Computer Graphics Project Work

OpenGL - Chess

Introduction

1.1 Introduction to computer graphics:

Computer graphics started with the display of data on hardcopy plotters and Cathode Ray Tube(CRT) screens soon after the Introduction of computers themselves. It has grown to include the Creation, Storage and Manipulation of Models and Images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural and even conceptual structures, natural phenomenon and so on. Computer graphics today is largely interactive: the user controls the contents, structure, and appearance of objects and of their displayed images by using input devices, such as a keyboard, mouse, or touch-sensitive panel on the screen. Because of the close relationship between the input devices and the display, the handling of such devices is included in the study of computer graphics.

1.2 Uses of Computer Graphics:

Computer graphics is used in many different areas of industry, business, government, education, entertainment, and many other fields, following are some of the applications:

Computer Art:
Using computer graphics we can create fine and commercial art which include animation packages, paint packages. These packages provide facilities for designing object shapes and specifying object motion.

Presentation Graphics:
For the preparation of reports or summarizing the financial, statistical, mathematical, scientific, economic data for research reports, managerial reports, moreover creation of bar graphs, pie charts, time chart, can be done using the tools present in computer graphics.

Graphical User Interface:
The use of pictures, images, icons, pop-up menus, graphical objects helps in creating a user friendly environment where working is easy and pleasant, using computer graphics we can create such an atmosphere where everything can be automated and anyone can get the desired action performed in an easy fashion.

Entertainment and Gaming:
Computer graphics finds a major part of its utility in the movie industry and game industry. Used for creating motion pictures, music video, television shows, cartoon animation films.

In the game industry where focus and interactivity are the key players, computer graphics helps in providing such features in the efficient way.

About the project:

This project implements the game of chess with basic rules and movements in two-dimensional view with OpenGL and C++ programming language.

1.3 Introduction to OpenGL:

Open Graphics Library (OpenGL) is a cross-language (language independent), cross-platform (platform-independent) API for rendering 2D and 3D Vector Graphics(use of polygons to represent image). OpenGL API is designed mostly in hardware.

- **Design :** This API is defined as a set of functions which may be called by the client program. Although functions are similar to those of **C** language but it is language independent.
- **Development :** It is an evolving API and **Khronos Group** regularly releases its new version having some extended feature compare to previous one. GPU vendors may also provide some additional functionality in the form of extension.
- **Associated Libraries :** The earliest version is released with a companion library called OpenGL utility library. But since OpenGL is quite a complex process. So in order to make it easier other library such as OpenGL Utility Toolkit is added which is later superseded by free glut. Later included library were GLEE, GLEW, and gliding.
- **Implementation :** Mesa 3D is an open source implementation of OpenGL. It can do pure software rendering and it may also use hardware acceleration on BSD, Linux, and other platforms by taking advantage of Direct Rendering Infrastructure.

GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms.GLUT is designed for constructing small to medium sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications,

The GLUT library has C, C++ (same as C), FORTRAN, and ADA programming bindings. The GLUT source code distribution is portable to nearly all OpenGL implementations and platforms.

In C or C++, we use the following headers to implement GLUT in the program.

#include<GL/glut.h> or **#include<GLUT/glut.h>**

System Requirements

2.1 Hardware requirements

The hardware requirements are as follows:

Processer: 2 GHz, single core processor or higher

Memory: 512 MB RAM

Storage: 50GB HDD

Display: Color 1980 x 1080 Resolution Display

2.2 Software Requirements

The software requirements are as follows:

Operating System: Linux Distro (Ubuntu 21.4)

Packages and Softwares required: g++ (C++ Compiler), GLUT (freeglut3-dev), OpenGL 3.0

Bash command to compile and execute:

```
g++ chessboard.cpp -o chess -lGL -lGLU –lglut

./chess
```

Algorithm

Step 1: Initialize a GLUT window with resolution 820 * 820 px

Step 2: Display the Chessboard with each square with resolution of 100 * 100 px. Display the label and indexes.

