Snake Game

Group 14

BSc Management and Information Technology

Department of Industrial Management

Faculty of Science

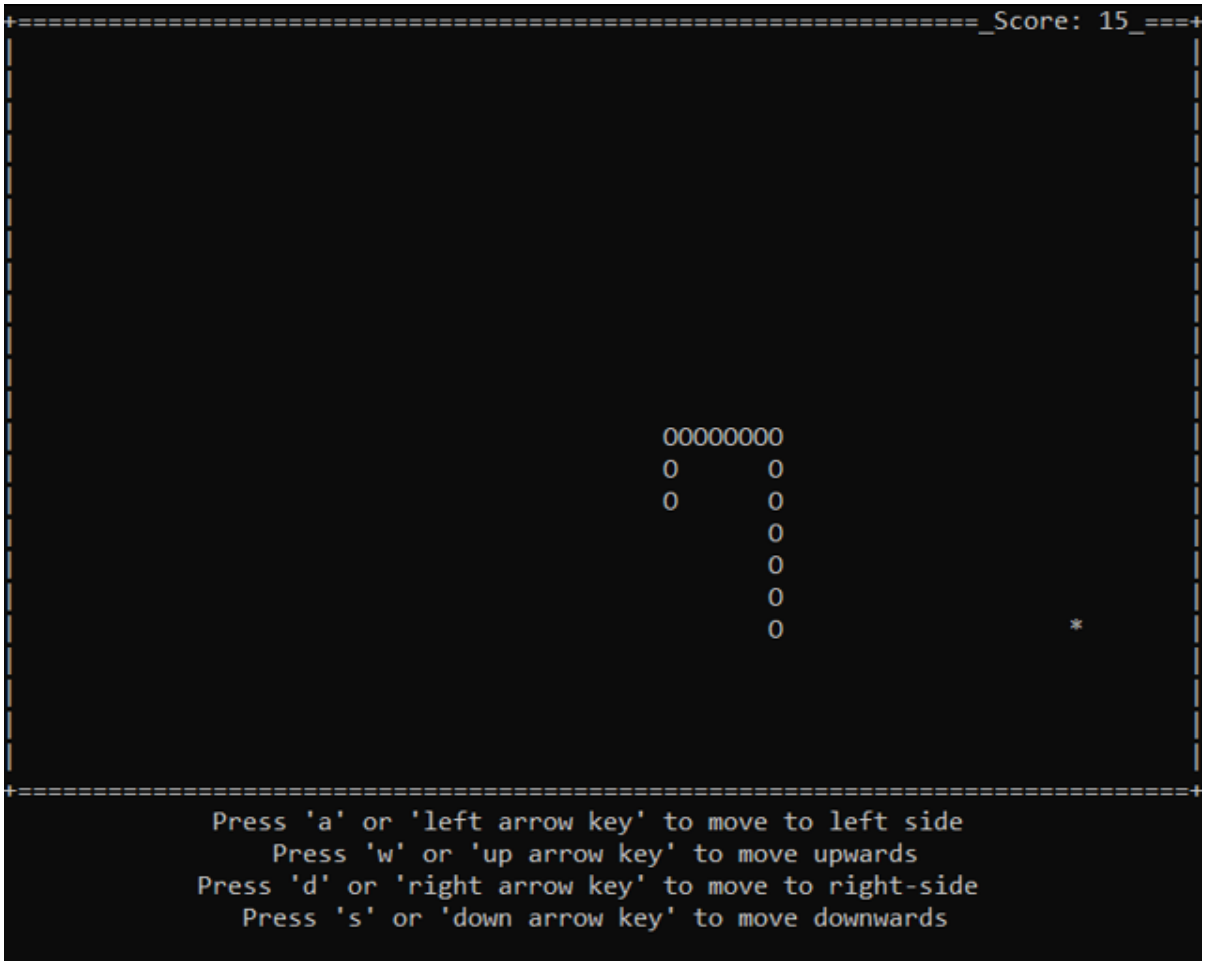
University of Kelaniya

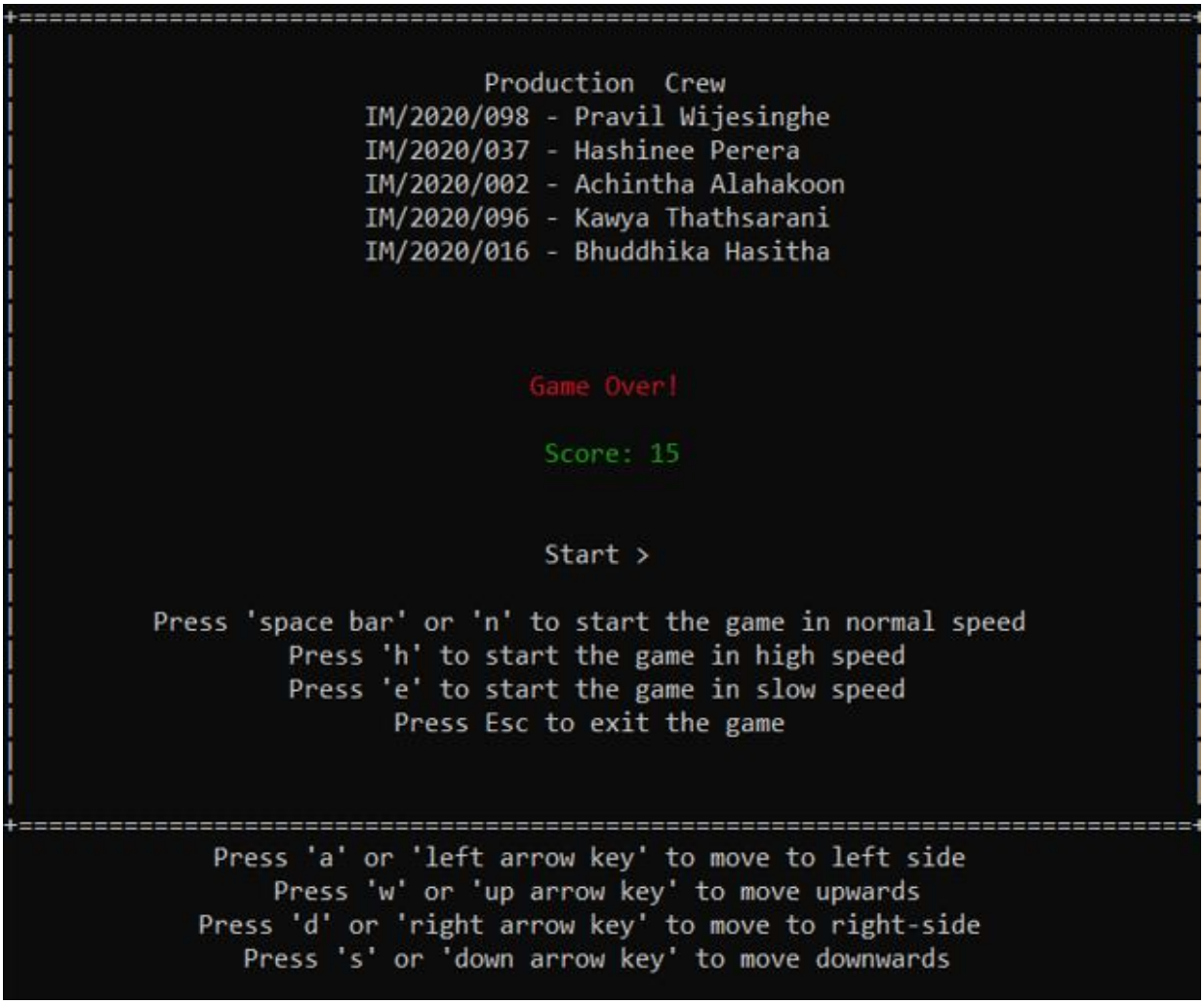
Contents

Introduction of Game

Snake game is a single-player console Game, developed by c++ functions. The game allows users to score points by obtaining eggs. Eggs will be generated at a given interval of time randomly. <iostream>, <fstream>, <windows.h> ,<conio.h>, <cstdlib> , <ctime> , <string> , <vector>, <cmath> <thread> are used as main libraries to create this game.

.


Score: 15
00000000
0 0
0 0
0
0
0
0
*
Press 'a' or 'left arrow key' to move to left side
Press 'w' or 'up arrow key' to move upwards
Press 'd' or 'right arrow key' to move to right-side
Press 's' or 'down arrow key' to move downwards



Instructions to play Game

Every time the snake eats the egg, the length of the snake increases, and the score will increase by 1 point. If the snake hits its own tail player or hits the wall the player will lose and the game will be terminated. The Player must manually adjust the speed of the game using keyboard keys. No extra points are allocated for speed changes. After the game is over, the user is directed to replay the game.

