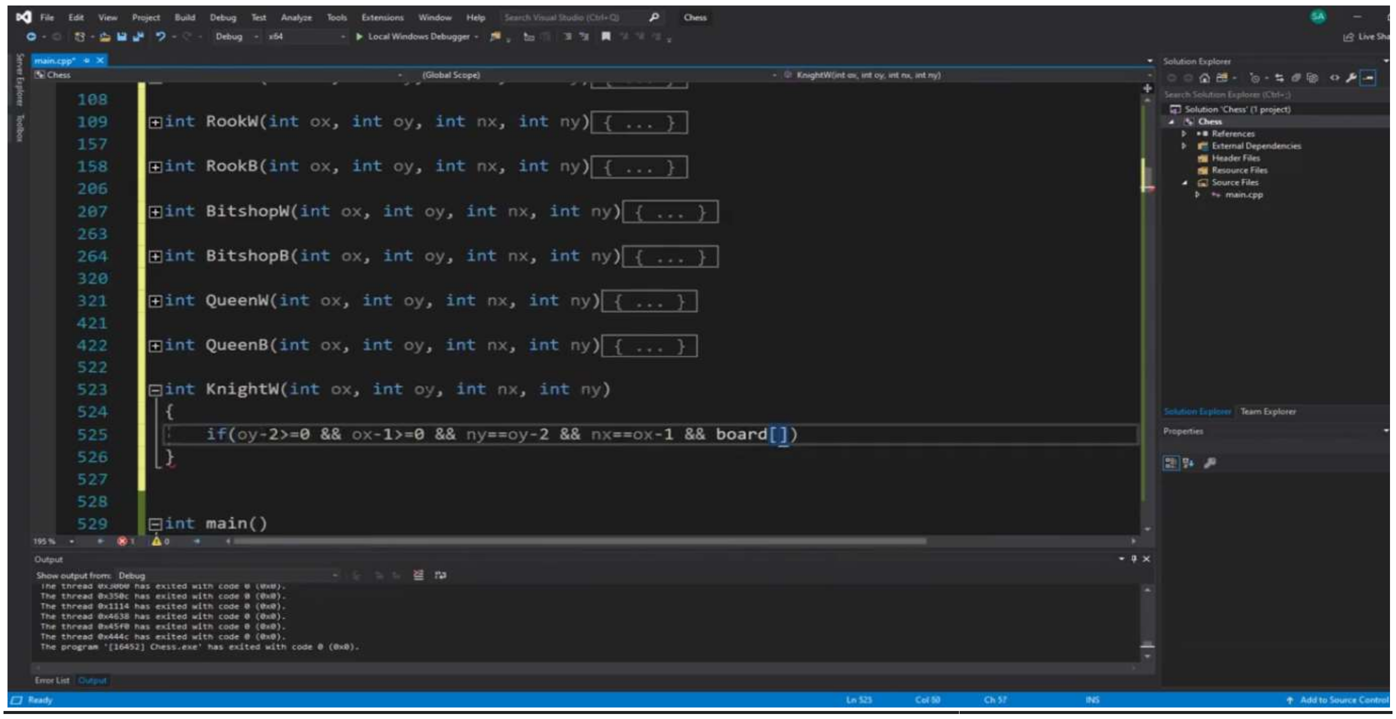
1) Part 1 is completed.

**Defining its own header file for each character in the chess game.**

2) Currently working on part 2.

