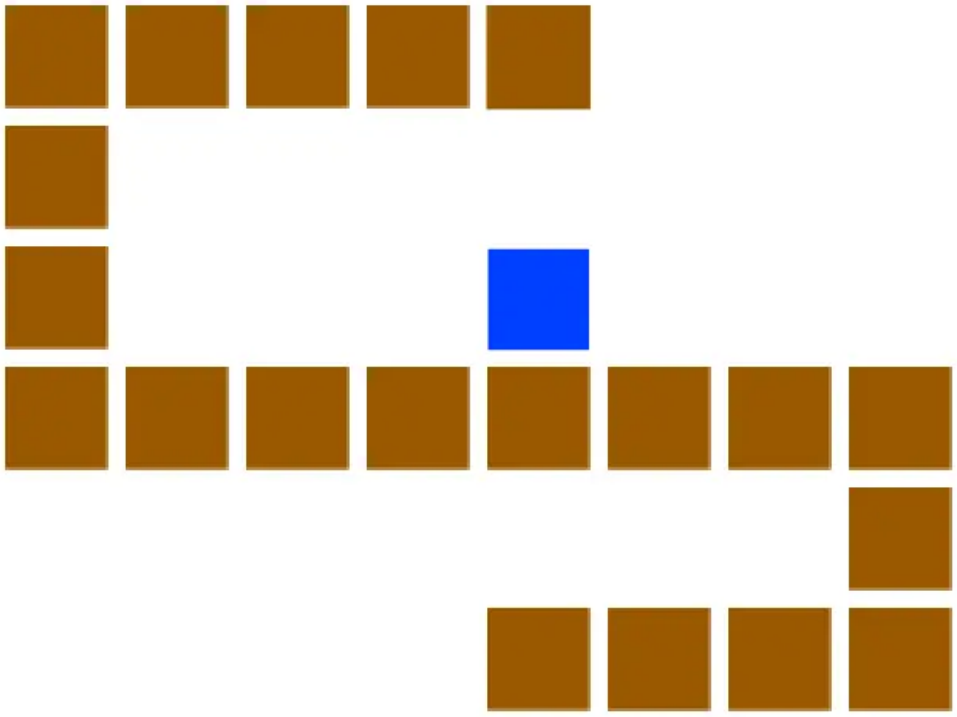
Object Oriented Programming

Retro Snake Game



Abhigyan Adarsh(2K19/EE/005)
abhigyanadarsh_2k19ee005@dtu.ac.in

Abhik Kumar (2K19/EE/009)
abhikkumar_2k19ee009@dtu.ac.in

Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and colleagues. We would like to extend our sincere thanks to all of them. We are highly indebted to DTU and especially Anjali Bansal Ma'am for her guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project. We are grateful for everything we have learned due to this opportunity. All of the research papers and documents that we read in order to finish this job were really beneficial. We are sure that these things will prove to be of skillful use in the future as well. Thank you so much to all of our coworkers who have been so helpful at this time. We believe that words will fall short of expressing our gratitude for the opportunity to work on this project and to explore and broaden our perspectives.

Thank you everyone,
With Regards,

Abhigyan Adarsh (2K19/EE/005)
abhigyanadarsh_2k19ee005@dtu.ac.in

Abhik Kumar (2K19/EE/009)

abhikkumar_2k19ee009@dtu.ac.in

Index

Abstract

During our childhood most of us love to play arcade snake game on gameboy or on our parent's cellphones and a lot of memories has been linked with it, we have tried to replicate this game in such a way that it can be executed on any system without any system requirements with as minimal effort as possible. To explore the various possibilities that can be achieved using an object oriented programming language such as C++ and giving a touch of entertainment for the user using a simple print-clear-reprint the screen approach. We have introduced the 3 stages of the game (Easy, Medium, Hard) by adjusting the playing field area and speed of the snake.

Introduction

Object oriented programming:

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

Concepts Incorporation:

- **Classes and objects:** The project involves class definitions as blueprints for the objects to be created for the snake, the food and the board.

- **Encapsulation:** The use of classes implies the concept of encapsulation being incorporated as classes bind the data and the methods that work upon them together. E.g. the snake class which handles the coordinates of the snake body as an array of coordinates, also contains the methods to update the position and length of the snake as the game progresses.

- **Constructor:** The classes used in the project have their constructors to initialize the state of the game.