Step 3: Initialize the chess pieces based on the 2D Matrix as follows:

```
// Initial chess pieces positions
int squares[8][8] =
{
        { 4,  3,  2,  5,  6,  2,  3,  4},
        { 1,  1,  1,  1,  1,  1,  1,  1},
        { 0,  0,  0,  0,  0,  0,  0,  0},
        { 0,  0,  0,  0,  0,  0,  0,  0},
        { 0,  0,  0,  0,  0,  0,  0,  0},
        { 0,  0,  0,  0 , 0,  0,  0,  0},
        {-1, -1, -1, -1, -1, -1, -1, -1},
        {-4, -3, -2, -5, -6, -2, -3, -4}
};

//Positive = White pieces
//Negetive = Black pieces
//6 = King
//5 = Queen
//4 = Rook
//3 = Knight
//2 = Bishop
//1 = Pawn
```

Step 4: Input the index of chess piece to move and the index of the square the chess piece has to move into.

Step 5: Check if the move is valid, if valid make the move, else print invalid

Step 6: Redraw the board with updated positions.

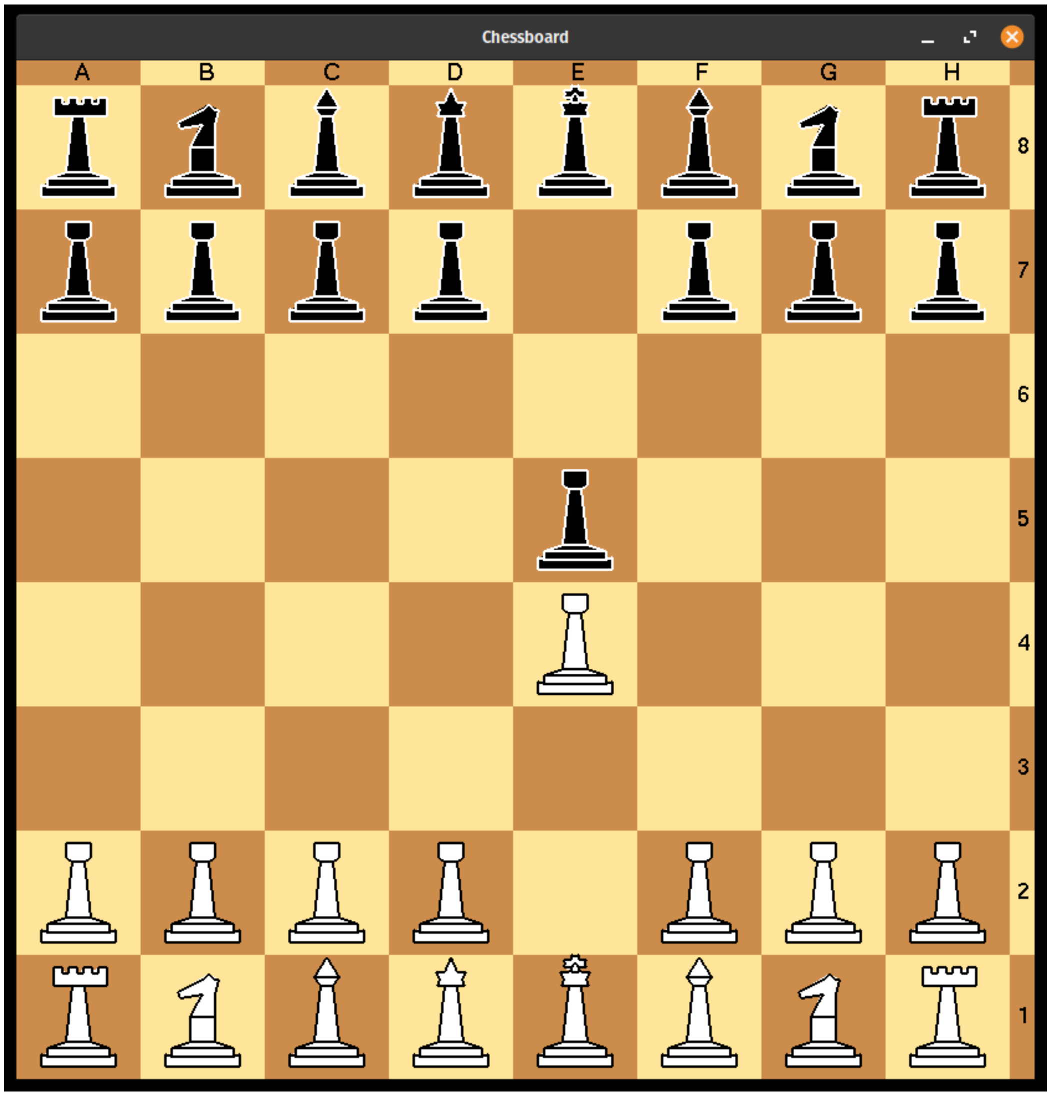
Step 7: Repeat until Step 4 to Step 6 until user exits from the application.

