**Project Report**

**On**

**TIC TAC TOE GAME**

**SUBMITTED BY**

**WILLIAM C FRANCIS**

**REG. NO: \_\_\_\_\_\_\_**

**AS PART OF CLASS XII SYLLABUS OF**

**COMPUTER SCIENCE**



**INFANT JESUS CENTRAL SCHOOL**

**(Affiliation No: 930691)**

**KURAVANKONAM, KOWDIAR P.O TRIVANDRUM-3**

**Phone: 0471-2430477**

**Email: infantjesuskuravankonam@yahoo.co.in**

**INFANT JESUS CENTRAL SCHOOL**

**(Affiliation No: 930691)**

**KURAVANKONAM, KOWDIAR P.O TRIVANDRUM-3**

**Phone: 0471-2430477**

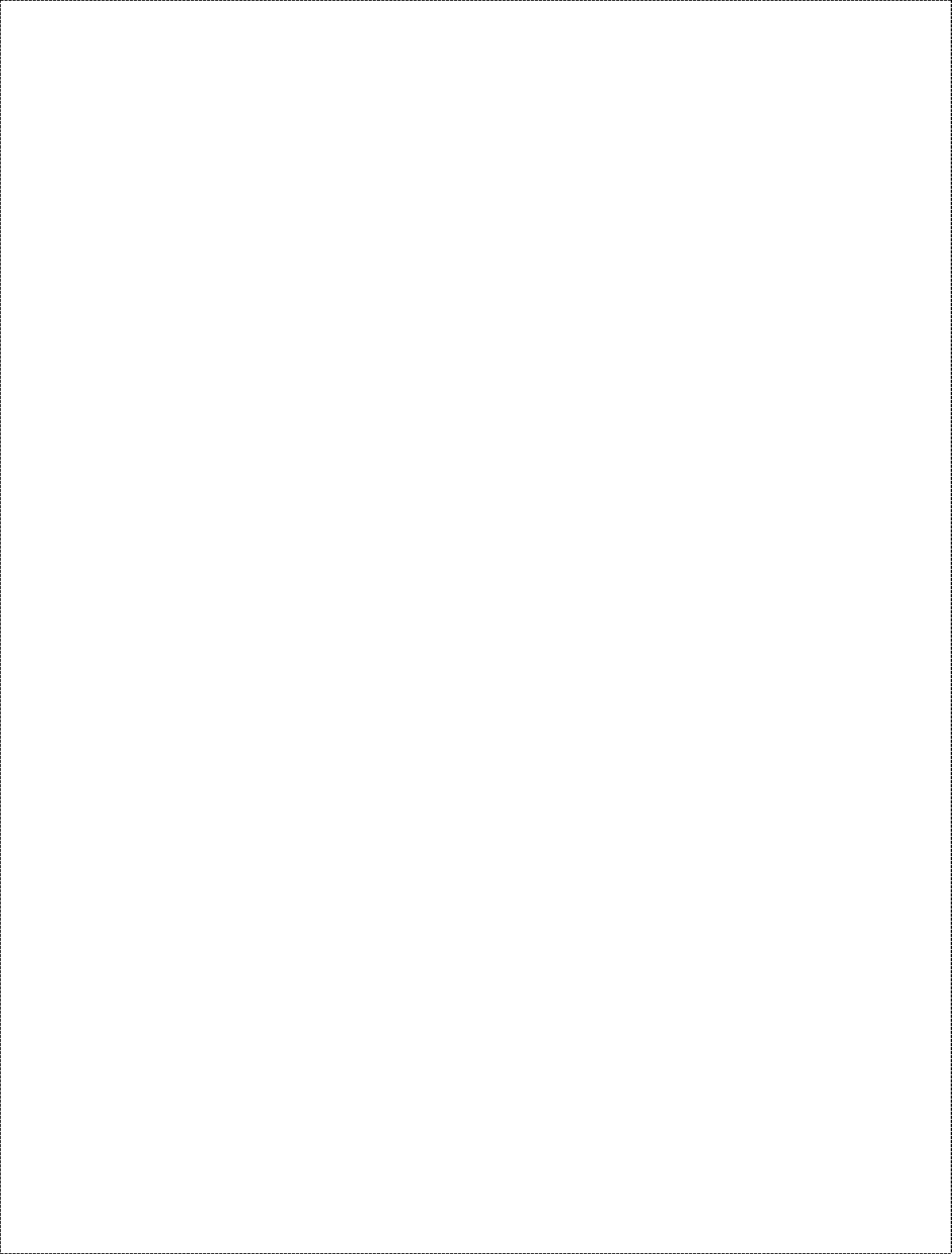
**Email: infantjesuskuravankonam@yahoo.co.in**



**CERTIFICATE**

***It is certified that this project entitled “*Tic Tac Toe Game*” is a bona fide piece of work in Computer Science by* William C Francis*, Class XII, Infant Jesus Central School in accordance with the CBSE syllabus of class XII.***

***Teacher in Charge Examiner***

(seal)

**Acknowledgement**

I am extremely grateful to Mrs. Anju, Teacher of Department of Computer Science for her able guidance and useful suggestions, which helped me in completing the project work, in time.

I would also like to thank all the teaching and non-teaching staff of Computer Science department who helped me directly or indirectly in the completion of this project .

Finally, yet importantly, I would like to express my heartfelt thanks to my beloved parents for their blessings, my friends for their help and wishes for the successful completion of this project.

**William C Francis**

**Synopsis**

Through this project we made use of C++ to convert a paper-pencil game called “**Tic Tac Toe**” into a computer based game using the basic concepts of Artificial Intelligence.

Tic Tac Toe (also known as Noughts and crosses or Xs and Os) is a paper-and-pencil game played since the ancient Roman Empire. This game is played between two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row, wins the game.

Two game modes are implemented into the program, namely, Single player mode (player vs computer) and Multi player mode (player vs player). The program also stores and displays the score statistics/match history of each game played.

.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sl no** | **CONTENT** | **PAGE NO** |
| 1. | Introduction | 1 |
|  | * About C++ |  |
|  | * Data File Handling | 2 |
|  | * Object Oriented Programming | 3 |
|  | * Drawbacks of previous versions | 4 |
| 2. | Aim of the Project | 5 |
| 3. | System Requirements | 6 |
|  | * Hardware |  |
|  | * Software |  |
| 4. | Data Dictionary | 7 |
|  | * Header Files |  |
|  | * Functions | 9 |
|  | * Data Files | 12 |
| 5. | Source Code | 13 |
| 6. | Output Screen Layout | 40 |
| 7. | Conclusion | 52 |
| 8. | Bibliography | 53 |

**Introduction**

**About C++**

C++ is the modified version of C to support Object Oriented Programming. The C++ programming was developed at AT&T Bell laboratories in the early 1980’s by Bjaren Stroustrup. He found ‘C’ lacking for simulations and decided to extend the language by adding features from his favorite language, Simula 67. Simula 67 was one of the earliest object oriented language. Bjaren Stroustrup called it ”C with classes” originally. The name C++ was coined by Rick Mascitti. ”++” is the C increment operator. Ever since its birth, C++ evolved to cope with problems encountered by the users and through discussion at AT&T. The Latest C++ standard documents were issued by ANSI/ISO in the year 2003.