Code Segmentation

Header files:

- iostream
  - Including this header may automatically include other headers, such as <ios>, <streambuf>, <istream>, <ostream> and/or <iosfwd>.
- windows
  - Windows.h is the main header file for WinAPI. WinAPI is anything and everything related to programming on Windows. Anything that involves window creation/management or communication with the OS or filesystem.
  - Things like:
    - Creating Windows
    - Basic graphical capabilities (WinGDI)
    - Enumerating files in a directory
    - Popping up common dialog boxes ("Save As" dialog, "Pick a color" dialog, etc)
    - Querying information about the system (like running processes, etc)
    - etc

Classes:

- position

- Field
  - **Functions:**
    - Void display()
    - Void clear()
    - Int get_width() const
    - Int get_height() const
    - Void draw()
  - **Constructor:** Field
  - **Destructor:** ~Field
  - **Object:** field

- Food
    - **Functions:**
        - Void set_pos()
        - Void relocate()
        - Int get_x() const
        - Int get_y() const
        - Char get_symbol() const
    - **Constructor:** Food
    - **Object:** food

- Snake
    - **Function:**
        - Bool check_food()
        - Void get_input()
        - Void move()
        - Void draw()
        - Int get_x() const
        - Int get_y() const
        - Char get_symbol() const
    - **Constructor:** Snake
    - **Object:** snake

Function:

- gotoxy()

Code

```
//Retro Snake Game
#include <iostream>
#include <windows.h>
#define UP 0
#define DOWN 1
#define LEFT 2
#define RIGHT 3
using namespace std;

// Class for position of elements
class position
{
	public:
		int x,y;
};

// Access the the location in the display
void gotoxy (int x, int y)
{
	COORD co_ord;	// coordinates is declared as COORD
	co_ord.X = x;	// defining x-axis
	co_ord.Y = y;	//defining  y-axis
	SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),co_ord);
// inside windows header
}

// Play Area
class Field
{
		char ** field;
		Field(const Field &);
	public:
		static int field_height;
		static int field_width;
		Field() // contructor
		{
			field = new char*[Field::field_height];
			for(int c = 0; c < Field::field_height; ++c)
			{
```

```
                        field[c] = new char[Field::field_width];
                }
        }
        ~Field() // destructor
        {
                for(int c = 0; c < Field::field_height; ++c)
                {
                        delete[] field[c];
                }
                delete[] field;
        }

        //For game play boundary
        void display()
        {
                gotoxy(0,0);
                for(int y = 0; y < field_height; ++y)
                {
                        for(int x = 0; x < field_width; ++x)
                        {

if(x==0||x==field_width-1||y==0||y==field_height-1)
                                        cout<<"~";
                                else
                                        cout << field[y][x];
                        }
                        cout << endl;
                }
        }

        // Blank space
        void clear()
        {
                for(int y = 0; y < field_height; ++y)
                {
                        for(int x = 0; x < field_width; ++x)
                        {
                                field[y][x] = ' ';
                        }
                }
        }
```

```
		int get_width() const // constant function
		{
			return field_width;
		}
		int get_height() const
		{
			return field_height;
		}
		void draw(int y, int x, char what)
		{
			field[y][x] = what;
		}
} field;

// food class
class Food
{
		position pos;
		char symbol;
	public:
		Food(): symbol('X'), pos()
		{
			pos.x = pos.y = -1;
		}
		void set_pos(int x, int y)
		{
			pos.x = x;
			pos.y = y;
		}
		void relocate(const Field & field)
		{
			pos.x = 1+(rand() % (field.get_width()-2));
			pos.y = 1+(rand() % (field.get_height()-2));
		}
		int get_x() const
		{
			return pos.x;
		}
		int get_y() const
		{
			return pos.y;
```

```
		}
		char get_symbol() const
		{
			return symbol;
		}
} food;

// Snake class
class Snake
{
		int dir;
		char symbol, head_symbol;
		position pos[100]; // Array of type position
		position & head; // Reference of position uisng head
		int speed;
		int size;
		bool can_turn;
	public:
		Snake(int x, int y): //
			symbol('*'), head_symbol('O'), pos(),
			speed(1), size(1), dir(RIGHT),
			head(pos[0]), can_turn(true)
		{
			pos[0].x = x;
			pos[0].y = y;
		}
		bool check_food(const Food & food)
		{
			if(food.get_x() == head.x && food.get_y() == head.y)
			{
				size += 1;
				return true;
			}
			return false;
		}
		// to get control from keyboard
		void get_input()
		{
			if(GetAsyncKeyState(VK_UP) && dir != DOWN) //
GetAsyncKeyState to automatically keep snake moving
			{
```