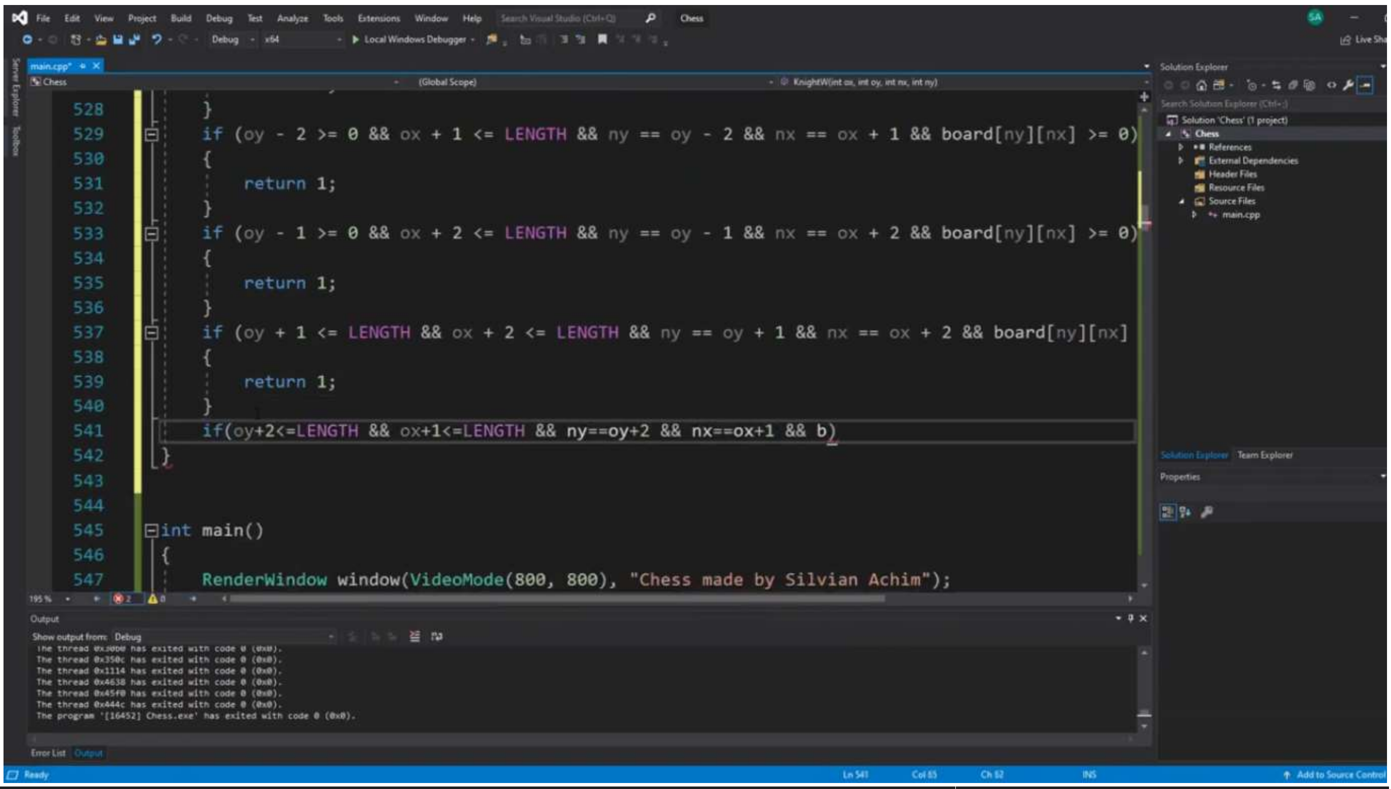
**To define remaining character’s own header, and merging them all in a common function which will have a main function.**

## SCREENSHOTS



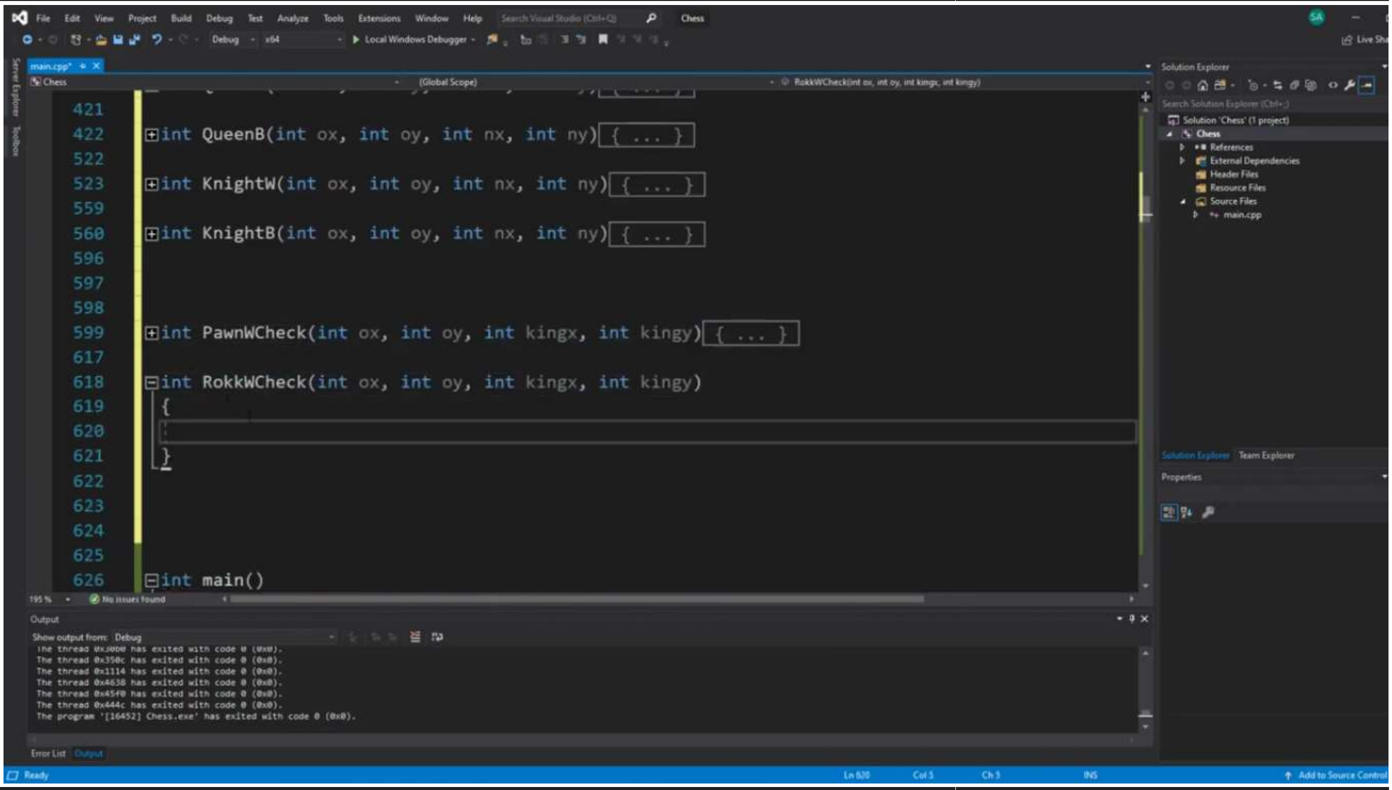
```
int RookW(int ox, int oy, int nx, int ny){ ... }

int RookB(int ox, int oy, int nx, int ny){ ... }

int BitshopW(int ox, int oy, int nx, int ny){ ... }

int BitshopB(int ox, int oy, int nx, int ny){ ... }

int QueenW(int ox, int oy, int nx, int ny){ ... }

int QueenB(int ox, int oy, int nx, int ny){ ... }

int KnightW(int ox, int oy, int nx, int ny)
{
    if(oy-2>=0 && ox-1>=0 && ny==oy-2 && nx==ox-1 && board[])
}



int main()
```

```