Output

```
                    Current State:


  4(00)  3(01)  2(02)  5(03)  6(04)  2(05)  3(06)  4(07)

  1(10)  1(11)  1(12)  1(13)  1(14)  1(15)  1(16)  1(17)

  0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

  0(30)  0(31)  0(32)  0(33)  0(34)  0(35)  0(36)  0(37)

  0(40)  0(41)  0(42)  0(43)  0(44)  0(45)  0(46)  0(47)

  0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

 -1(60) -1(61) -1(62) -1(63) -1(64) -1(65) -1(66) -1(67)

 -4(70) -3(71) -2(72) -5(73) -6(74) -2(75) -3(76) -4(77)


Move : 1

Enter the label and index of the chess piece:

> e2

Enter the label and index to move the chess piece:

> e4


                 Move: (1, 4) -> (3, 4)


  4(00)  3(01)  2(02)  5(03)  6(04)  2(05)  3(06)  4(07)

  1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

  0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

  0(30)  0(31)  0(32)  0(33)  1(34)  0(35)  0(36)  0(37)

  0(40)  0(41)  0(42)  0(43)  0(44)  0(45)  0(46)  0(47)

  0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

 -1(60) -1(61) -1(62) -1(63) -1(64) -1(65) -1(66) -1(67)

 -4(70) -3(71) -2(72) -5(73) -6(74) -2(75) -3(76) -4(77)
```