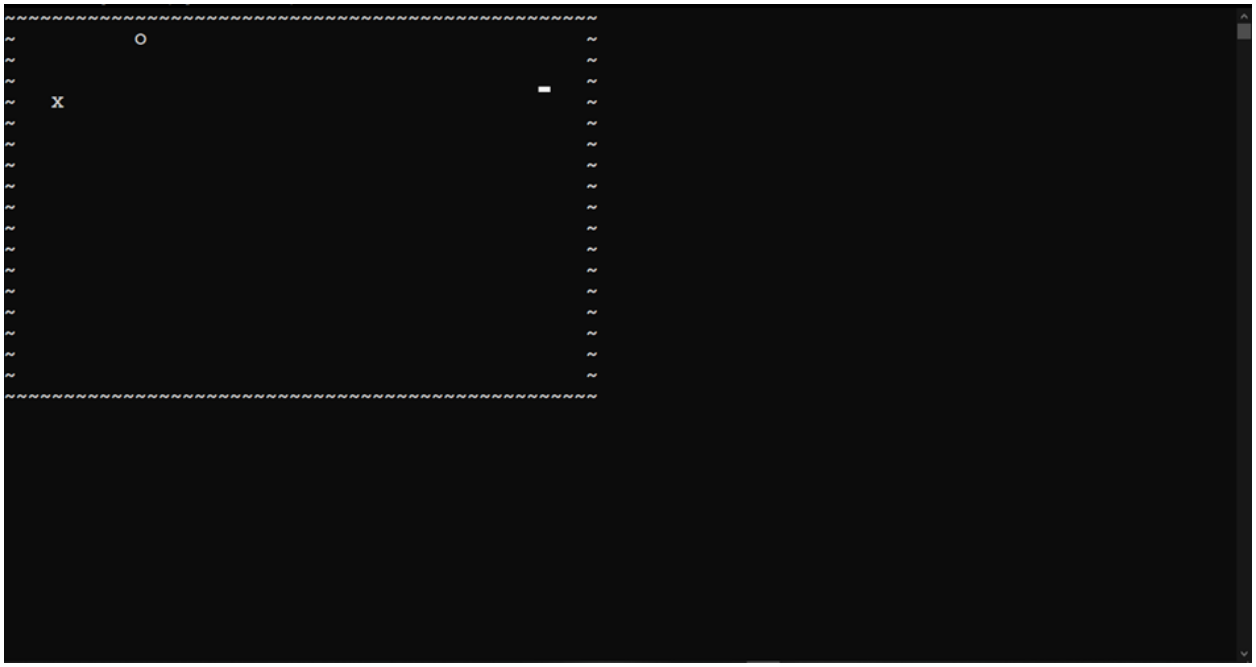
```
                dir = UP;
        }
        if(GetAsyncKeyState(VK_DOWN) && dir != UP)
        {
                dir = DOWN;
        }
        if(GetAsyncKeyState(VK_LEFT) && dir != RIGHT)
        {
                dir = LEFT;
        }
        if(GetAsyncKeyState(VK_RIGHT) && dir != LEFT)
        {
                dir = RIGHT;
        }
}
// To move the snake
void move(const Field & field)
{
        position next = {0, 0};
        switch(dir)
        {
                case UP:
                        next.y = -speed;
                        break;
                case DOWN:
                        next.y = speed;
                        break;
                case LEFT:
                        next.x = -speed;
                        break;
                case RIGHT:
                        next.x = speed;
        }
        for(int c = size - 1; c > 0; --c)
        {
                pos[c] = pos[c-1];
        }
        // move the snake
        head.x += next.x;
        head.y += next.y;
```