In this game here are the basic controls**:**

- Press the ‘space bar’ or ‘n’ to start the game in normal speed
- Press ‘h’ to start the game in high speed
- Press ‘e’ to start the game in a slow speed
- Press Esc to exit the game

The Snake can be moved in any direction with the help of the keyboard keys. In addition, players can also use normal keyboard arrow keys to move the snake in any direction.

- Press the ‘a’ or ‘left arrow key’ to move to the left side
- Press the ‘w’ or ‘up arrow key’ to move upwards
- Press the ‘d’ or ‘right arrow key’ to move to the right side
- Press the ‘s’ or ‘down arrow key’ to move downwards

Challenges & problems of Game

It was necessary to have wide knowledge about c++ to add advanced features for the game.

The game is supported only in the window's operating system. It is the limitation of this game.

Most of the functions are interconnected with each other. One function error may cause the entire program to crash. Therefore, it was difficult to manage all functions performed by the snake game.

It was difficult to handle too many functions and variables.

Plans to develop Game

Add a user-friendly interface for the game

Improve the multiplayer feature for the game using an online platform

Used sound effects for every significant task

Use graphic effects to make the game more attractive

Adjust the size of the game which display on the monitor

Code

```
#include <iostream>
#include <fstream>
#include <windows.h>
#include <conio.h>
#include <cstdlib>
#include <ctime>
#include <string>
#include <vector>
#include <cmath>
#include <thread>
```

Using namespace and define constants and macros

```
using namespace std;


#define WIDTH 80
#define HEIGHT 25

#define KEY_UP 72
#define KEY_DOWN 80
#define KEY_LEFT 75
#define KEY_RIGHT 77

#define NC "\e[0m"
#define RED "\e[0;31m"
#define GRN "\e[0;32m"
#define CYN "\e[0;36m"
```

Game functions

```
void gotxy(int x, int y);
void ScreenSize();
void screenBorder();
void snakegame();
void Instructions();
void Controls();
void drawSnake();
void drawFood();
void drawScore();
void drawGame();
void moveSnake();
void checkFood();
void generateFood();
void checkSnake();
void crew();
void checkGame();
void startGame();
void changeDirection();
void increaseSpeed();
void clrAndStartGame();
```

Global Variables

```
int score, snakeSize;
int foodX, foodY;
int snakeX[100], snakeY[100];
bool gameOver;
int tailX, tailY;
int gameDelay = 100;
```

enum Function

```
enum eDirection
{
    STOP = 0,
    LEFT,
    RIGHT,
    UP,
    DOWN
};
eDirection direction;
```

gotoxy Function

```
void gotxy(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
```

//screen size is 80x30 cursor is not visible

```
void ScreenSize()
{
    system("mode con cols=80 lines=30");
    CONSOLE_CURSOR_INFO info;
    info.dwSize = 100;
    info.bVisible = false;
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &info);
}
```

Function to draw the screen border

```
void screenBorder()
{
    for (int i = 0; i < WIDTH; i++)
    {
        gotxy(i, 0);
        cout << char(61);
        gotxy(i, HEIGHT - 1);
        cout << char(61);
    }
    for (int i = 0; i < HEIGHT; i++)
    {
        gotxy(0, i);
        cout << char(124);
        gotxy(WIDTH - 1, i);
        cout << char(124);
    }
    gotxy(0, 0);
    cout << char(43);
    gotxy(0, HEIGHT - 1);
    cout << char(43);
    gotxy(WIDTH - 1, 0);
    cout << char(43);
    gotxy(WIDTH - 1, HEIGHT - 1);
    cout << char(43);
}
```

```
void snakegame()
{
    gotxy(18,3); cout <<CYN "000000  00    0      00      0    0  000000";
    gotxy(18,4); cout <<    "0       0 0   0     0  0     0  0    0     ";
    gotxy(18,5); cout <<    "000000  0  0  0    000000    000     000000";
    gotxy(18,6); cout <<    "     0  0   0 0   0      0   0  0    0     ";
    gotxy(18,7); cout <<    "000000  0    00  0        0  0    0  000000" ;

    gotxy(20,9); cout <<     "000000      00      00       00  000000";
    gotxy(20,10); cout <<    "0          0  0     0 0     0 0  0     ";
    gotxy(20,11); cout <<    "0 0000    000000    0  0   0  0  000000";
    gotxy(20,12); cout <<    "0    0   0      0   0   0 0   0  0     ";
    gotxy(20,13); cout <<    "000000  0        0  0    0    0  000000" NC;

}
```