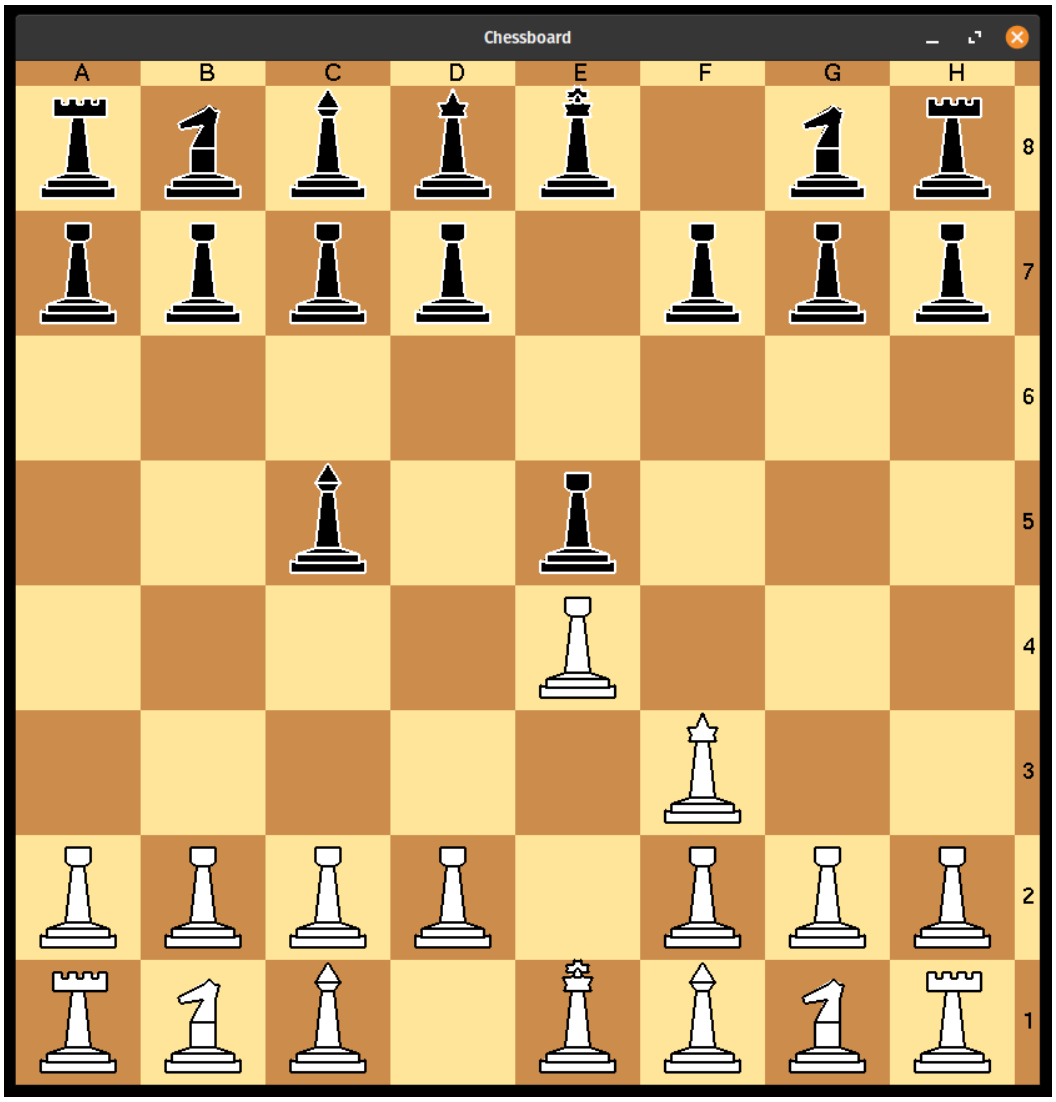
```
                               Current State:


            4(00)  3(01)  2(02)  5(03)  6(04)  2(05)  3(06)  4(07)

            1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

            0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

            0(30)  0(31)  0(32)  0(33)  1(34)  0(35)  0(36)  0(37)

            0(40)  0(41)  0(42)  0(43)  0(44)  0(45)  0(46)  0(47)

            0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

           -1(60) -1(61) -1(62) -1(63) -1(64) -1(65) -1(66) -1(67)

           -4(70) -3(71) -2(72) -5(73) -6(74) -2(75) -3(76) -4(77)

Move : 2

Enter the label and index of the chess piece:

> e7

Enter the label and index to move the chess piece:

> e5


                           Move: (6, 4) -> (4, 4)


            4(00)  3(01)  2(02)  5(03)  6(04)  2(05)  3(06)  4(07)

            1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

            0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

            0(30)  0(31)  0(32)  0(33)  1(34)  0(35)  0(36)  0(37)

            0(40)  0(41)  0(42)  0(43)  -1(44) 0(45)  0(46)  0(47)

            0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

           -1(60) -1(61) -1(62) -1(63) 0(64)  -1(65) -1(66) -1(67)

           -4(70) -3(71) -2(72) -5(73) -6(74) -2(75) -3(76) -4(77)
```

```
Current State:

 4(00)   3(01)   2(02)   5(03)   6(04)   2(05)   3(06)   4(07)

 1(10)   1(11)   1(12)   1(13)   0(14)   1(15)   1(16)   1(17)

 0(20)   0(21)   0(22)   0(23)   0(24)   0(25)   0(26)   0(27)

 0(30)   0(31)   0(32)   0(33)   1(34)   0(35)   0(36)   0(37)

 0(40)   0(41)   0(42)   0(43)  -1(44)   0(45)   0(46)   0(47)

 0(50)   0(51)   0(52)   0(53)   0(54)   0(55)   0(56)   0(57)

-1(60)  -1(61)  -1(62)  -1(63)  0(64)   -1(65)  -1(66)  -1(67)

-4(70)  -3(71)  -2(72)  -5(73)  -6(74)  -2(75)  -3(76)  -4(77)
```



```
Move : 3

Enter the label and index of the chess piece:

> d1

Enter the label and index to move the chess piece:

> f3


                    Move: (0, 3) -> (2, 5)


4(00)  3(01)  2(02)  0(03)  6(04)  2(05)  3(06)  4(07)

1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

0(20)  0(21)  0(22)  0(23)  0(24)  5(25)  0(26)  0(27)

0(30)  0(31)  0(32)  0(33)  1(34)  0(35)  0(36)  0(37)

