# SNOW BROS

**Session 2022 – 2026**

**Submitted by:**

Tayyaba Afzal 2022-CS-134

**Supervised by:**

Dr. Awai s Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

## Table of Contents

## 1. Short Description of Project

Nick is the main character who lives in the world filled with mazes, enemies, money, and food. Nick is tasked with travelling through different stages, throwing, and building snowballs, jumping on and off platforms to navigate level obstacles while dodging and defeating monsters to rescue the princess. In this journey, Nick faces different difficulties caused his enemies named Elites, Letho, Dark Gannon and Death claws. These enemies terrorize the maze and always lurk around every corner, trying to catch Nick.

Nick can throw snow at enemies until the die because of cold. Nick eventually succeeds to kill all enemies and save the princess. He becomes a hero, known for his bravery, determination, and never-say-die spirit. The enemies may try to stop him time and time again, but Nick always perseveres and triumphs in the end.

## 2. Game Characters Description.

### 2.1 Player

There is one human player in the Game.

**Nick:**

Nick is the main character in the game and is known for his blue and white, robotic shape. Nick is brave, determined, and has a never-say-die spirit. He is the hero of the game, admired for his bravery and determination in the face of danger.

### 2.2 Enemies

There are 4 enemies in the game.

**i. Death-Claws:**

Death Claws is one of the four evil enemies in the game and is known for being aggressive and difficult to shake. He is always chasing Nick through the maze, trying to catch him at every turn. Blinky is fast and relentless, making him one of the most dangerous foes that Nick must face.

**ii. Dark-Gannon:**

Dark Gannon is another of the evil enemies and is known for her unpredictable movements and moves randomly in the game.

**iii. Elites:**

Elites is one of the four evil enemies in the game and is known for his vertical movement in the game.

### iv. Letho:

Letho is the final of the four evil enemies and is known for his horizontal movement in the game.

## 3. Game Object Description

Following are the Objects in the Game

### i. Money:

Some money appears in every level after specific time. When Lick collide with this money, he gets increment in the score according to the money. This money comes for some specific time, after that time this money will disappear.

### ii. Food:

There is also some food in the game that also appear for specific time. If Lick does not collide with it in that time that food will disappear. And score will increment according to food.

### iii. Slides:

There are some slides in the game on which Lick, and enemies can walk.