The thread 0x30b0 has exited with code 0 (0x0).
The thread 0x350c has exited with code 0 (0x0).
The thread 0x1114 has exited with code 0 (0x0).
The thread 0x4638 has exited with code 0 (0x0).
The thread 0x45f0 has exited with code 0 (0x0).
The thread 0x444c has exited with code 0 (0x0).
The program '[16452] Chess.exe' has exited with code 0 (0x0).
```



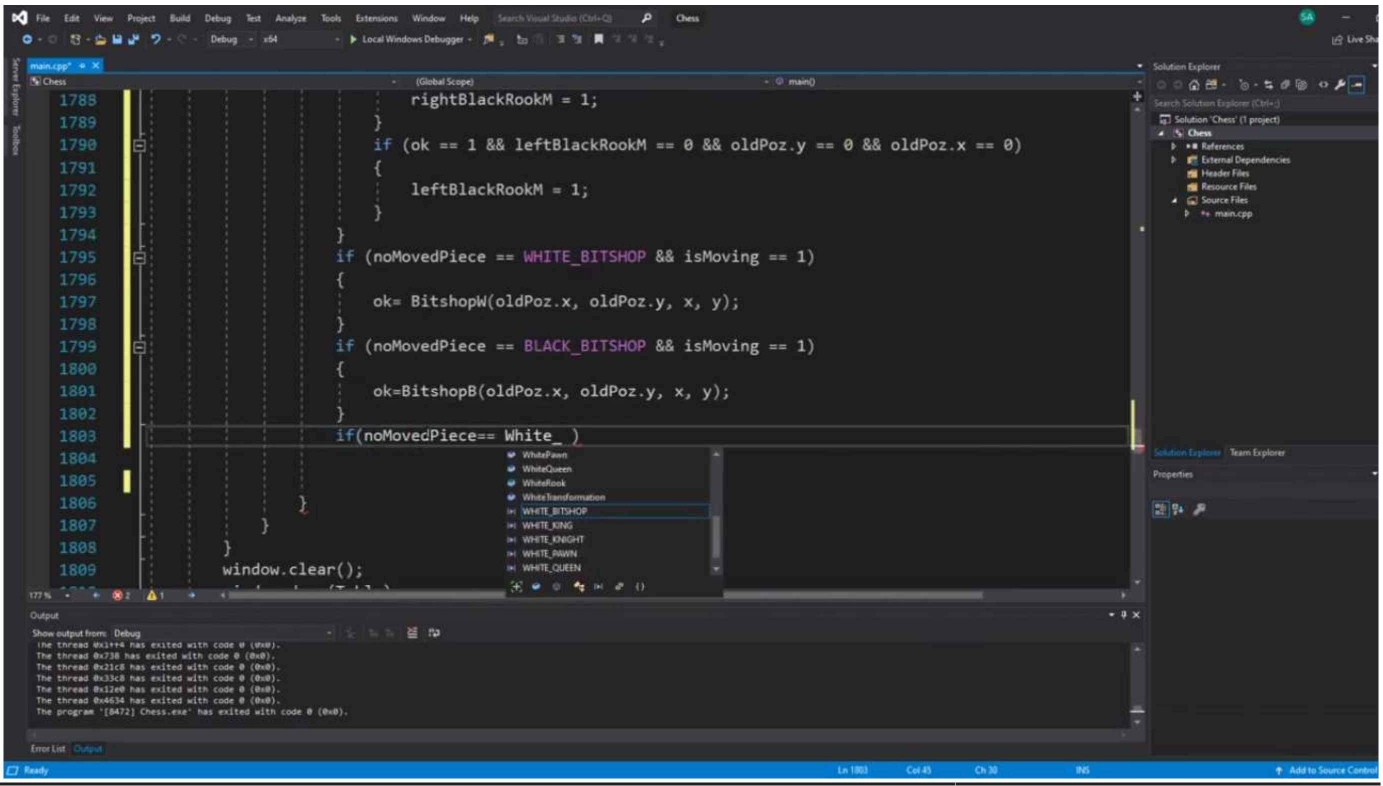
```cpp
    }
    if (oy - 2 >= 0 && ox + 1 <= LENGTH && ny == oy - 2 && nx == ox + 1 && board[ny][nx] >= 0)
    {
        return 1;
    }
    if (oy - 1 >= 0 && ox + 2 <= LENGTH && ny == oy - 1 && nx == ox + 2 && board[ny][nx] >= 0)
    {
        return 1;
    }