0(40)  0(41)  0(42)  0(43)  -1(44) 0(45)  0(46)  0(47)

0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

-1(60) -1(61) -1(62) -1(63) 0(64)  -1(65) -1(66) -1(67)

-4(70) -3(71) -2(72) -5(73) -6(74) -2(75) -3(76) -4(77)


                    Current State:


4(00)  3(01)  2(02)  0(03)  6(04)  2(05)  3(06)  4(07)

1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

0(20)  0(21)  0(22)  0(23)  0(24)  5(25)  0(26)  0(27)

0(30)  0(31)  0(32)  0(33)  1(34)  0(35)  0(36)  0(37)

0(40)  0(41)  0(42)  0(43)  -1(44) 0(45)  0(46)  0(47)

0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

-1(60) -1(61) -1(62) -1(63) 0(64)  -1(65) -1(66) -1(67)

-4(70) -3(71) -2(72) -5(73) -6(74) -2(75) -3(76) -4(77)
```

```
Move : 4

Enter the label and index of the chess piece:

> f8

Enter the label and index to move the chess piece:

> c5


                                   Move: (7, 5) -> (4, 2)


                        4(00)  3(01)  2(02)  0(03)  6(04)  2(05)  3(06)  4(07)

                        1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

                        0(20)  0(21)  0(22)  0(23)  0(24)  5(25)  0(26)  0(27)

                        0(30)  0(31)  0(32)  0(33)  1(34)  0(35)  0(36)  0(37)

                        0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)

                        0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

                        -1(60) -1(61) -1(62) -1(63) 0(64)  -1(65) -1(66) -1(67)

                        -4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  -3(76) -4(77)


                                      Current State:


                        4(00)  3(01)  2(02)  0(03)  6(04)  2(05)  3(06)  4(07)

                        1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

                        0(20)  0(21)  0(22)  0(23)  0(24)  5(25)  0(26)  0(27)

                        0(30)  0(31)  0(32)  0(33)  1(34)  0(35)  0(36)  0(37)

                        0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)

                        0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

                        -1(60) -1(61) -1(62) -1(63) 0(64)  -1(65) -1(66) -1(67)

                        -4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  -3(76) -4(77)
```