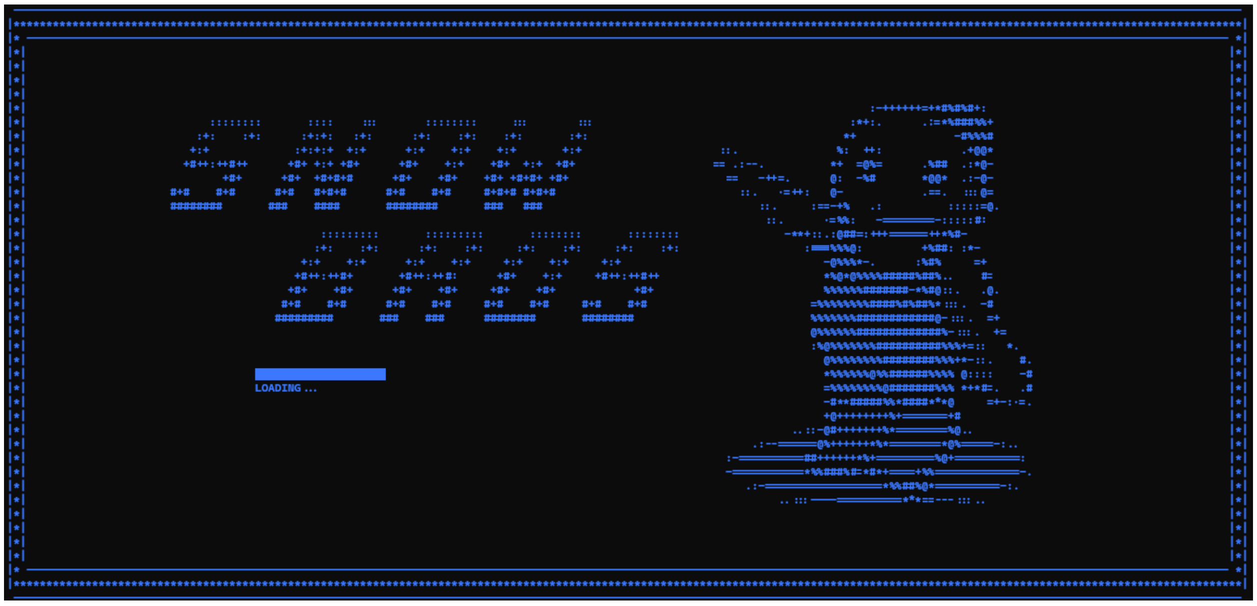
## 4. Rules & Interactions

Lick can eat food pallets that have been put across the maze. Lick loses a life if he collides with any of the enemies. If Lick eats Power Pallets, then Lick can touch the enemies as well. Score increases when the Lick eats food, money, or power pallets.

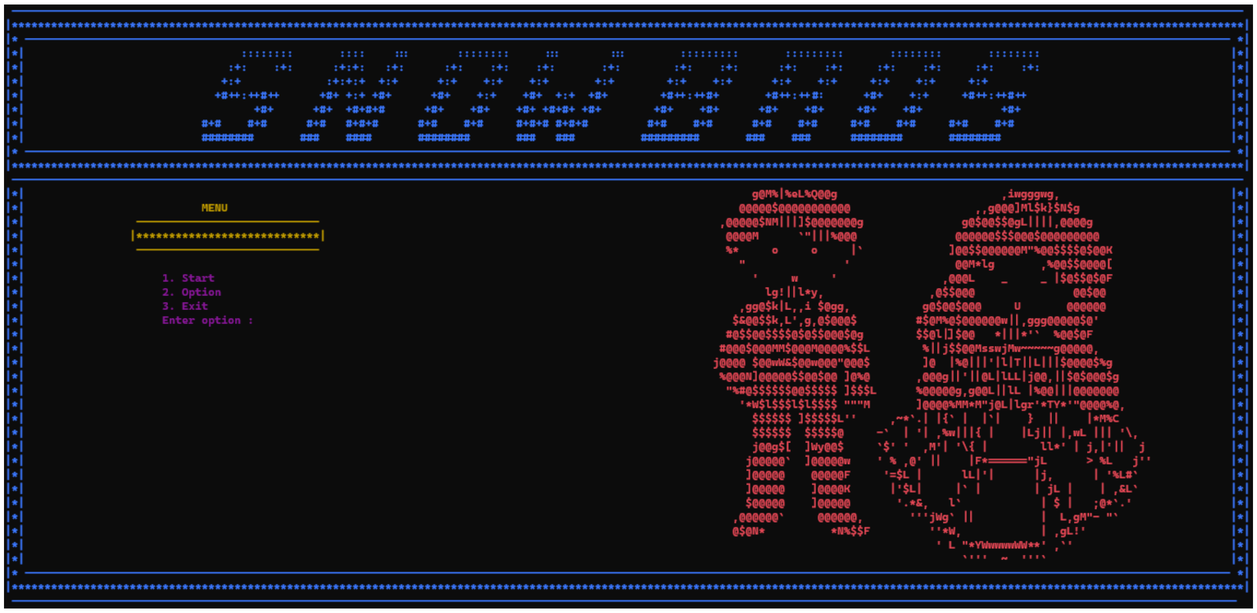
## 5. Goal of the Game

The goal of the game is to kill all of the enemies that have been put across the maze.