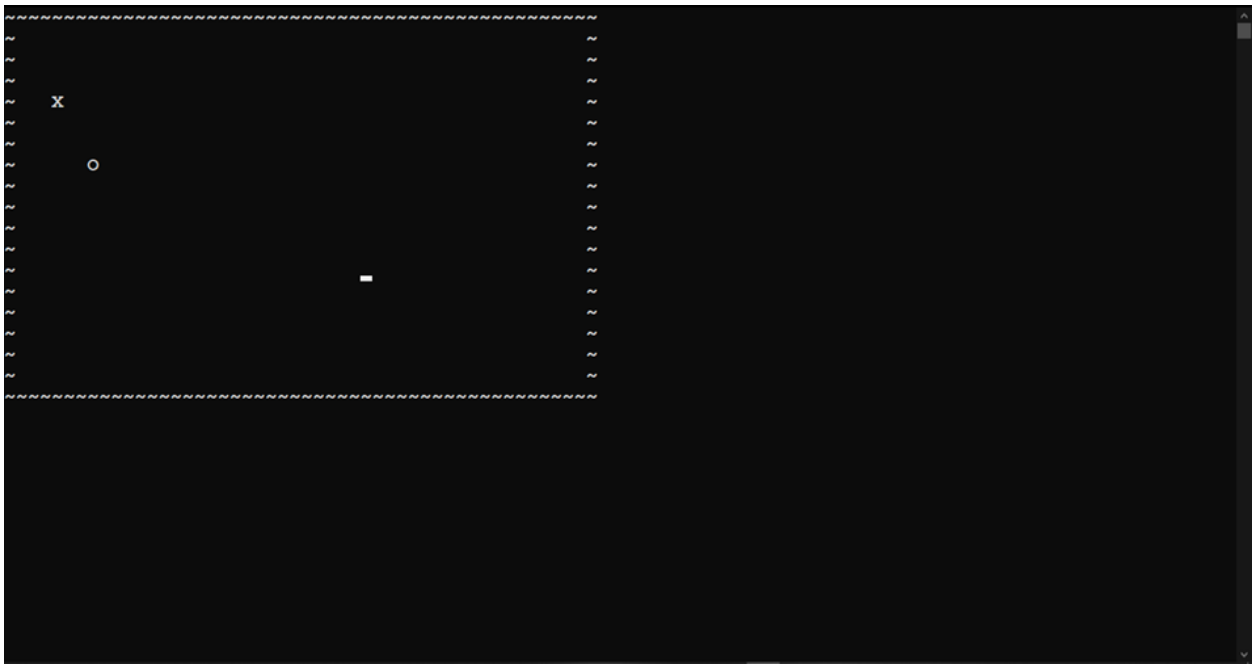
```
                        if(head.x <= 0 || head.y <= 0 || head.x >=
field.get_width()-1 || head.y >= field.get_height()-1)
                        {
                                string s="Your score is ";
                                s.append(to_string(10*(size-1)));
                                throw s; // to terminate the program (Error
handling)
                        }
                }
                // to draw snake
                void draw(Field & field)
                {
                        for(int c = 0; c < size; ++c)
                        {
                                if(c == 0)
                                {
                                        field.draw(pos[c].y, pos[c].x,
head_symbol);
                                }
                                else
                                {
                                        field.draw(pos[c].y, pos[c].x, symbol);
                                }
                        }
                }
                int get_x() const
                {
                        return head.x;
                }
                int get_y() const
                {
                        return head.y;
                }
                char get_symbol() const
                {
                        return symbol;
                }
} snake(1, 1);

int Field::field_height = 24; //
int Field::field_width = 79;
```