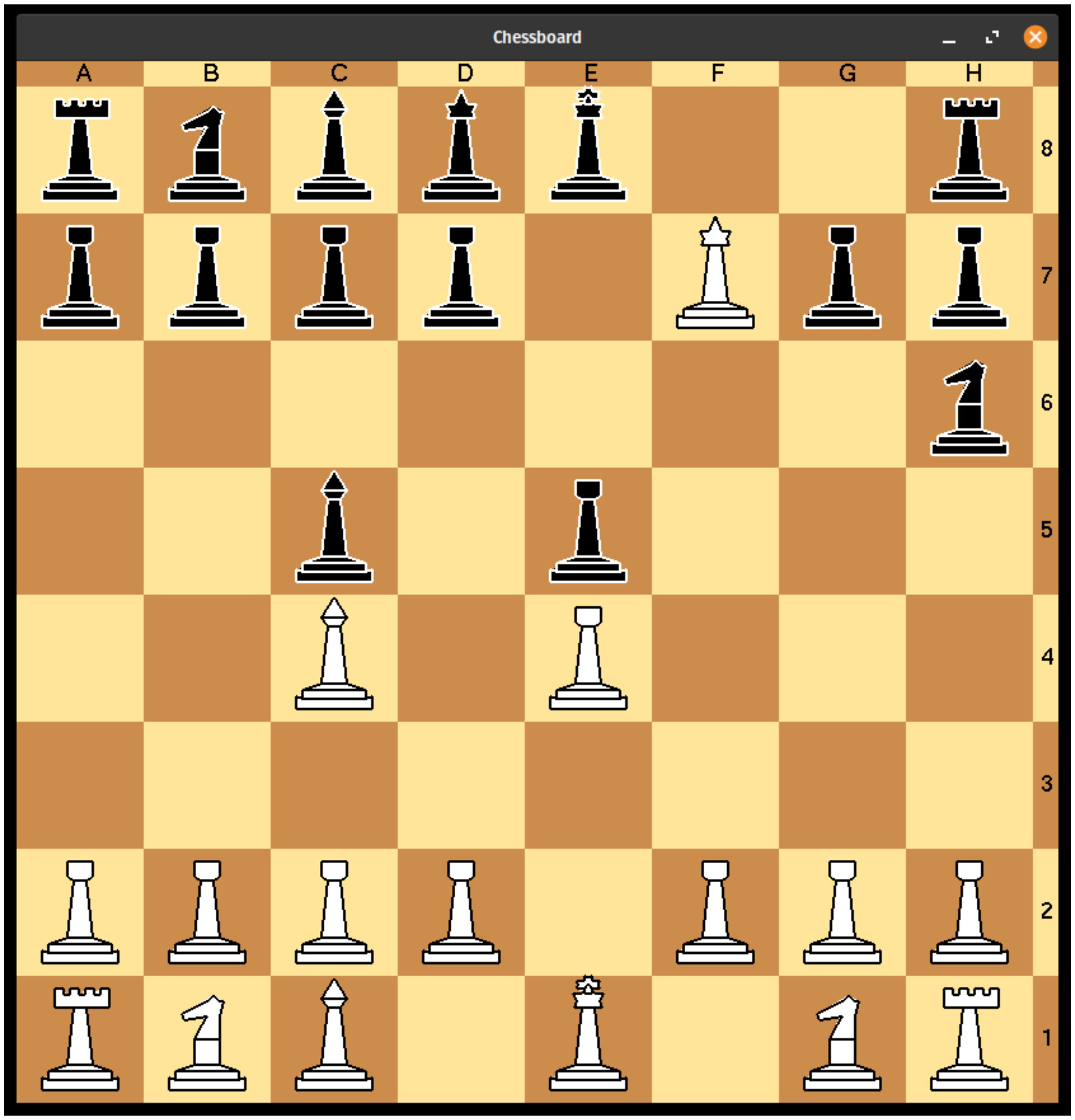


```
Move : 5

Enter the label and index of the chess piece:

> f1

Enter the label and index to move the chess piece:

> c4
```

```
Move: (0, 5) -> (3, 2)

 4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)
 1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)
 0(20)  0(21)  0(22)  0(23)  0(24)  5(25)  0(26)  0(27)
 0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)
 0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)
 0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)
-1(60) -1(61) -1(62) -1(63) 0(64)  -1(65) -1(66) -1(67)
-4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  -3(76) -4(77)

Current State:

 4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)
 1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)
 0(20)  0(21)  0(22)  0(23)  0(24)  5(25)  0(26)  0(27)
 0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)
 0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)
 0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)
-1(60) -1(61) -1(62) -1(63) 0(64)  -1(65) -1(66) -1(67)
-4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  -3(76) -4(77)

Move : 6
Enter the label and index of the chess piece:
> g8
Enter the label and index to move the chess piece:
> h6
```

```
Move: (7, 6) -> (5, 7)

 4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)
 1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)
 0(20)  0(21)  0(22)  0(23)  0(24)  5(25)  0(26)  0(27)
 0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)
 0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)
0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  -3(57)
-1(60) -1(61) -1(62) -1(63) 0(64)  -1(65) -1(66) -1(67)
-4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  0(76)  -4(77)

Current State:

 4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)
 1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)
 0(20)  0(21)  0(22)  0(23)  0(24)  5(25)  0(26)  0(27)
 0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)
 0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)
0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  -3(57)
-1(60) -1(61) -1(62) -1(63) 0(64)  -1(65) -1(66) -1(67)
-4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  0(76)  -4(77)

Move : 7
Enter the label and index of the chess piece:
> f3
Enter the label and index to move the chess piece:
> f7

Move: (2, 5) -> (6, 5)
```

```
4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)

1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)

0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)

0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  -3(57)

-1(60) -1(61) -1(62) -1(63) 0(64)  5(65)  -1(66) -1(67)

-4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  0(76)  -4(77)


                    Current State:


4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)

1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)

0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)

0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  -3(57)

-1(60) -1(61) -1(62) -1(63) 0(64)  5(65)  -1(66) -1(67)

-4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  0(76)  -4(77)
```