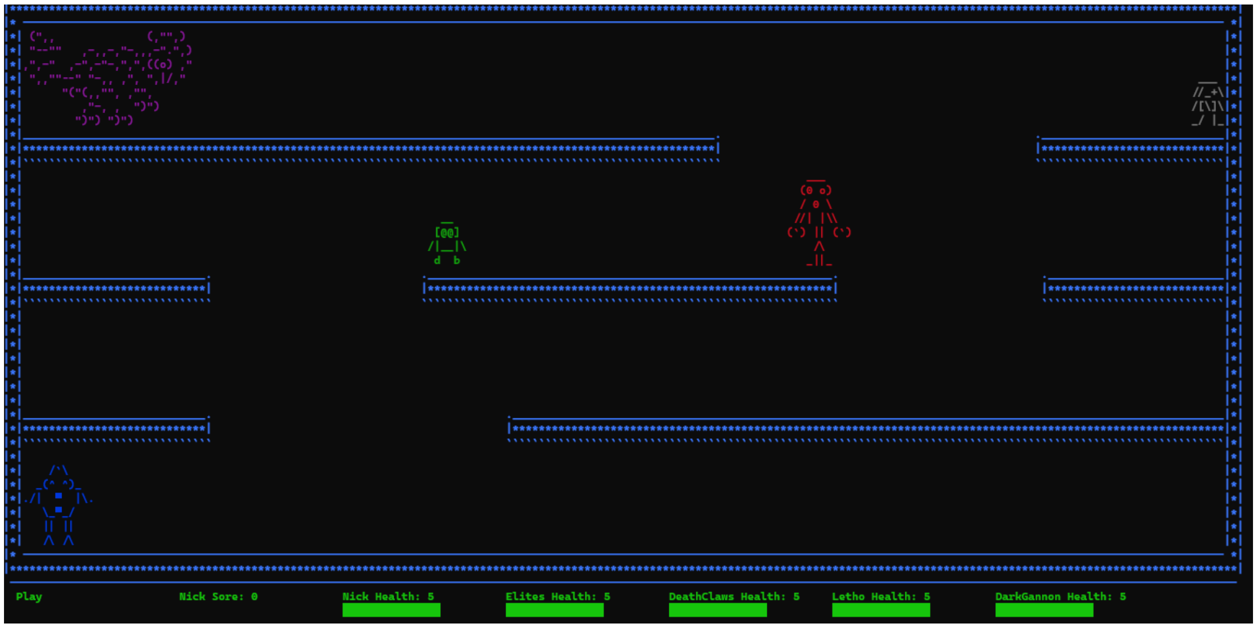
## 6. Wireframes



### 6.1 Game Start Menu

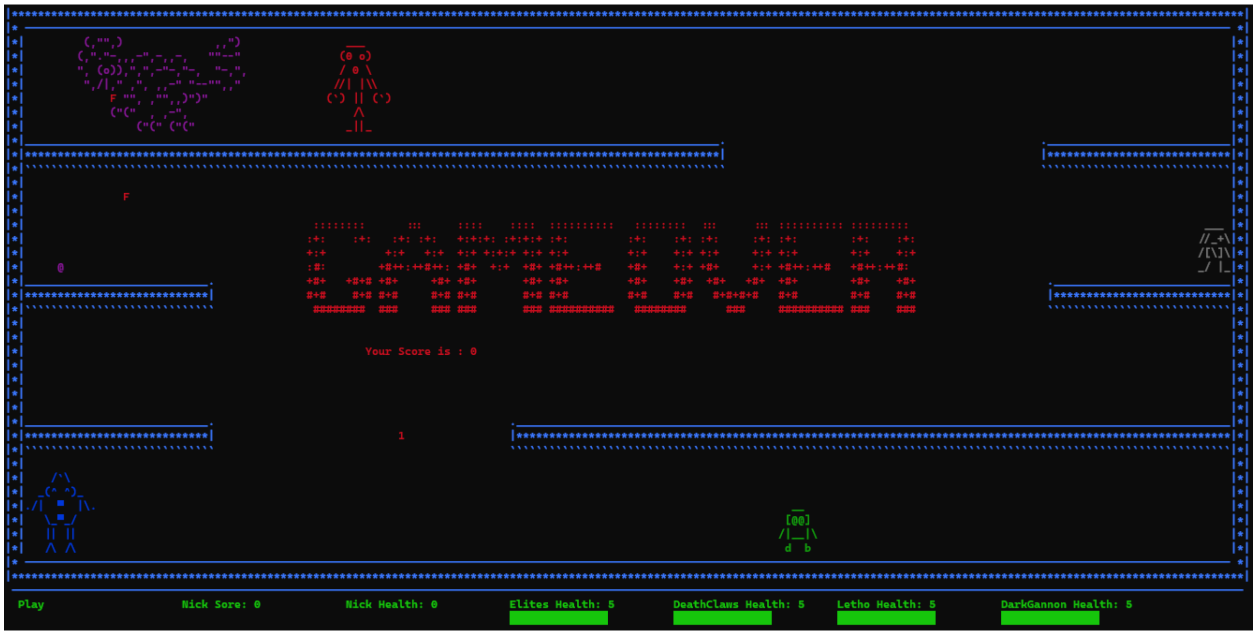


### 6.2 Option Menu



```
Play          Nick Sore: 0          Nick Health: 