The striking features of C++ are:

* Data File Handling
* Object Oriented Programming
* Reusability of code
* Data Encapsulation
* Polymorphism

**Data File Handling**

Files are a means to store data in a storage device. C++ file handling provides a mechanism to store output of a program in a file and read from a file on the disk. So far, we have been using <iostream.h> header file which provide functions cin and cout to take input from console and write output to a console respectively. Data File Handling introduces one more header file <fstream.h> which provides data types or classes (ifstream, ofstream, fstream) to read from a file and write to a file.

Some of the function of Data File Handling are listed in the table below:

|  |  |
| --- | --- |
| **Function** | **Description** |
| seekg( ) | Moves the **get** pointer to a specific location in the file |
| seekp( ) | Moves the **put** pointer to a specific location in the file |
| tellg( ) | Returns the position of **get** pointer |
| tellp( ) | Returns the position of **put** pointer |

**Object Oriented Programming**

The core of the pure object-oriented programming is to create an object, in code, that has certain properties and methods. While designing C++ modules, we try to see whole world in the form of objects. For example a car is an object which has certain properties such as color, number of doors, and the like. It also has certain methods such as accelerate, brake, and so on.

There are a few principle concepts that form the foundation of object-oriented programming:

***Object*** - This is the basic unit of object oriented programming. That is both data and function that operate on data are bundled as a unit called as object.

***Class*** - When we define a class, you define a blueprint for an object. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

***Abstraction -*** Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

***Encapsulation -*** Encapsulation is placing the data and the functions that work on that data in the same place.

***Inheritance -*** One of the most useful aspects of object-oriented programming is code reusability. As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.

***Polymorphism -*** The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism.

**Drawbacks of previous versions**

The previous versions are not user friendly and reliable. In those versions, only a player vs player platform was available. It was not possible for users to play the game without a human opponent. Furthermore, user will not be able to store game data or access help in playing if he/she wants to. This makes the existing system non user-friendly.

**Aim of the Project**

The aim of the project is to make a virtual form of the world famous game of Tic Tac Toe which is user-friendly and more attractive. Graphics have been implemented into the program to make the game more fresh and interactive. All efforts were taken to make the game simple and usable for the people of all ages.

**System Requirements**

**Hardware**

Processor : Pentium II or above

Ram : 256 MB or above

Hard disk drive : 20 GB or Above

Monitor : SVGA

Operating Frequency : 500MHz

GPU : 512 MB

**Software**

Operating System : MS Dos/Windows 95

Language : C++

Compiler : Borland C++/Turbo C++

**Data Dictionary**

**Header Files**

**iostream.h –** This header file is used in the program for calling the basic input and output functions namely cin and cout.

**conio.h** – This header file is used to enable functions such as window(), clrscr(), gotoxy(), getch(), textcolor(), textbackground().

**iomanip.h** – This header file provides parametric manipulators. The function included in the program from this header file is setprecision().

**stdio.h** – This header file is used in the program to wait for an input from player before proceeding to the next screen using the function getchar(). It is also used for accepting values into a character array using gets() function.

**stdlib.h –** This header file is the Standard General Utilities Library. It is included in the program to invoke the functions such as system(“pause”), exit() and system(“cls”).

**string.h -** This header introduces string types, character traits and a set of converting functions. This header file is used in the program to:

1. copy a character array using strcpy(),
2. compare two character arrays using strcmpi()
3. find the length of a character array using strlen()

**fstream.h** – Header providing file stream classes for initializing an object of fstream.

**graphics.h –** All the graphics functions used in the program such as settextstyle(), setbkcolor(), line(), getmaxx(), getmaxy() etc. are included in this header file.

**dos.h** - It contains functions for handling interrupts, producing sound, date and time functions etc. The functions used in the program from this header file are delay() and sleep().

**Functions**

1. Function name: **main**

Return type: void

Use of function: To display welcome graphics, display menu and to redirect to another function according to users choice from menu.

1. Function name: **computerAI**

Return type: void

Use of function: The computer decides its move using Artificial Intelligence.

1. Function name: **singleplayer**

Return type: void

Use of function: This function initializes single player mode of the game. It inputs the player name, player’s choice and lead the user to the game board where the user will play head-on with the computer. When the game is over, it prompts the user to choose whether to view their score statistics. Then it prompts the user whether to play the game again or not and acts accordingly.

1. Function name: **multiplayer**

Return type: void

Use of function: This function initializes Multi player mode of the game. It inputs the name of two players, player’s choice and lead the users to the game board where the users will play head-on with each other. When the game is over, it prompts the users to choose whether to view their score statistics. Then it prompts the users whether to play the game again or not and acts accordingly.

1. Function name: **playerinput**

Return type: void

Use of function: To check validity of player input and insert if valid.

1. Function name: **boardsingle**

Return type: void

Use of function: Displays game board for single player mode

1. Function name: **boardmulti**

Return type: void

Use of function: Displays game board for multi player mode

1. Function name: **checkwinsingle**

Return type: integer

Use of function: This function checks the game status of single player mode and return a value to the single player function accordingly. It returns ‘2’ when the computer wins, ‘1’ when the player wins, ‘0’ when the game is a draw and ‘-1’ when the game is still in progress.

1. Function name: **checkwinmulti**

Return type: integer

Use of function: This function checks the game status of multi player mode and return a value to the multiplayer function accordingly. It returns ‘1’ when the current player wins, ‘0’ when the game is a draw and ‘-1’ when the game is still in progress.

**Data Files**

*Files for Multi Player mode*

1. **Winners.dat –** File that stores the name of players who have

won the game.

1. **Losers.dat –** File that stores the name of players who have lost

the game.

1. **Draw.dat –** File that stores the name of players who have

encountered a draw match.

*Files for Single Player mode*

1. **Singlewinners.dat** – File that stores the name of players who have won the game against the computer AI.
2. **Singlelosers.dat** - File that stores the name of players who have lost the game against the computer AI.
3. **SingleDraws.dat –** File that stores the name of players who have encountered a draw match against the computer AI.