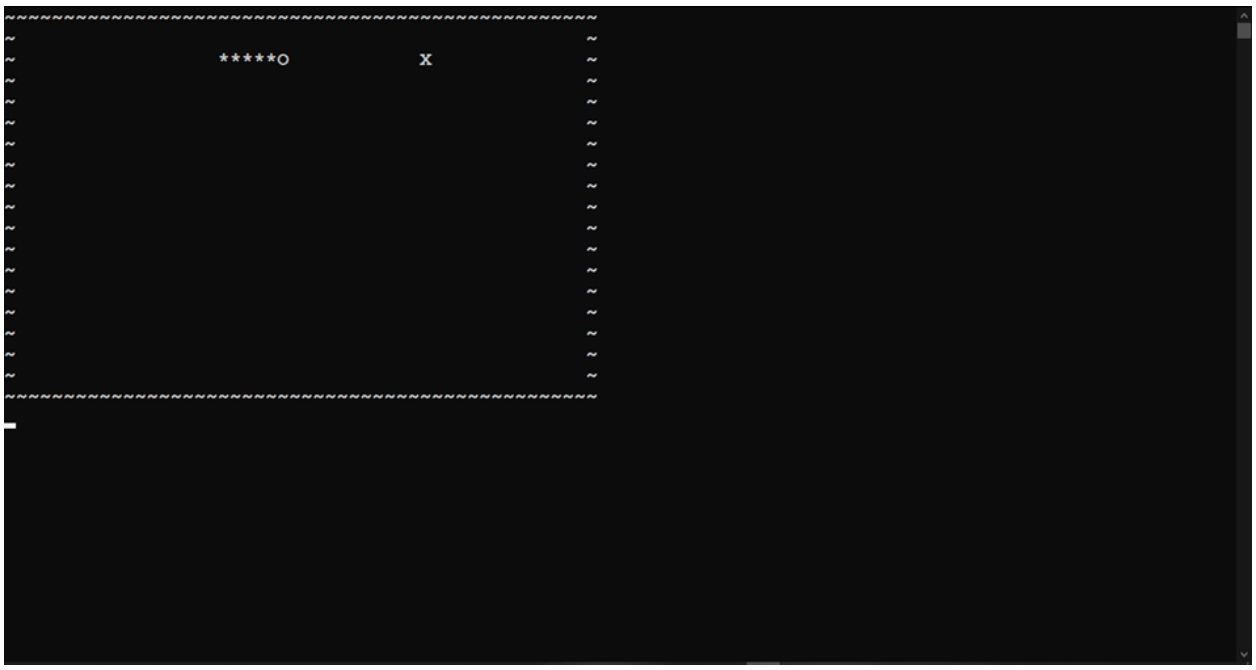
```
int main()
{
	field.clear(); //
	cout<<"Enter Level \n1. Hard\n2. Medium\n3. Easy\n";
	int x;
	cin>>x;
	// By adjusting the field size we are setting the diffucilty.
	if(x==1)
	{
		Field:: field_height=12;
		Field:: field_width=25;
	}
	else if(x==2)
	{
		Field:: field_height=19;
		Field:: field_width=50;
	}
	else
	{
		Field:: field_height=24;
		Field:: field_width=79;
	}
	system("cls");
	food.set_pos(4, 4);
	// Updating the scene
	while(1)
	{
		field.clear();
		snake.get_input();
		try
		{
			snake.move(field);
		}
		catch (string s)
		{
			field.clear();
			cerr << s << endl; // Display the error message
immediately
			system("pause");
			return -1;
```

```
                }
                snake.draw(field);
                field.draw(food.get_y(), food.get_x(), food.get_symbol());
                if(snake.check_food(food))
                {
                        food.relocate(field);
                }
                field.display();
                Sleep(100/x); //Update time, keeping the program on hold
        }
        return 0;
}
```