5          Elites Health: 5          DeathClaws Health: 5          Letho Health: 5          DarkGannon Health: 5
```

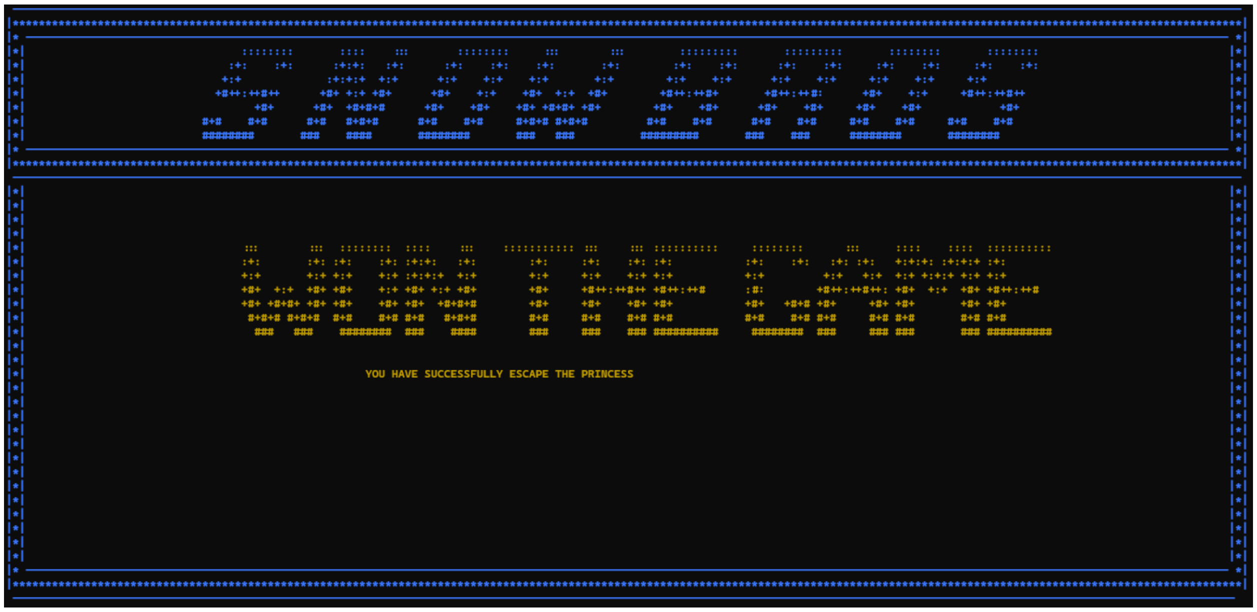
**6.3 Game**



```
Your Score is : 0
```