    if (oy + 1 <= LENGTH && ox + 2 <= LENGTH && ny == oy + 1 && nx == ox + 2 && board[ny][nx]
    {
        return 1;
    }
    if(oy+2<=LENGTH && ox+1<=LENGTH && ny==oy+2 && nx==ox+1 && b)
}


int main()
{
    RenderWindow window(VideoMode(800, 800), "Chess made by Silvian Achim");
```

```
The thread 0x3000 has exited with code 0 (0x0).
The thread 0x350c has exited with code 0 (0x0).
The thread 0x1114 has exited with code 0 (0x0).
The thread 0x4638 has exited with code 0 (0x0).
The thread 0x45f0 has exited with code 0 (0x0).
The thread 0x444c has exited with code 0 (0x0).
The program '[16452] Chess.exe' has exited with code 0 (0x0).
```



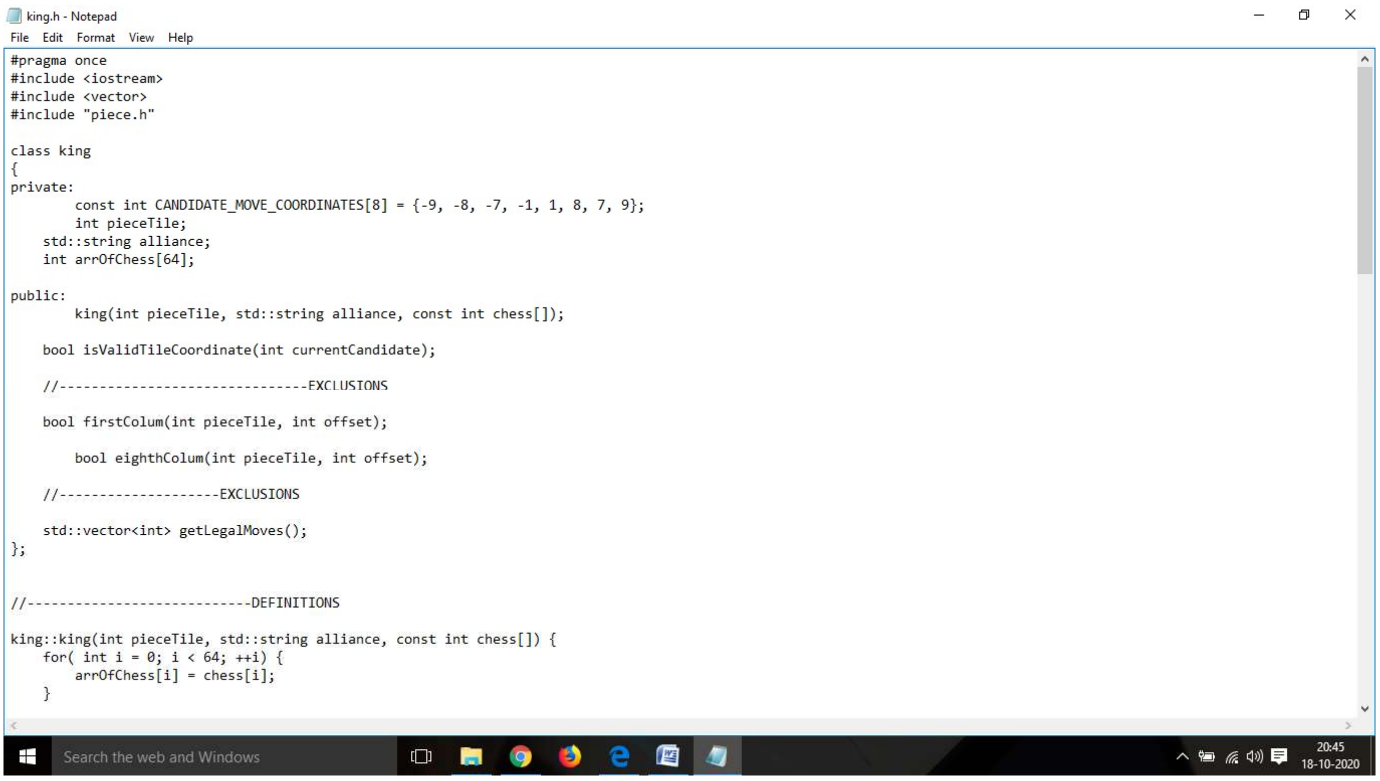
```cpp
int QueenB(int ox, int oy, int nx, int ny){ ... }