**Source Code**

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

***File name****: TICTACTOE.cpp*

***Name of the program****: TIC TAC TOE Game.*

***Programmers****: William C. Francis*

*Anantha Krishnan A. S.*

*Mukundan A.*

***Description****:*

*Tic Tac Toe (also known as Noughts and crosses or Xs and Os) is a paper-and-pencil game played since the ancient Roman Empire. This game is played between two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row, wins the game.*

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

*//header files*

#include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<iomanip.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

#include<fstream.h>

#include<graphics.h>

#include<dos.h>

void computerAI();

void singleplayer();

void playerinput();

void multiplayer();

void boardsingle();

void boardmulti();

int checkwinsingle();

int checkwinmulti();

char square[10]={'0','1','2','3','4','5','6','7','8','9’}; *//Array representing grids of board*

void main()

{

window(10,10,400,400); *//creates window*

clrscr();

textcolor(15);  *//changes text colour to white*

gotoxy(40,50);

int x,y,i;

int g=DETECT, d;

initgraph(&g, &d,"C:\\TC\\BGI");

cleardevice();

x=170;

settextstyle(DEFAULT\_FONT,HORIZ\_DIR,2); *// changes font to DEFAULT*

setbkcolor(9);

setcolor(13);

outtextxy(150,180,"\n LOADING, PLEASE WAIT \n");

setcolor(10);

for(i=0;i<350;++i) *//Animating progress bar*

{

delay(11);

line(x,220,x,240);

x++;

}

cleardevice();

x=getmaxx()/2; *//control goes to centre of screen*

y=getmaxy()/2;

setcolor(10);

settextstyle(TRIPLEX\_FONT, HORIZ\_DIR, 5); *//changes font to TRIPLEX*

settextjustify(x,y);

outtextxy(x-70,y-90,"WELCOME"); *//Welcome screen*

delay(200);

outtextxy(x-10,y-40,"TO\n");

delay(200);

setcolor(14);

settextstyle(DEFAULT\_FONT, HORIZ\_DIR, 5);

outtextxy(x-200,y+40,"TIC TAC TOE");

cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t ";

setcolor(15);

system("pause");

closegraph(); *//closes graphics functions*

system("cls");

textcolor(15); *//changes text colour to white*

gotoxy(40,50);

window(1,1,400,400);

textbackground(CYAN); *//changes background colour to CYAN*

clrscr();

menu: *//returns to menu after Help window*

system("cls");

window(10,10,400,400);

textbackground(11);

clrscr();

textcolor(15);

gotoxy(40,50);

cout<<"\t\n\n\n\n\n";

cout<<"\t\t\t \_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\n\n"; *//Menu screen*

cout<<"\t\t\t 1. Single Player\n\n";

cout<<"\t\t\t 2. Multi Player\n\n";

cout<<"\t\t\t 3. Help\n\n";

cout<<"\t\t\t 4. Exit Game\n";

cout<<"\t\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

cout<<"\n\n\n\t\t\t Enter your choice: ";

int ch;

cin>>ch; *//Accepts menu choice*

if(ch==2) *//redirects to Multi Player function*

{

system("cls");

window(10,10,400,400);

textbackground(CYAN);

clrscr();

textcolor(15);

gotoxy(40,50);

multiplayer();

}

else if(ch==1) *//Redirects to Single Player*

{

system("cls");

window(10,10,400,400);

textbackground(CYAN);

clrscr();

textcolor(15);

gotoxy(40,50);

singleplayer();

}

else if(ch==4) *//choice to exit from game*

{

system("cls");

textcolor(15); *//Displays Exit Screen*

gotoxy(40,50);

window(1,1,400,400);

textbackground(15);

clrscr();

cout<<"\n\n\n\n\n\n\n\n\n\n\n\t\t\t ================\n";

cout<<"\t\t\t = Exiting Game =\n";

cout<<"\t\t\t ================\n";

getch();

exit(0); *//Exits from game*

}

else if(ch==3) *//opens Help window*

{

system("cls");

window(10,10,400,400);

textbackground(CYAN);

clrscr();

textcolor(15);

gotoxy(40,50);

cout<<"\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_HELP\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n";

cout<<" \* Select the game mode from menu using a number (1-4).\n\n";

cout<<" \* After selecting your desired game mode, enter your name to

keep your\n details intact and view them every time

you play.\n\n";

cout<<" \* You will be redirected to the game board where you can play

the game.\n\n";

cout<<" \* The object tive of the game is to complete a row or column or

a diagonal\n with your mark (X or O).\n\n";

cout<<" \* The game will be declared as a draw if neither of the two

players win.\n\n";

cout<<" \* Choose a position (1-9) on the game board to place your mark

(X or O).\n\n";

cout<<" \* The game provides with an option to view your score statistics

of all\ the games you have played.\n\n";

cout<<" \* The score statistics show your total number of games played, total\n number of wins, looses and draws and also your win

percentage.\n\n";

cout<<" \* Choose whether to play again or to quit the game when

prompted.\n";

getch();

goto menu; *//returns to menu after Help*

}

}

void multiplayer() *//Function for Multi Player*

{

char re;

char player1[50];

cout<<"\n\n\n\t\t\tPlayer 1, Please enter your name: ";

cin>>player1;

char player2[50];

cout<<"\n\n\n\n\t\t\tPlayer 2, Please enter your name: ";

cin>>player2;

do

{

for(int j=0;j<10;j++) *//Resets board*

{

char k[10]={'0','1','2','3','4','5','6','7','8','9'};

square[j]=k[j];

}

clrscr();

int player=1;

int i,choice;

char mark;

do

{

boardmulti();

player=(player%2)?1:2; *//checking players turn*

mark=(player == 1) ? 'X' : 'O'; *//assigns mark to corresponding player*

cout << "\n\t\tPlayer "<<player<<", Please choose a grid to place

("<<mark<<"): ";

cin >> choice; *//accepts players choice*

cout<<"\n";

if (choice == 1 && square[1] == '1')

square[1] = mark;

else if (choice == 2 && square[2] == '2')

square[2] = mark;

else if (choice == 3 && square[3] == '3')

square[3] = mark;

else if (choice == 4 && square[4] == '4')

square[4] = mark;

else if (choice == 5 && square[5] == '5')

square[5] = mark;

else if (choice == 6 && square[6] == '6')

square[6] = mark;

else if (choice == 7 && square[7] == '7')

square[7] = mark;

else if (choice == 8 && square[8] == '8')

square[8] = mark;

else if (choice == 9 && square[9] == '9')

square[9] = mark;

else

{

cout<<"\t\t\t\tInvalid move! ";

player--; *//Resets players turn after invalid move*

getch();

}

i=checkwinmulti();

player++;

}while(i==-1); *//Loop functions till completion of the game*

boardmulti(); *//Displays Multi Player board*

if(i==1||i==0)

{

char winner[50],loser[50];

if(i==1)

{

int winnernum= --player;

if(winnernum==1) *//Player 1 wins*

{

cout<<"\a\t\t\t"<<player1<<" wins against "<<player2<<"\n";

strcpy(winner, player1);

strcpy(loser, player2);

getchar();

}

else if(winnernum==2) *//Player 2 wins*

{

cout<<"\a\t\t\t"<<player2<<" wins against "<<player1<<"\n";

strcpy(winner, player2);

strcpy(loser, player1);

getchar();

}

}

else if(i==0) *//Game is draw*

{

cout<<"\a\t\t\t\t Game Draw\n";

strcpy(winner, player2);

strcpy(loser, player1);

getchar();

}

int winnerloses=0, loserloses=0;

float loserwins=0.0, winnerwins=0.0, winnertotal=0.0,losertotal=0.0, winnerpercent=0.0, loserpercent=0.0;

char readname[50];

fstream victory;

victory.open("Winners.dat",ios::in|ios::out|ios::app); *//Opens file containing*  *names of all winners*

if(i==1)