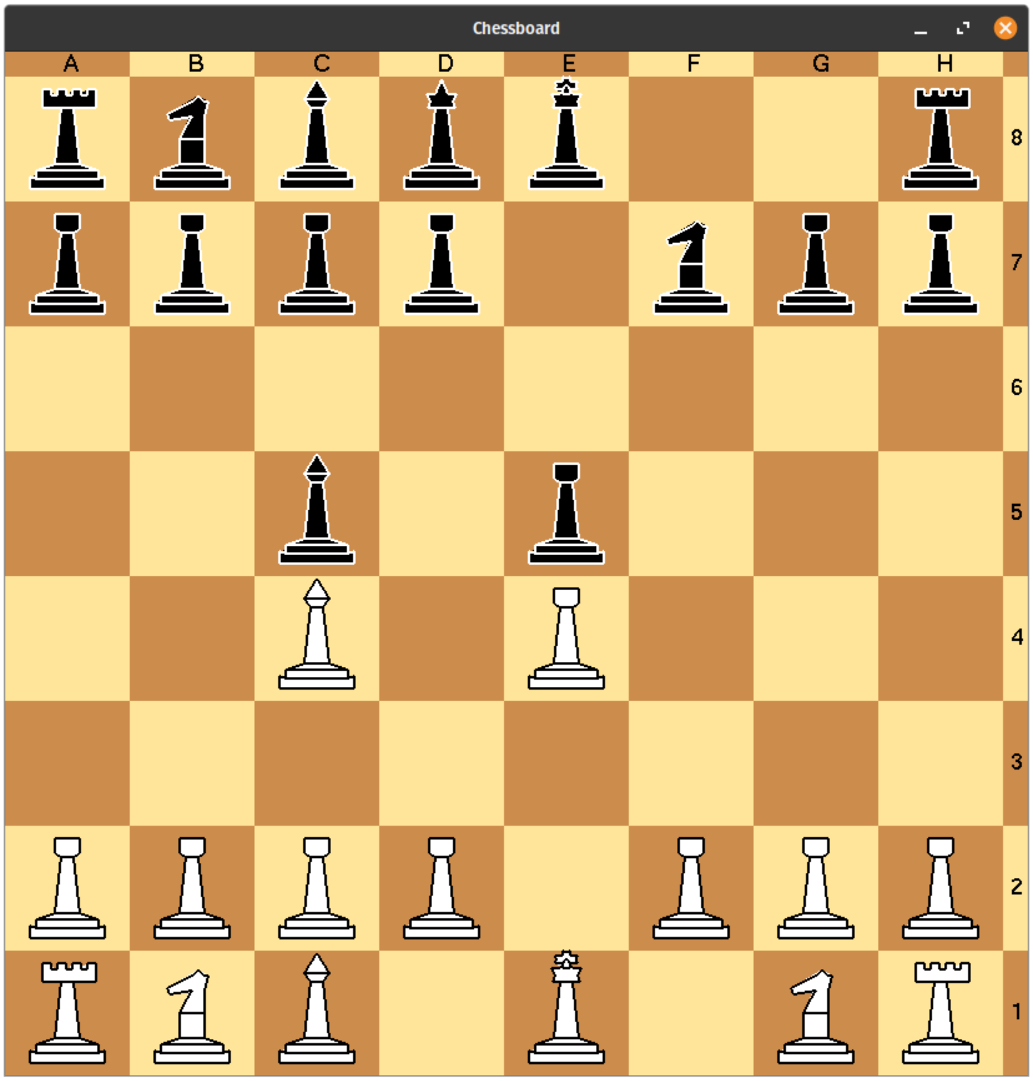
```
Move : 8

Enter the label and index of the chess piece:

> e8

Enter the label and index to move the chess piece:

> f7
```

```
Move: (7, 4) -> (6, 5)
```