int KnightW(int ox, int oy, int nx, int ny){ ... }

int KnightB(int ox, int oy, int nx, int ny){ ... }



int PawnWCheck(int ox, int oy, int kingx, int kingy){ ... }

int RokkWCheck(int ox, int oy, int kingx, int kingy)
{

}




int main()
```

```
The thread 0x3000 has exited with code 0 (0x0).
The thread 0x350c has exited with code 0 (0x0).
The thread 0x1114 has exited with code 0 (0x0).
The thread 0x4638 has exited with code 0 (0x0).
The thread 0x45f0 has exited with code 0 (0x0).
The thread 0x444c has exited with code 0 (0x0).
The program '[16452] Chess.exe' has exited with code 0 (0x0).
```

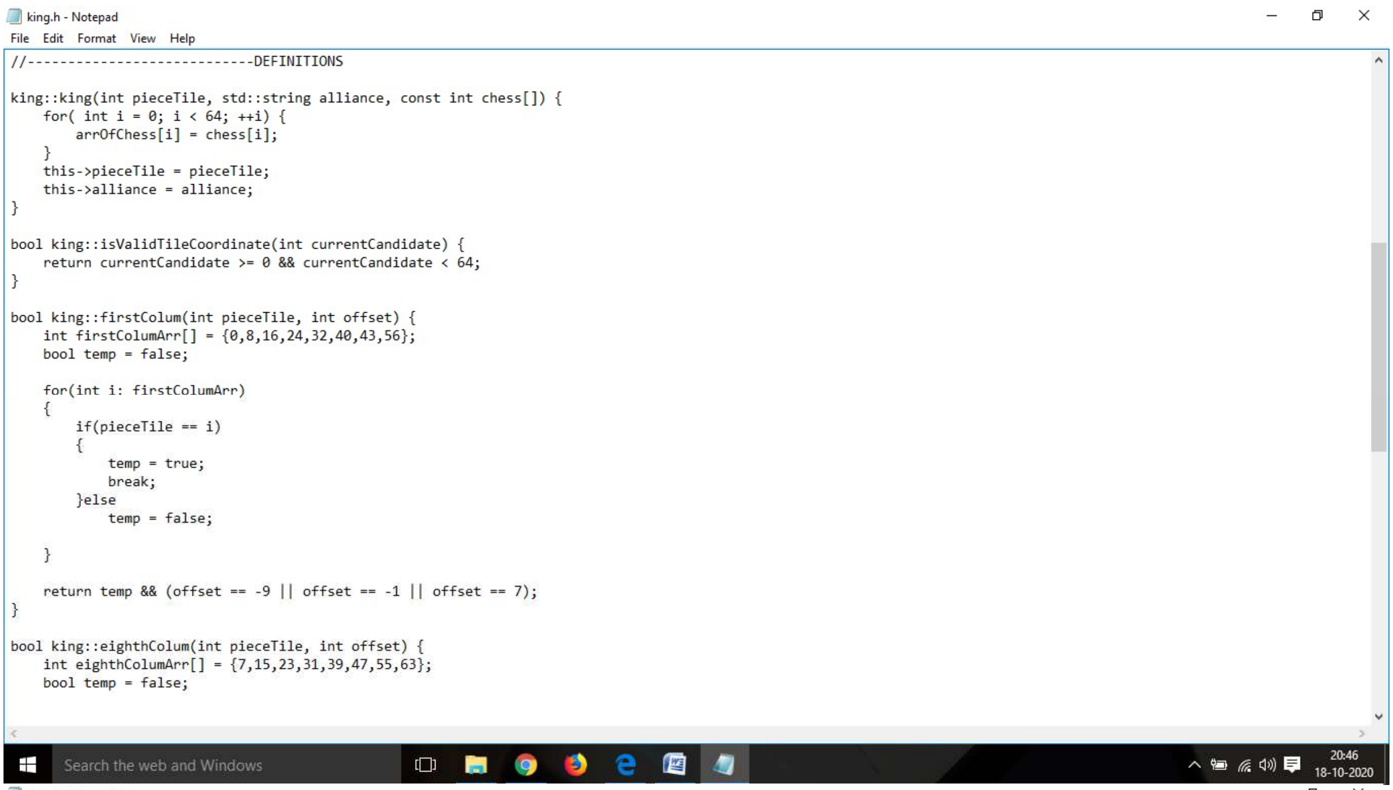


```