{

victory.seekg(0,ios::end);

victory<<winner; *//Inserts name of winner into Winners.dat*

victory<<' ';

victory.seekg(ios::beg);

}

do

{

victory>>readname;

if(strcmpi(readname,winner)==0)

winnerwins++; *//counts number of wins of the winner*

if(strcmpi(readname,loser)==0)

loserwins++; *//counts number of wins of the loser*

}while(victory.eof()==0);

victory.close();

fstream defeat;

defeat.open("Losers.dat",ios::in|ios::out|ios::app); *//Opens file containing* *names of all losers*

if(i==1)

{

defeat.seekg(0,ios::end);

defeat<<loser; *//Inserts name of loser into Losers.dat*

defeat<<' ';

defeat.seekg(0,ios::beg);

}

do

{

defeat>>readname;

if(strcmpi(readname,winner)==0)

winnerloses++; *//Counts number of loses of the winner*

if(strcmpi(readname,loser)==0)

loserloses++; *//Counts number of loses of the loser*

}while(defeat.eof()==0);

defeat.close();

int winnerdraws=0,loserdraws=0;

fstream draw;

draw.open("Draw.dat",ios::in|ios::out|ios::app); *//Opens file*

*containing* *details of draw matches*

draw.seekg(0,ios::end);

if(i==0)

{

draw<<winner; *//Writes the names of both the players in the draw*

*file*

draw<<' ';

draw<<loser;

draw<<' ';

}

draw.seekg(0,ios::beg);

do

{

draw>>readname;

if(strcmpi(readname,winner)==0)

winnerdraws++; *//Counts number of draws of winner*

if(strcmpi(readname,loser)==0)

loserdraws++; *//Counts number of draws of loser*

}while(draw.eof()==0);

winnertotal=winnerwins+winnerloses+winnerdraws; *//Calculates scores*

losertotal=loserwins+loserloses+loserdraws;

winnerpercent=(winnerwins/winnertotal)\*100;

loserpercent=(loserwins/losertotal)\*100;

char choice;

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(CYAN);

clrscr();

cout<<"\n\n\n\n\n "<<winner<<", Do you like to view your Score

Statistics? (y/n): ";

cin>>choice; *//Choice for viewing score of winner*

if(choice=='y'||choice=='Y')

{

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(6);

clrscr();

cout<<"\n\n\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_Score Statistics of "<<winner<<"\_\_\_\_\_\_\_\_\_\_\_\_\_"; *//Displays score of winner*

cout<<"\n\n\t\t Total Games Played : "<<winnertotal;

cout<<"\n\t\t Total number of Victories : "<<winnerwins;

cout<<"\n\t\t Total number of Defeats : "<<winnerloses;

cout<<"\n\t\t Total number of Draws : "<<winnerdraws;

cout<<"\n\t\t Win Percentage : "<<setprecision(1)<<winnerpercent<<"%";

cout<<"\n\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_";

getchar();

}

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(CYAN);

clrscr();

cout<<"\n\n\n\n\n "<<loser<<", Do you like to view your Score Statistics? (y/n): ";

cin>>choice; *//Choice for viewing score of loser*

if(choice=='y'||choice=='Y')

{

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(6);

clrscr();

cout<<"\n\n\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_Score Statistics of "<<loser<<"\_\_\_\_\_\_\_\_\_\_\_\_\_";

cout<<"\n\n\t\t Total Games Played : "<<losertotal*; //Displays*

*score of* *loser*

cout<<"\n\t\t Total number of Victories : "<<loserwins;

cout<<"\n\t\t Total number of Defeats : "<<loserloses;

cout<<"\n\t\t Total number of Draws : "<<loserdraws;

cout<<"\n\t\t Win Percentage :

"<<setprecision(1)<<loserpercent<<"%";

cout<<"\n\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

getchar();

}

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(CYAN);

clrscr();

cout<<"\n\n\n\n\n Do you like to play again? (y/n): ";

cin>>re; //Choice for replay

system("cls");

}

if(re=='N'||re=='n') *//Displays exiting screen*

{

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(15);

clrscr();

cout<<"\n\n\n\n\n\n\n\n\n\n\n\t\t\t ================\n";

cout<<"\t\t\t = Exiting Game =\n";

cout<<"\t\t\t ================\n";

getch();

}

}while(re=='y'||re=='Y');

}

void singleplayer() *//Function for single player mode*

{

char re;

char human[50];

cout<<"\n\n\n\t\t\tPlayer, Please enter your name: ";

gets(human); *//Inputs Name of player*

do

{

for(int j=0;j<10;j++) *//Resets Board*

{

char k[10]={'0','1','2','3','4','5','6','7','8','9'};

square[j]=k[j];

}

clrscr();

char nametaken[50];

int i=-1,randompick;

int player=1;

do

{

if (player==1)

{

playerinput(); *//Accepting players choice*

player=2;

boardsingle(); *//Displays board of single player*

}

else if(player==2)

{

cout<<"\t\t\tComputer is deciding its move";

for(int s=0;s<2;s++) *//Loop for delay*

sleep(s);

computerAI(); *//Computer thinks and inputs a logical move*

player=1;

boardsingle(); *//Displays Single player board*

}

i=checkwinsingle();

}while(i==-1);

if(i==1)

cout<<"\a\t\t\t "<<human<<" wins against Computer"<<"\n";

else if(i==2)

cout<<"\a\t\t\t Computer wins against "<<human<<"\n";

else if(i==0)

cout<<"\a\t\t\t\t Game draw\n";

getchar();

float win=0.0, lose=0.0;

fstream playerwin;

playerwin.open("Singlewinners.dat",ios::in|ios::out|ios::app); *//opens file*

*containing names of all winners of single player mode*

playerwin.seekg(0,ios::end);

if(i==1)

{

playerwin<<human;*//Writes the name of player to the Singlewinners.dat file*

playerwin<<' ';

}

playerwin.seekg(ios::beg);

do

{

playerwin>>nametaken;

if(strcmpi(nametaken,human)==0)

win++; *//counts the number of wins of the player*

}while(playerwin.eof()==0);

playerwin.close();

fstream playerlose;

playerlose.open("Singlelosers.dat",ios::in|ios::out|ios::app); *//opens file*

*containing names of all losers of single player mode*

playerlose.seekg(0,ios::end);

if(i==2)

{

playerlose<<human; *//writes the name of player to Singlelosers.dat file*

playerlose<<' ';

}

playerlose.seekg(0,ios::beg);

do

{

playerlose>>nametaken;

if(strcmpi(nametaken,human)==0)

lose++;

}while(playerlose.eof()==0);

playerlose.close();

int draw=0;

fstream sysdraw;

sysdraw.open("SingleDraws.dat",ios::in|ios::out|ios::app); *//opens file*

*containing details of draw matches*

sysdraw.seekg(0,ios::end);

if(i==2)

{

sysdraw<<human; *//writes the name of player to SingleDraws.dat file*

sysdraw<<' ';

}

sysdraw.seekg(0,ios::beg);

do

{

sysdraw>>nametaken;

if(strcmpi(nametaken,human)==0)

draw++; *//counts the number of draws of the player*

}while(sysdraw.eof()==0);

sysdraw.close();

float winpercent=0.0; *//Calculates scores*

winpercent=((win)/(win+lose))\*100;

system("cls");

window(1,1,50,60);

textbackground(3);

clrscr();

textcolor(5);

gotoxy(40,50);

cout<<"\n\n\n\n\n "<<human<<", Do you like to view your Score Statistics?