Output

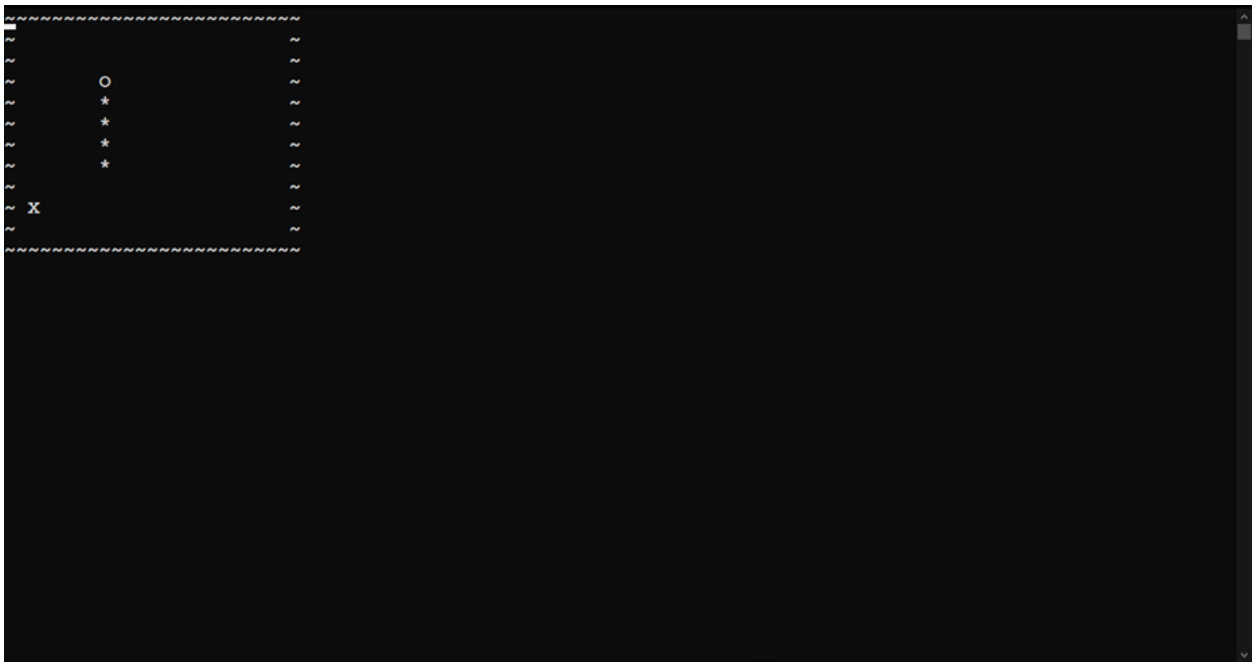


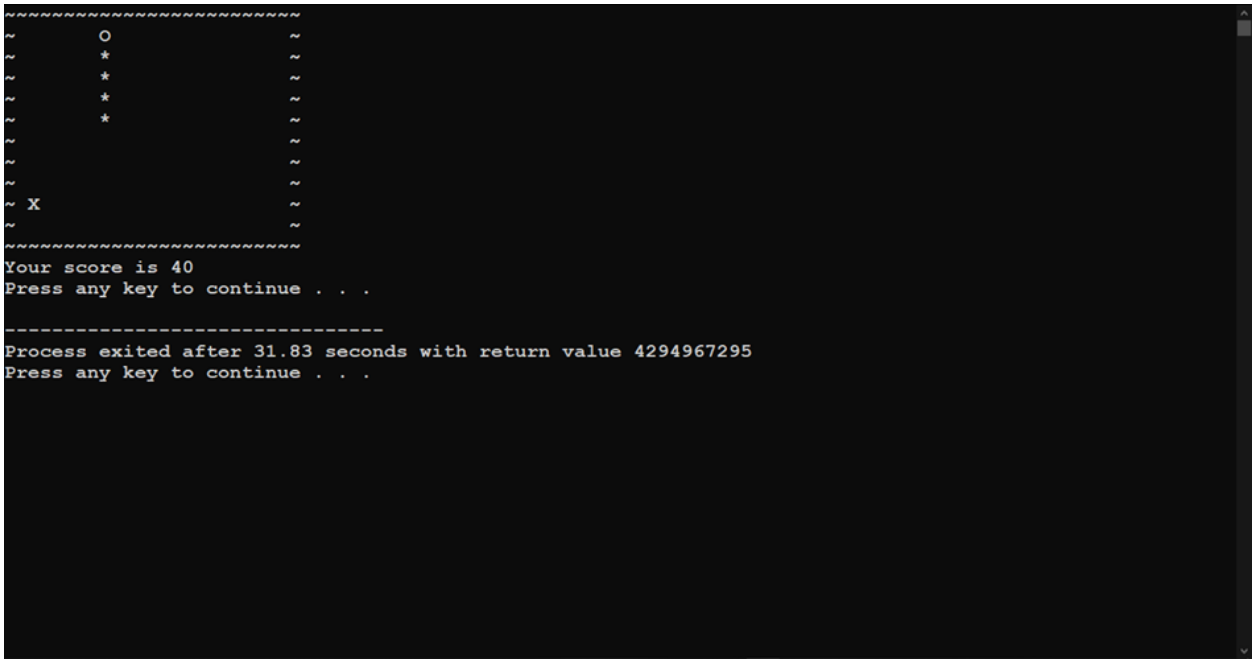

x
o


*****o
x

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~                                                ~
~                                                ~
~                                                ~
~                                                ~
~                                                ~
~                                                ~
~                                                ~
~                                                ~
~                                                ~
~        **                                      ~
~         *                                      ~
~         *                                      ~
~         *                                      ~
~         *                                      ~
~         *                                      ~
~         *               X                      ~
~         O                                      ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Your score is 80
Press any key to continue . . .

--------------------------------
Process exited after 75.13 seconds with return value 4294967295
Press any key to continue . . .
```

```
Enter Level
1. Hard
2. Medium
3. Easy
1
```


~~~~~~~~~~~~~~~~~~~~~~~~~~
~   ~
~   ~
~   O   ~
~   *   ~
~   *   ~
~   *   ~
~   *   ~
~   ~
~ X   ~
~   ~
~~~~~~~~~~~~~~~~~~~~~~~~~~


~~~~~~~~~~~~~~~~~~~~~~~~~~
~   O   ~
~   *   ~
~   *   ~
~   *   ~
~   *   ~
~   ~
~   ~
~   ~
~ X   ~
~   ~
~~~~~~~~~~~~~~~~~~~~~~~~~~
Your score is 40
Press any key to continue . . .

Process exited after 31.83 seconds with return value 4294967295
Press any key to continue . . .

THANK YOU

Thank You for your time and support.