                    rightBlackRookM = 1;
                }
                if (ok == 1 && leftBlackRookM == 0 && oldPoz.y == 0 && oldPoz.x == 0)
                {
                    leftBlackRookM = 1;
                }
            }
            if (noMovedPiece == WHITE_BITSHOP && isMoving == 1)
            {
                ok= BitshopW(oldPoz.x, oldPoz.y, x, y);
            }
            if (noMovedPiece == BLACK_BITSHOP && isMoving == 1)
            {
                ok=BitshopB(oldPoz.x, oldPoz.y, x, y);
            }
            if(noMovedPiece== White_ )

        }
    }
}
window.clear();
```

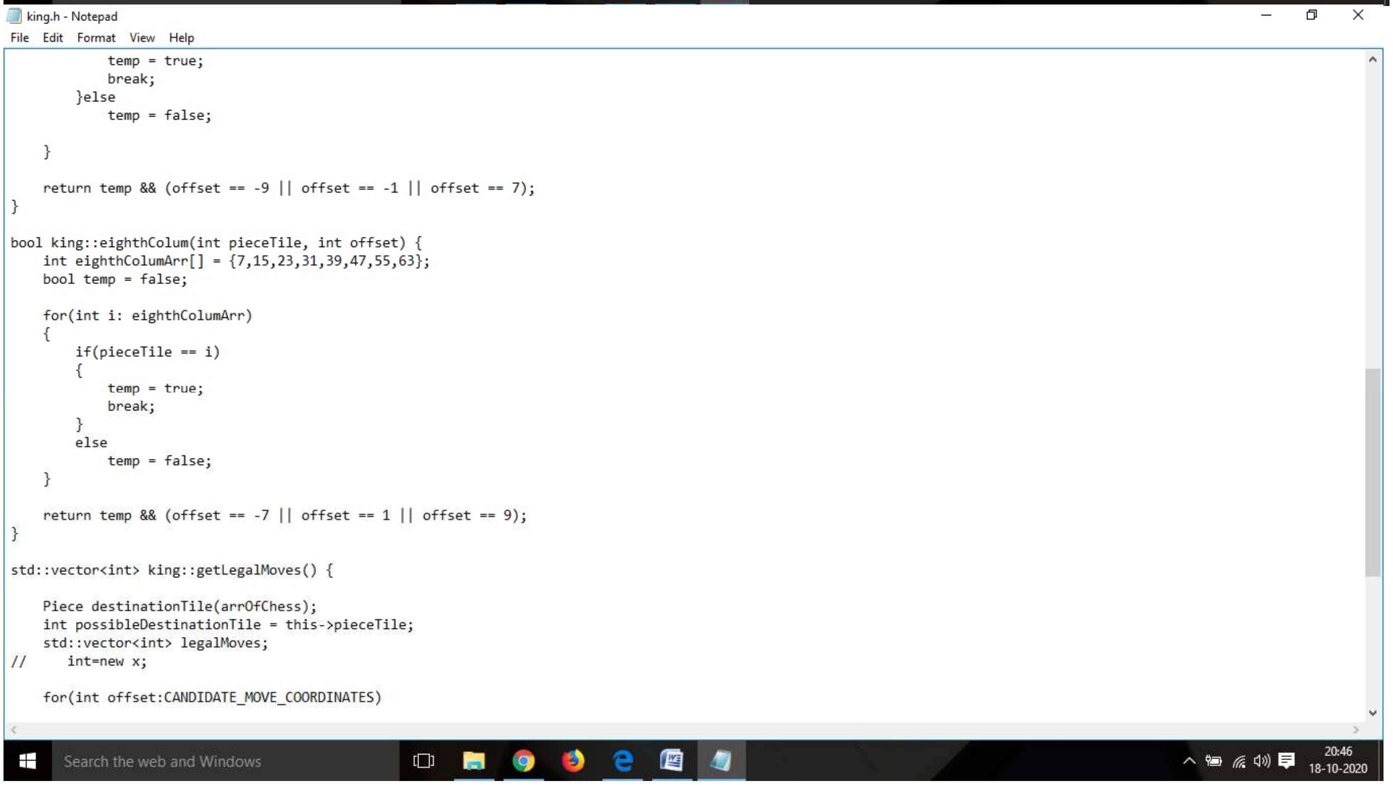
```
The thread 0x1ff4 has exited with code 0 (0x0).