Invalid Move, King Under Check

```
                    4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)

                    1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

                    0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

                    0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)

                    0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)

                    0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  -3(57)

                    -1(60) -1(61) -1(62) -1(63) 0(64)  5(65)  -1(66) -1(67)

                    -4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  0(76)  -4(77)


                                  Current State:


                    4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)

                    1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

                    0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

                    0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)

                    0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)

                    0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  -3(57)

                    -1(60) -1(61) -1(62) -1(63) 0(64)  5(65)  -1(66) -1(67)

                    -4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  0(76)  -4(77)


Move : 8

Enter the label and index of the chess piece:

> h6

Enter the label and index to move the chess piece:

> f7


                              Move: (5, 7) -> (6, 5)


                    4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)
```

```
1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)

0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)

0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

-1(60) -1(61) -1(62) -1(63) 0(64)  -3(65) -1(66) -1(67)

-4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  0(76)  -4(77)


                      Current State:


4(00)  3(01)  2(02)  0(03)  6(04)  0(05)  3(06)  4(07)

1(10)  1(11)  1(12)  1(13)  0(14)  1(15)  1(16)  1(17)

0(20)  0(21)  0(22)  0(23)  0(24)  0(25)  0(26)  0(27)

0(30)  0(31)  2(32)  0(33)  1(34)  0(35)  0(36)  0(37)

0(40)  0(41)  -2(42) 0(43)  -1(44) 0(45)  0(46)  0(47)

0(50)  0(51)  0(52)  0(53)  0(54)  0(55)  0(56)  0(57)

-1(60) -1(61) -1(62) -1(63) 0(64)  -3(65) -1(66) -1(67)

-4(70) -3(71) -2(72) -5(73) -6(74) 0(75)  0(76)  -4(77)


Move : 9

Enter the label and index of the chess piece:

> x

Exit
```


Chessboard
A
B
C
D
E
F
G
H
8
7
6
5
4
3
2
1

Conclusion and Future Ideas

The project was successfully implemented Chess with Chess pieces, movements and basic chess rules in OpenGL and C++ Programming Language,

Future Ideas

- In the future we could make the application accept mouse inputs instead of the labels and indexes of the chess piece and square.

- When the chess piece is click on, all the moves the piece could take could be displayed to make the application beginner friendly.

- We could add timer feature that can holds the time remaining for each player.