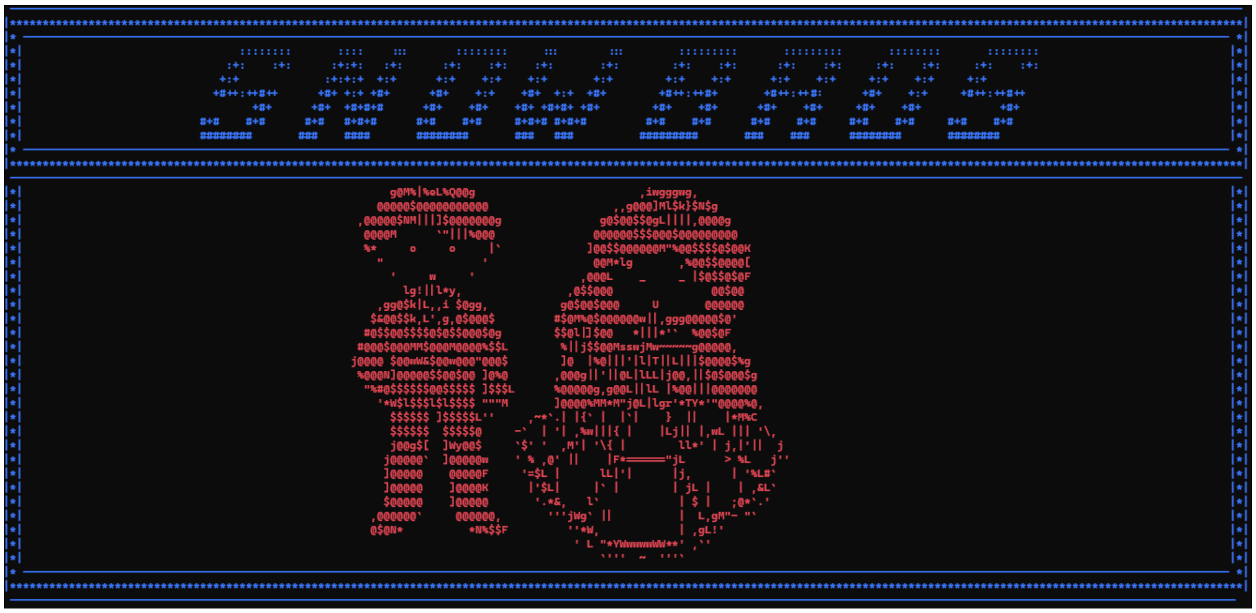
```
Play          Nick Sore: 0          Nick Health: 0          Elites Health: 5          DeathClaws Health: 5          Letho Health: 5          DarkGannon Health: 5
```

**6.4 Game Over Menu**

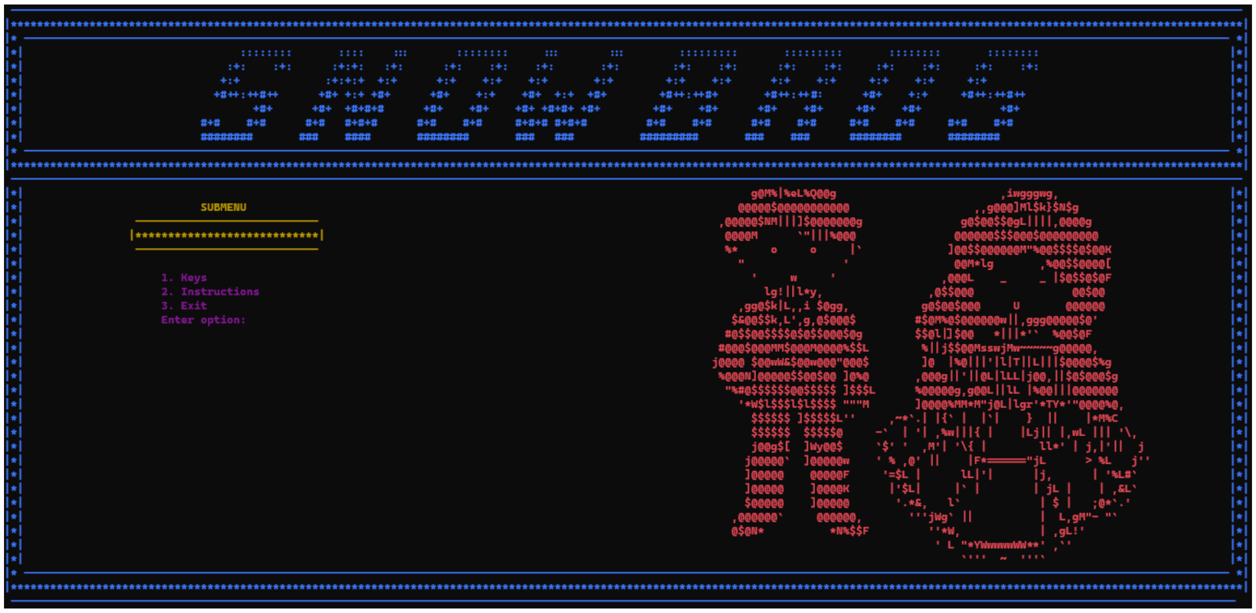


```
YOU HAVE SUCCESSFULLY ESCAPE THE PRINCESS
```