The thread 0x738 has exited with code 0 (0x0).
The thread 0x21c8 has exited with code 0 (0x0).
The thread 0x33c8 has exited with code 0 (0x0).
The thread 0x12e0 has exited with code 0 (0x0).
The thread 0x4634 has exited with code 0 (0x0).
The program '[8472] Chess.exe' has exited with code 0 (0x0).
```



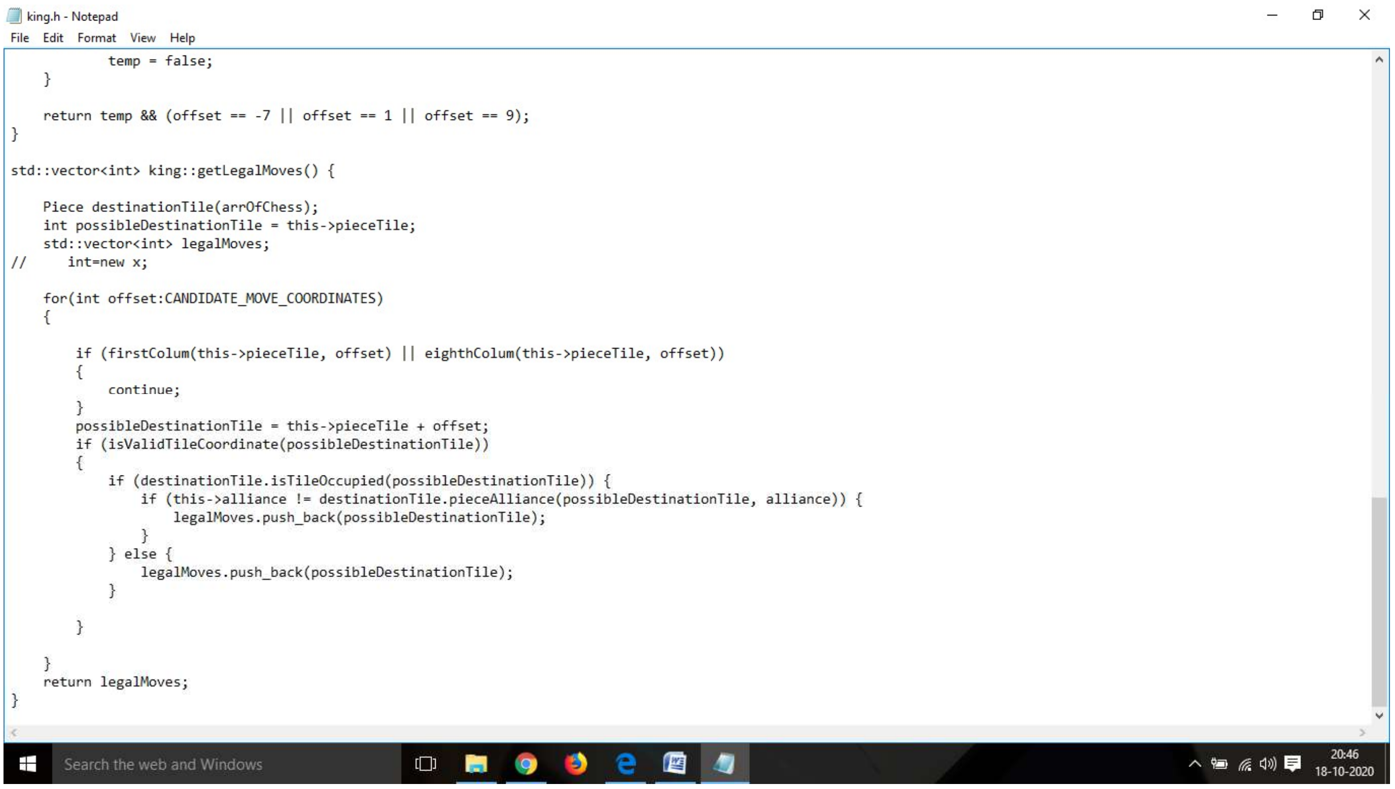
```
#pragma once
#include <iostream>
#include <vector>
#include "piece.h"

class king
{
private:
        const int CANDIDATE_MOVE_COORDINATES[8] = {-9, -8, -7, -1, 1, 8, 7, 9};
        int pieceTile;
    std::string alliance;
    int arrOfChess[64];

public:
        king(int pieceTile, std::string alliance, const int chess[]);

    bool isValidTileCoordinate(int currentCandidate);

    //------------------------------EXCLUSIONS

    bool firstColum(int pieceTile, int offset);

        bool eighthColum(int pieceTile, int offset);

    //--------------------EXCLUSIONS

    std::vector<int> getLegalMoves();
};


//---------------------------DEFINITIONS

king::king(int pieceTile, std::string alliance, const int chess[]) {
    for( int i = 0; i < 64; ++i) {
        arrOfChess[i] = chess[i];
    }
```



```
//---------------------------DEFINITIONS

king::king(int pieceTile, std::string alliance, const int chess[]) {
    for( int i = 0; i < 64; ++i) {
        arrOfChess[i] = chess[i];
    }
    this->pieceTile = pieceTile;
    this->alliance = alliance;
}

bool king::isValidTileCoordinate(int currentCandidate) {
    return currentCandidate >= 0 && currentCandidate < 64;
}

bool king::firstColum(int pieceTile, int offset) {
    int firstColumArr[] = {0,8,16,24,32,40,43,56};
    bool temp = false;

    for(int i: firstColumArr)
    {
        if(pieceTile == i)
        {
            temp = true;
            break;
        }else
            temp = false;

    }

    return temp && (offset == -9 || offset == -1 || offset == 7);
}

bool king::eighthColum(int pieceTile, int offset) {
    int eighthColumArr[] = {7,15,23,31,39,47,55,63};