Function to fetch the multiline instruction from instructions.txt

// display it in the center of the screen

```
void Instruction()
{
    ifstream file;
    file.open("instructions.txt");
    string line;
    int y = 16;
    while (getline(file, line))
    {
        int x = ((WIDTH / 2) - (line.length() / 2) - 1);
        gotxy(x, y);
        cout << line;
        y++;
    }
    file.close();
    gotxy(WIDTH / 2, y);
    Controls();
    clrAndStartGame();
}
```

Function to read the controls instructions from controls.txt

// display it in the bottom of game area

```
void Controls()
{
    ifstream file;

  void drawFood()
  {
      gotxy(foodX, foodY);
      cout <<  char(42);
      screenBorder();
  }
        cout << line;
        y++;
    }
    file.close();
    gotxy(WIDTH / 2, y);
}
```

Function to draw snake

```
void drawSnake()
{
    for (int i = 0; i < snakeSize; i++)
    {
        gotxy(snakeX[i], snakeY[i]);
        cout << 'O';
    }
}
```

Function to display the score

```
void drawScore()
{
    gotxy(WIDTH - 15, 0);
    cout << "_Score: " << score <<"_";
}
```

Function to draw the game

```
void drawGame()
{
    Controls();
    drawSnake();
    drawFood();
    drawScore();
}
```

Function to move the snake according to the direction active

```
void moveSnake()
{
    tailX = snakeX[snakeSize - 1];
    tailY = snakeY[snakeSize - 1];
    changeDirection();
    for (int i = snakeSize - 1; i > 0; i--)
    {
        snakeX[i] = snakeX[i - 1];
        snakeY[i] = snakeY[i - 1];
    }
    if (direction == LEFT)
    {
        snakeX[0]--;
    }
    else if (direction == RIGHT)
    {
        snakeX[0]++;
    }
    else if (direction == UP)
    {
        snakeY[0]--;
    }
    else if (direction == DOWN)
    {
        snakeY[0]++;
    }
    drawSnake();
    gotxy(tailX, tailY);
    cout << " ";
}
```

Function to change the direction of the snake when the user presses the arrow keys

//detect the key pressed

```
void changeDirection()
{

    if (_kbhit())
    {
        char ch = _getch();
        if (ch == 'a' || ch == 'A' || ch == KEY_LEFT)
        {
            if (direction != RIGHT)
            {
                direction = LEFT;
            }
        }
        else if (ch == 'd' || ch == 'D' || ch == KEY_RIGHT)
        {
            if (direction != LEFT)
            {
                direction = RIGHT;
            }
        }
        else if (ch == 'w' || ch == 'W' || ch == KEY_UP)
        {
            if (direction != DOWN)
            {
                direction = UP;
            }
        }
        else if (ch == 's' || ch == 'S' || ch == KEY_DOWN)
        {
            if (direction != UP)
            {
                direction = DOWN;
            }
        }
    }
}
```

Function to check if the snake eats the food

// if the snake eats the food:

// increase the score by 1

// increase the size of the snake by 1

// increase the speed

// generate a new food

```
void checkFood()
{
    if (snakeX[0] == foodX && snakeY[0] == foodY)
    {
        score++;
        snakeSize++;
        // increase the speed
        increaseSpeed();
        generateFood();
    }
}
```

Function to generate the food

// generate the food randomly

// if the snake eats the food, new food is generated

```
void generateFood()
{
    foodX = rand() % (WIDTH - 2) + 1;
    foodY = rand() % (HEIGHT - 2) + 1;
    drawFood();
}
```