(y/n): ”;

char ch; *//Choice for viewing score statistics*

cin>>ch;

if(ch=='y'||ch=='Y')

{

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(6);

clrscr();

cout<<"\n\n\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_Score Statistics of "<<human<<"\_\_\_\_\_\_\_\_\_\_\_\_\_"; *//Displays score statistics*

*` of player* *in single player mode*

cout<<"\n\n\t\t Total Games Played : "<<win+lose+draw;

cout<<"\n\t\t Total number of Victories : "<<win;

cout<<"\n\t\t Total number of Defeats : "<<lose;

cout<<"\n\t\t Total number of Draws : "<<draw;

cout<<"\n\t\t Win Percentage : "<<setprecision(1)<<winpercent<<"%";

cout<<"\n\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

getchar();

}

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(CYAN);

clrscr();

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(CYAN);

clrscr();

cout<<"\n\n\n\n\n Do you like to play again? (y/n): ";

cin>>re; *//Choice for Replay*

system("cls");

if(re=='n'||re=='N') *//Display Exiting Screen*

{

system("cls");

textcolor(15);

gotoxy(40,50);

window(1,1,400,400);

textbackground(15);

clrscr();

cout<<"\n\n\n\n\n\n\n\n\n\n\n\t\t\t ================\n";

cout<<"\t\t\t = Exiting Game =\n";

cout<<"\t\t\t ================\n";

getch();

}

}while(re=='y'||re=='Y');}

int checkwinmulti() *//Function to return game status, 1 for game is over with result, -1 for* *game is in progress, 0 for game draw*

{

if (square[1] == square[2] && square[2] == square[3])

return 1;

else if (square[4] == square[5] && square[5] == square[6])

return 1;

else if (square[7] == square[8] && square[8] == square[9])

return 1;

else if (square[1] == square[4] && square[4] == square[7])

return 1;

else if (square[2] == square[5] && square[5] == square[8])

return 1;

else if (square[3] == square[6] && square[6] == square[9])

return 1;

else if (square[1] == square[5] && square[5] == square[9])

return 1;

else if (square[3] == square[5] && square[5] == square[7])

return 1;

else if (square[1] != '1' && square[2] != '2' && square[3] != '3' &&

square[4] != '4' && square[5] != '5' && square[6] != '6' && square[7] != '7' && square[8] '8' && square[9] != '9')

return 0;

else

return -1;

}

void boardmulti() *//Function to display the board of multiplayer*

{

clrscr();

cout<<"\n\t\t\t\_\_\_\_\_\_\_\_\_Multi\_Player\_\_\_\_\_\_\_\_\_\n\n";

cout << "\n\t\t\tPlayer 1 (X) - Player 2 (O)" << endl << endl;

cout << endl;

cout << "\t\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t\t\t | | | | " << endl;

cout << "\t\t\t | " << square[1] << " | " << square[2] << " | " << square[3] <<"

|"<<endl;

cout << "\t\t\t |\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_| " << endl;

cout << "\t\t\t | | | | " << endl;

cout << "\t\t\t | " << square[4] << " | " << square[5] << " | " << square[6] <<"

|"<<endl;

cout << "\t\t\t |\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_| " << endl;

cout << "\t\t\t | | | | " << endl;

cout << "\t\t\t | " << square[7] << " | " << square[8] << " | " << square[9] <<"

|"<<endl;

cout << "\t\t\t |\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_| " << endl << endl;

}

void boardsingle() *//Function to display the board of singleplayer*

{

clrscr();

cout<<"\n\t\t\t\_\_\_\_\_\_\_\_\_Single\_Player\_\_\_\_\_\_\_\_\_\n";

cout << "\n\n\t\t\tPlayer (X) - Computer (O)" << endl << endl;

cout << endl;

cout << "\t\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t\t\t | | | | " << endl;

cout << "\t\t\t | " << square[1] << " | " << square[2] << " | " << square[3] <<"

|"<<endl;

cout << "\t\t\t |\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_| " << endl;

cout << "\t\t\t | | | | " << endl;

cout << "\t\t\t | " << square[4] << " | " << square[5] << " | " << square[6] <<"

|"<<endl;

cout << "\t\t\t |\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_| " << endl;

cout << "\t\t\t | | | | " << endl;

cout << "\t\t\t | " << square[7] << " | " << square[8] << " | " << square[9] <<"

|"<<endl;

cout << "\t\t\t |\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_| " << endl << endl;

}

void playerinput() *//Function to check validity of player input and insert if valid*

{

textbackground(3);

textcolor(15);

int length;

int i=0;

char choice[50];

char ch;

do

{

boardsingle();

cout << "\n\t\t\tPlease choose a grid to place (X): ";

cin>>choice; *//Accepts player input*

ch=choice[0];

length=strlen(choice); *//Length of input*

cout<<"\n";

if(length==1) *//Checks if choice is single digit*

{

if(ch>=48||ch<=56) *//Checks if input is numeric*

{

if (ch == '1' && square[1] == '1')

{

square[1] = 'X';

i=0;

}

else if (ch == '2' && square[2] == '2')

{

square[2] = 'X';

i=0;

}

else if (ch == '3' && square[3] == '3')

{

square[3] = 'X';

i=0;

}

else if (ch == '4' && square[4] == '4')

{

square[4] = 'X';

i=0;

}

else if (ch == '5' && square[5] == '5')

{

square[5] = 'X';

i=0;

}

else if (ch == '6' && square[6] == '6')

{

square[6] = 'X';

i=0;

}

else if (ch == '7' && square[7] == '7')

{

square[7] = 'X';

i=0;

}

else if (ch == '8' && square[8] == '8')

{

square[8] = 'X';

i=0;

}

else if (ch == '9' && square[9] == '9')

{

square[9] = 'X';

i=0;

}

else

{

cout<<"\t\t\t\tInvalid move! ";

getch();

i=1;

}

}

}

if(length>1)

{

cout<<"\t\t\t\tInvalid move! ";

getch();

i=1;

}

}while(i==1);