### 6.5 Game Won Menu



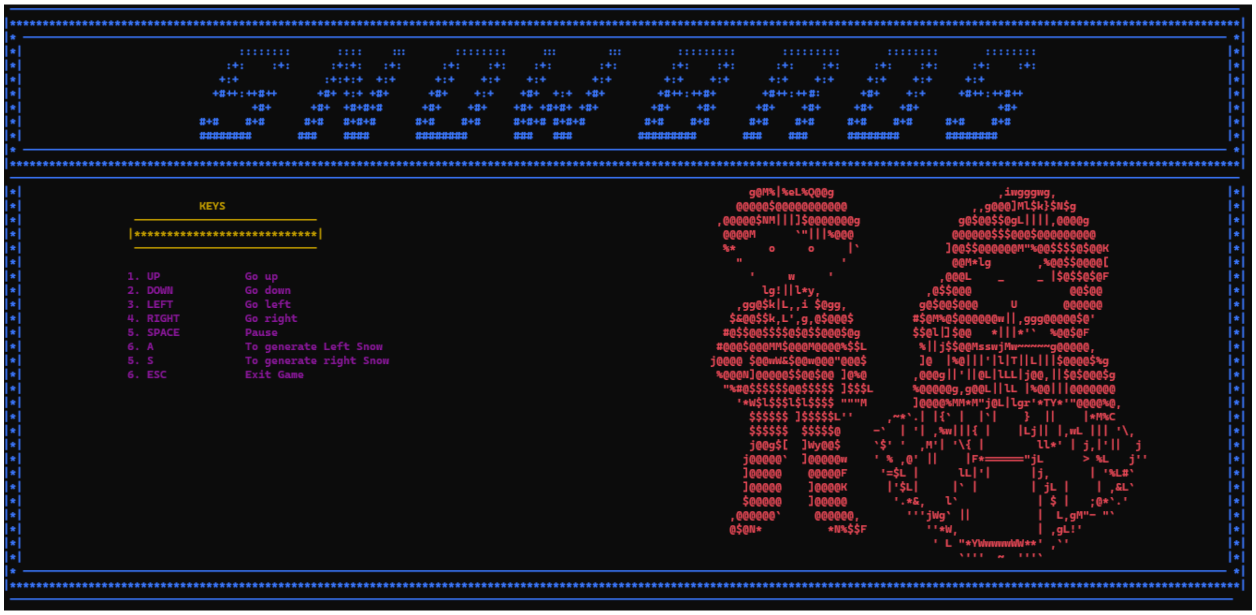
### 6.6 Escape Princess Menu



```
SUBMENU
|****************************|

1. Keys
2. Instructions
3. Exit
Enter option:
```