    bool temp = false;
```



```
            temp = true;
            break;
        }else
            temp = false;

    }

    return temp && (offset == -9 || offset == -1 || offset == 7);
}

bool king::eighthColum(int pieceTile, int offset) {
    int eighthColumArr[] = {7,15,23,31,39,47,55,63};
    bool temp = false;

    for(int i: eighthColumArr)
    {
        if(pieceTile == i)
        {
            temp = true;
            break;
        }
        else
            temp = false;
    }

    return temp && (offset == -7 || offset == 1 || offset == 9);
}

std::vector<int> king::getLegalMoves() {

    Piece destinationTile(arrOfChess);
    int possibleDestinationTile = this->pieceTile;
    std::vector<int> legalMoves;
//     int=new x;

    for(int offset:CANDIDATE_MOVE_COORDINATES)
```



```
            temp = false;
    }

    return temp && (offset == -7 || offset == 1 || offset == 9);
}

std::vector<int> king::getLegalMoves() {

    Piece destinationTile(arrOfChess);
    int possibleDestinationTile = this->pieceTile;
    std::vector<int> legalMoves;
//     int=new x;

    for(int offset:CANDIDATE_MOVE_COORDINATES)
    {

        if (firstColum(this->pieceTile, offset) || eighthColum(this->pieceTile, offset))
        {
            continue;
        }
        possibleDestinationTile = this->pieceTile + offset;
        if (isValidTileCoordinate(possibleDestinationTile))
        {
            if (destinationTile.isTileOccupied(possibleDestinationTile)) {
                if (this->alliance != destinationTile.pieceAlliance(possibleDestinationTile, alliance)) {
                    legalMoves.push_back(possibleDestinationTile);
                }
            } else {
                legalMoves.push_back(possibleDestinationTile);
            }

        }

    }
    return legalMoves;
}
```

## REFERENCES

- **https://www.sfml-dev.org/**
- **https://www.packtpub.com/product/sfml-game-development-by-example/9781785287343**