}

void computerAI() *//Computer decides its move using artificial intelligence*

{

if(square[1] == 'O' && square[2] == 'O' && square[3] == '3') *//Checks chance for* *offensive move*

square[3] = 'O';

else if(square[1] == 'O' && square[3] == 'O' && square[2] == '2')

square[2] = 'O';

else if(square[2] == 'O' && square[3] == 'O' && square[1] == '1')

square[1] = 'O';

else if(square[4] == 'O' && square[5] == 'O' && square[6] == '6')

square[6] = 'O';

else if(square[4] == 'O' && square[6] == 'O' && square[5] == '5')

square[5] = 'O';

else if(square[5] == 'O' && square[6] == 'O' && square[4] == '4')

square[4] = 'O';

else if(square[7] == 'O' && square[8] == 'O' && square[9] == '9')

square[9] = 'O';

else if(square[7] == 'O' && square[9] == 'O' && square[8] == '8')

square[8] = 'O';

else if(square[8] == 'O' && square[9] == 'O' && square[7] == '7')

square[7] = 'O';

else if(square[1] == 'O' && square[4] == 'O' && square[7] == '7')

square[7] = 'O';

else if(square[1] == 'O' && square[7] == 'O' && square[4] == '4')

square[4] = 'O';

else if(square[4] == 'O' && square[7] == 'O' && square[1] == '1')

square[1] = 'O';

else if(square[2] == 'O' && square[5] == 'O' && square[8] == '8')

square[8] = 'O';

else if(square[2] =='O' && square[8] == 'O' && square[5] == '5')

square[5] = 'O';

else if(square[5] == 'O' && square[8] == 'O' && square[2] == '2')

square[2] = 'O';

else if(square[3] == 'O' && square[6] == 'O' && square[9] == '9')

square[9] = 'O';

else if(square[3] == 'O' && square[9] == 'O' && square[6] == '6')

square[6] = 'O';

else if(square[6] == 'O' && square[9] == 'O' && square [3] == '3')

square[3] = 'O';

else if(square[1] == 'O' && square[5] == 'O' && square[9] == '9')

square[9] = 'O';

else if(square[1] == 'O' && square[9] == 'O' && square[5] == '5')

square[5] = 'O';

else if(square[5] == 'O' && square[9] == 'O' && square[1] == '1')

square[1] = 'O';

else if(square[3] == 'O' && square[5] == 'O' && square[7] == '7')

square[7] = 'O';

else if(square[3] == 'O' && square[7] == 'O' && square[5] == '5')

square[5] = 'O';

else if(square[5] == 'O' && square[7] == 'O' && square[3] == '3')

square[3] = 'O';

else if(square[1] == 'X' && square[5] == 'X' && square[9] == '9')

square[9] = 'O';

else if(square[1] == 'X' && square[9] == 'X' && square[5] == '5')

square[5] = 'O';

else if(square[5] == 'X' && square[9] == 'X' && square[1] == '1')

square[1] = 'O';

else if(square[7] == 'X' && square[5] == 'X' && square[3] == '3')

square[3] = 'O';

else if(square[7] == 'X' && square[3] == 'X' && square[5] == '5')

square[5] = 'O';

else if(square[5] == 'X' && square[3] == 'X' && square[7] == '7')

square[7] = 'O';

else if(square[1] == 'X' && square[2] == 'X' && square[3] == '3')

square[3] = 'O';

else if(square[1] == 'X' && square[3] == 'X' && square[2] == '2')

square[2] = 'O';

else if(square[2] == 'X' && square[3] == 'X' && square[1] == '1')

square[1] = 'O';

else if(square[4] == 'X' && square[5] == 'X' && square[6] == '6')

square[6] = 'O';

else if(square[4] == 'X' && square[6] == 'X' && square[5] == '5')

square[5] = 'O';

else if(square[5] == 'X' && square[6] == 'X' && square[4] == '4')

square[4] = 'O';

else if(square[7] == 'X' && square[8] == 'X' && square[9] == '9')

square[9] = 'O';

else if(square[7] == 'X' && square[9] == 'X' && square[8] == '8')

square[8] = 'O';

else if(square[8] == 'X' && square[9] == 'X' && square[7] == '7')

square[7] = 'O';

else if(square[1] == 'X' && square[4] == 'X' && square[7] == '7')

square[7] = 'O';

else if(square[1] == 'X' && square[7] == 'X' && square[4] == '4')

square[4] = 'O';

else if(square[4] == 'X' && square[7] == 'X' && square[1] == '1')

square[1] = 'O';

else if(square[2] == 'X' && square[5] == 'X' && square[8] == '8')

square[8] = 'O';

else if(square[2] == 'X' && square[8] == 'X' && square[5] == '5')

square[5] = 'O';

else if(square[5] == 'X' && square[8] == 'X' && square[2] == '2')

square[2] = 'O';

else if(square[3] == 'X' && square[6] == 'X' && square[9] == '9')

square[9] = 'O';

else if(square[3] == 'X' && square[9] == 'X' && square[6] == '6')

square[6] = 'O';

else if(square[6] == 'X' && square[9] == 'X' && square[3] == '3')

square[3] = 'O';

else

{ *//Checks for a possible move when neither offensive nor* *defensive move is possible*

if(square[1] == '1')

square[1] = 'O';

else if(square[2] == '2')

square[2] = 'O';

else if(square[3] == '3')

square[3] = 'O';

else if(square[4] == '4')

square[4] = 'O';

else if(square[5] == '5')

square[5] = 'O';

else if(square[6] == '6')

square[6] = 'O';

else if(square[7] == '7')

square[7] = 'O';

else if(square[8] == '8')

square[8] = 'O';

else if(square[9] == '9')

square[9] = 'O';

}

}

int checkwinsingle() *//Function that checks whether player or computer wins*

{

if(square[1] == 'O' && square[5] == 'O' && square[9] == 'O')

return 2; *//Returning 2 makes* *computer win*

else if(square[3] == 'O' && square[5] == 'O' && square[7] == 'O')

return 2;

else if(square[1] == 'O' && square[2] == 'O' && square[3] == 'O')

return 2;

else if(square[4] == 'O' && square[5] == 'O' && square[6] == 'O')

return 2;

else if(square[7] == 'O' && square[8] == 'O' && square[9] == 'O')

return 2;

else if(square[1] == 'O' && square[4] == 'O' && square[7] == 'O')

return 2;

else if(square[2] == 'O' && square[5] == 'O' && square[8] == 'O')

return 2;

else if(square[3] == 'O' && square[6] == 'O' && square[9] == 'O')

return 2;

else if(square[1] == 'X' && square[5] == 'X' && square[9] == 'X')

return 1; *//Returning 1 makes* *player win*

else if(square[3] == 'X' && square[5] == 'X' && square[7] == 'X')

return 1;

else if(square[1] == 'X' && square[2] == 'X' && square[3] == 'X')

return 1;

else if(square[4] == 'X' && square[5] == 'X' && square[6] == 'X')

return 1;

else if(square[7] == 'X' && square[8] == 'X' && square[9] == 'X')

return 1;

else if(square[1] == 'X' && square[4] == 'X' && square[7] == 'X')

return 1;

else if(square[2] == 'X' && square[5] == 'X' && square[8] == 'X')

return 1;

else if(square[3] == 'X' && square[6] == 'X' && square[9] == 'X')

return 1;

else if (square[1] != '1' && square[2] != '2' && square[3] != '3'

&& square[4] != '4' && square[5] != '5'

&& square[6] != '6' && square[7] != '7'