Function to check if the snake hits the wall or itself

// if the snake hits the wall, the snake passes through the wall and appear on the other side of the screen

```
void checkSnake()
{

    if (snakeX[0] == 0 || snakeX[0] == WIDTH - 1 || snakeY[0] == 0 || snakeY[0] == HEIGHT - 1)
    {
        if (snakeX[0] == 0)
        {
            gameOver = true;
        }
        else if (snakeX[0] == WIDTH - 1)
        {
            gameOver = true;
        }
        else if (snakeY[0] == 0)
        {
            gameOver = true;
        }
        else if (snakeY[0] == HEIGHT - 1)
        {
            gameOver = true;
        }
        screenBorder();
        drawScore();
    }

    for (int i = 1; i < snakeSize; i++)
    {
        if (snakeX[0] == snakeX[i] && snakeY[0] == snakeY[i])
        {
            gameOver = true;
        }
    }
}
```

```
void crew()
{
    gotxy(32,2); cout << "Production  Crew\n";
    gotxy(24,3); cout << "IM/2020/098 - Pravil Wijesinghe";
    gotxy(24,4); cout << "IM/2020/037 - Hashinee Perera";
    gotxy(24,5); cout << "IM/2020/002 - Achintha Alahakoon";
    gotxy(24,6); cout << "IM/2020/096 - Kawya Thathsarani";
    gotxy(24,7); cout << "IM/2020/016 - Bhuddhika Hasitha";
}
```

Function to check if the game is over

// if the game is over

```
void checkGame()
{

    if (gameOver)
    {
        system("cls");
        screenBorder();
        crew();

        gotxy(WIDTH/2-5, HEIGHT/2-1);
        cout << RED "Game Over!";
        gotxy(WIDTH/2-4, HEIGHT/2+1);
        cout << GRN "Score: " << score <<NC;
        Instruction();
        clrAndStartGame();
    }
}
```

Function to clear the screen and start the game with keyboard press

// wait for the user to press a key

// if the user presses the space bar or key 'n', the game starts with gameDelay=200

// if the user presses the key 'h', the game starts with gameDelay=100

// if the user pressed the key 'e', the game starts with gameDelay=300

// if the user presses the escape key, the game quits

```
void clrAndStartGame()
{
    while (true)
    {
        if (_kbhit())
        {
            char ch = _getch();
            if (ch == 32 || ch == 'n')
            {
                system("cls");
                screenBorder();
                gameDelay = 200;
                break;
            }
            else if (ch == 'h')
            {
                system("cls");
                screenBorder();
                gameDelay = 100;
                break;
            }
            else if (ch == 'e')
            {
                system("cls");
                screenBorder();
                gameDelay = 300;
                break;
            }
            else if (ch == 27)
            {
                exit(0);
            }
        }
    }
    startGame();
}
```

Function to increase the speed of the game

// if the gameDelay is greather than 200, decrease the gameDelay by 8

// else if the gameDelay is greater than 100, decrease the gameDelay by 7

// else if the gameDelay is greater than 50, decrease the gameDelay by 4

// else if the gameDelay is greater than 40, decrease the gameDelay by 3

```
void increaseSpeed()
{
    if (gameDelay > 200)
    {
        gameDelay -= 8;
    }
    else if (gameDelay > 100)
    {
        gameDelay -= 7;
    }
    else if (gameDelay > 50)
    {
        gameDelay -= 4;
    }
    else if (gameDelay > 40)
    {
        gameDelay -=3;
    }
}
```

Function to control the game

```
void controlGame()
{

    moveSnake();

    checkFood();

    checkSnake();

    checkGame();
}
```

Function to start the game

```
void startGame()
{

    direction = RIGHT;

    snakeX[0] = WIDTH / 2;
    snakeY[0] = HEIGHT / 2;

    foodX = (rand() % (WIDTH - 2)) + 1;
    foodY = (rand() % (HEIGHT - 2)) + 1;

    score = 0;

    snakeSize = 1;

    gameOver = false;

    while (!gameOver)
    {
        controlGame();

        Sleep(gameDelay);

        drawGame();
    }
}
```

```
int main()
{
    ScreenSize();

    screenBorder();

    snakegame();

    Instruction();

    startGame();

}
```

Group Members

IM/2020/098 - Pravil Wijesinghe

IM/2020/037 – Hashinee Perera

IM/2020/002 - Achintha Alahakoon

IM/2020/096 - Kawya Thathsarani

IM/2020/016 - Bhuddhika Hasitha