&&square[8] != '8' && square[9] != '9')

return 0; *//Returning 0 makes the game draw*

else

return -1; *//Returning -1 declares that the game is in progress*

}

/\*==========================END OF PROGRAM===============================\*/

**Output Screen Layout**

Loading screen

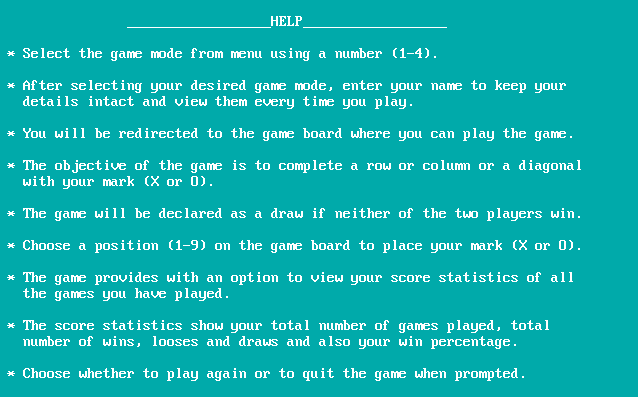


Welcome Screen 

Game Menu

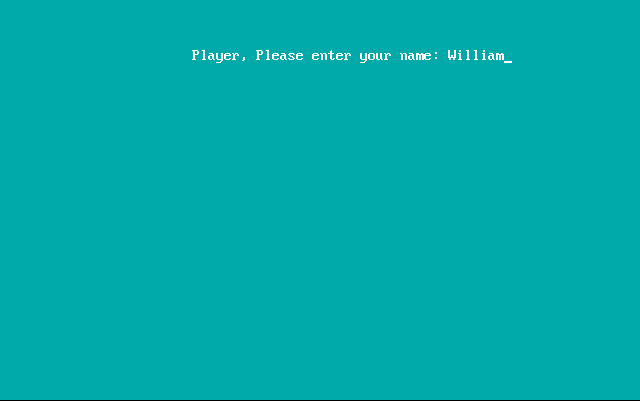


Help Window

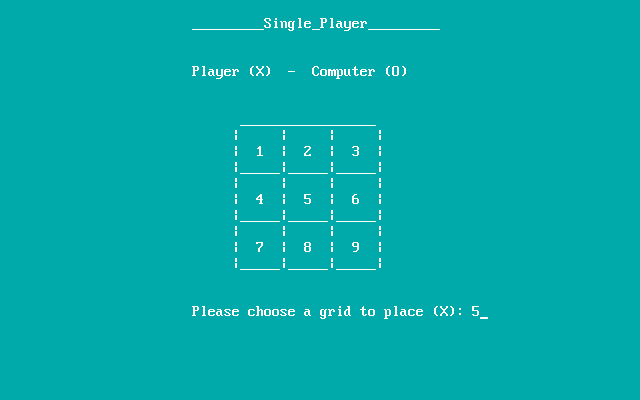


**Single Player Layout**

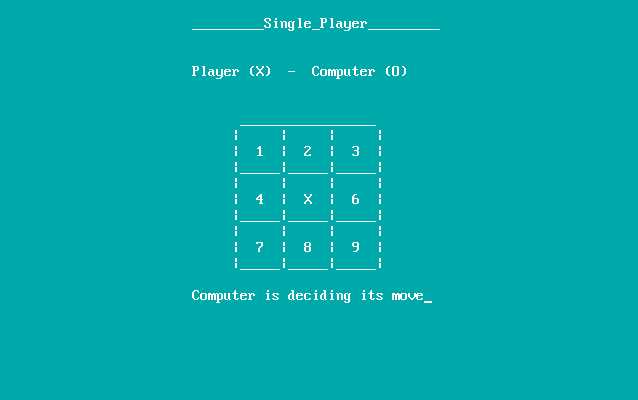
Name Input

****

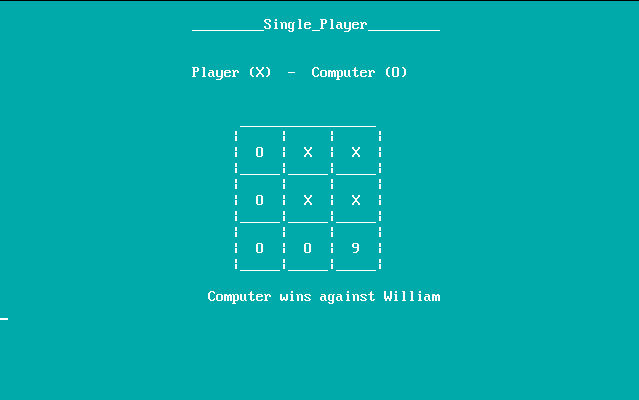
Single Player board



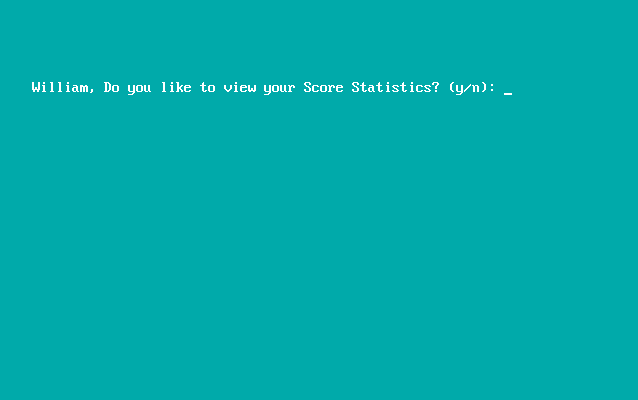
1 second pause for computer to decide its move



Displays the game result



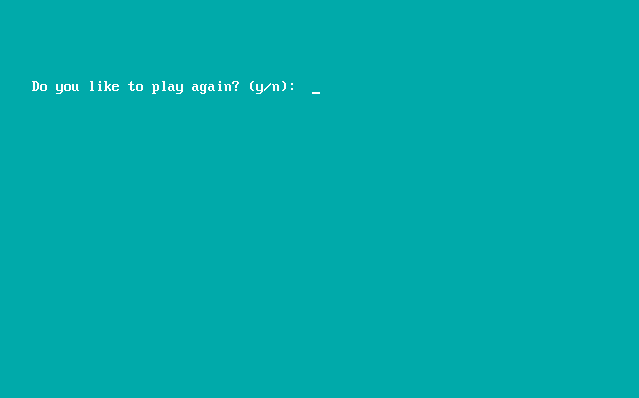
Prompt to view score



Displays Score Board

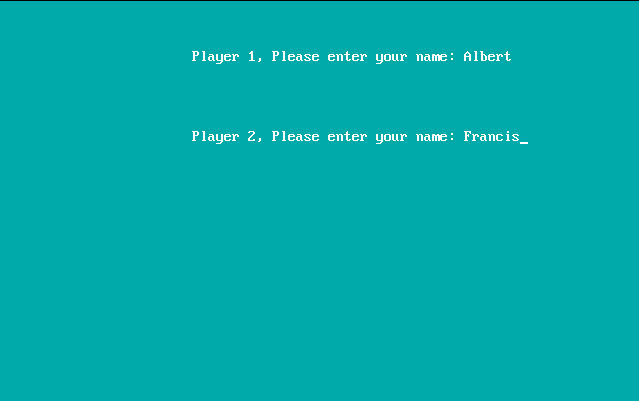


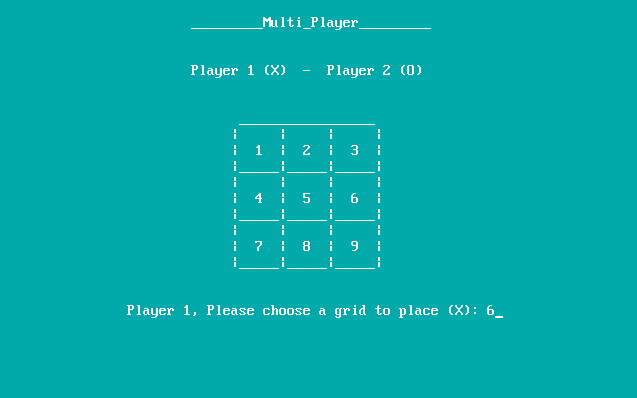
Prompt to play again



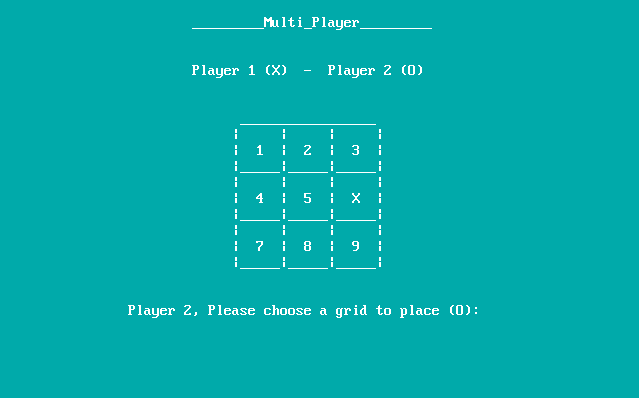
**Multi Player Layout**

Name Input

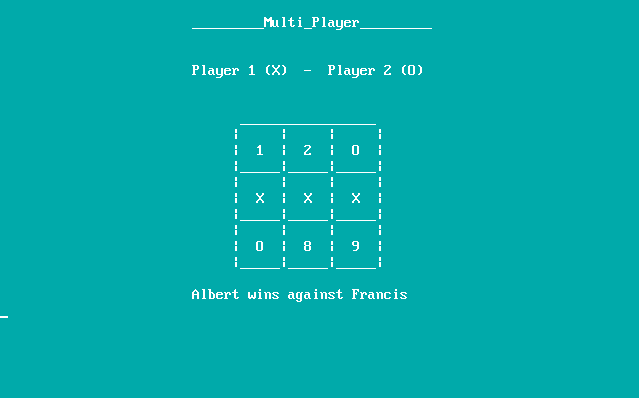


Multi Player Board 

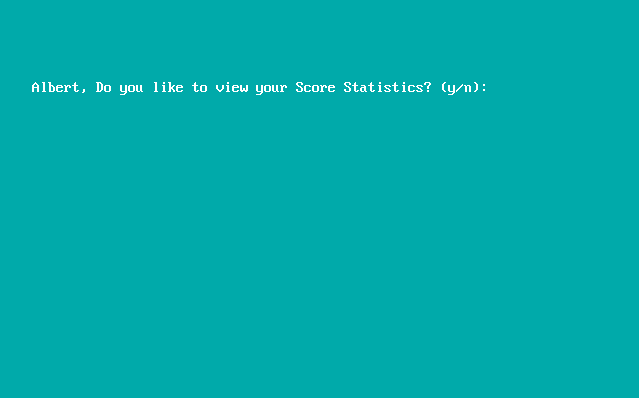
Turn of player 2



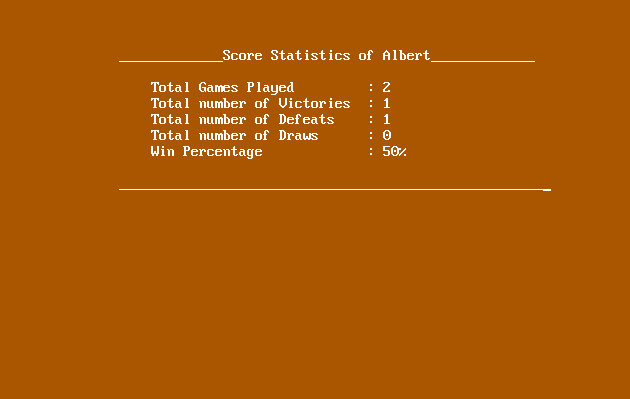
Displays the game result



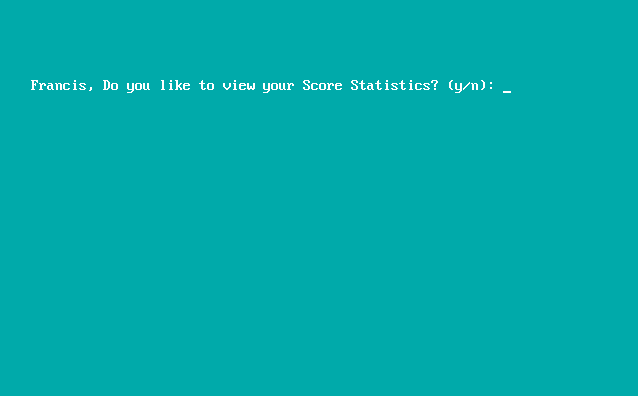
Prompt to view score of winner



Displays Score Board of winner



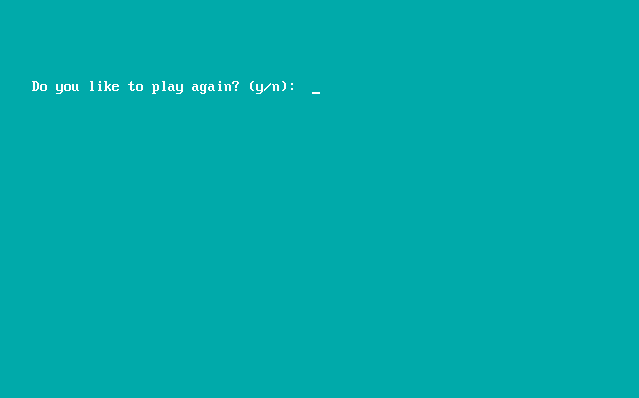
Prompt to view score of loser



Displays Score Board of loser



Prompt to play again



Exit Game Screen



**Conclusion**

This project which has evolved and improved step-by-step finally stands out to be a successful one. Basic concepts of Artificial Intelligence are implemented into the game without any bugs. The program paves way for further improvement, even up to the level of game marketing. The graphical-user interface of the game turns out to be much reliable and user-friendly than expected. We gladly conclude this project with complete satisfaction.

**Bibliography**

1. Computer Science with C++ – Sumita Arora
2. Introduction to C++ programming & graphics – C. Pozrikidis
3. https://en.wikipedia.org/
4. www.cplusplus.com
5. www.tutorialspoint.com