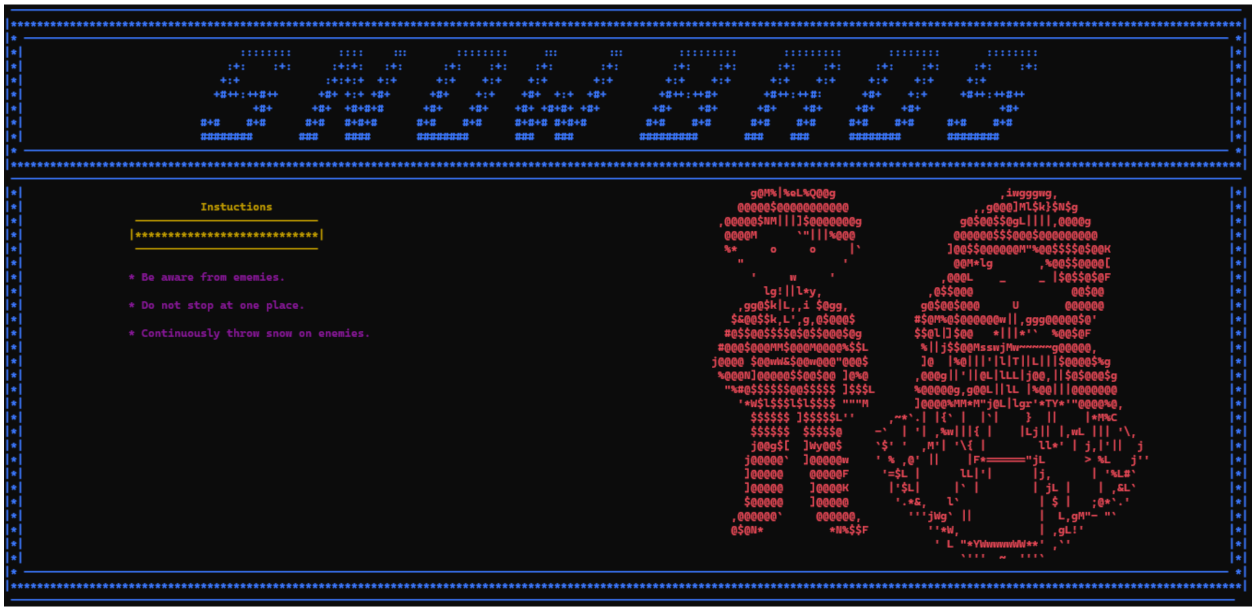
## 6.7 Option Sub Menu



```
KEYS
|****************************|

1. UP           Go up
2. DOWN         Go down
3. LEFT         Go left
4. RIGHT        Go right
5. SPACE        Pause
6. A            To generate Left Snow
5. S            To generate right Snow
6. ESC          Exit Game
```

## 6.8 Keys Menu



```
Instuctions
|****************************|

* Be aware from enemies.

* Do not stop at one place.

* Continuously throw snow on enemies.
```

**6.9 Instuctions menu**

## 7. Data Structures

char nick[6][11]

char elites01[7][26]

char elites02[7][26]

char letho[4][5]

char deathClaws[7][10]

char darkGannon[4][6]

## 8. Function Prototypes

void printDisplay();

void printLogo();

void showMenu();

void printMenu(string);

```
void showSubMenu();
void printKeys();
void printInstructions();
void printMaze();
bool printGameOver();
void printGameWon();
void printMoney();
void printFood();
void printNickAndPrincess(int x,int y);
void defineHealthAndScore();
void printHealthAndScore();

void printNick();
void printElites();
void printReverseElites();
void printLetho();
void printDeathClaws();
void printDarkGannon();
void eraseNick();
void eraseElites();
void eraseLetho();
void eraseDeathClaws();
void eraseDarkGannon();

void moveNickLeft();
void moveNickRight();
void moveNickUp();
void moveNickDown();
void gravityForNick();
void moveElites();
void moveLetho();
void moveDarkGannon();
void moveDeathClaws();
void moveDeathClawsDown();

void printSnowRight(int x, int y);
void eraseSnow(int x, int y);
void generateSnowRight();
void moveSnowRight();
void removeRightSnowFromArray(int index);
void printSnowLeft(int x, int y);
void generateSnowLeft();
void moveSnowLeft();
void removeLeftSnowFromArray(int index);

void printFire(int fireX, int fireY);
void eraseFire(int fireX, int fireY);
void generateElitesFire();
void moveElitesFire();
void generateLethoFire();
void moveLethoFire();
void generateDarkGannonFire();
void moveDarkGannonFire();
```

void collisionOfNickAndEnemies(int playerX , int playerY , int playerWidth , int playerHeight , int enemyX , int enemyY , int enemyWidth , int enemyHeight);

void collisionOfSnowWithElites();

void collisionOfSnowWithDeathClaws();

void collisionOfSnowWithLetho();

void collisionOfSnowWithDarkGannon();

void collisionOfFireAndNick();

void collisionOfFoodOrMoney();

void storeScoreAndHealth();

void loadScoreAndHealth();

string getfield(string line, int field);

void gotoxy(int x, int y);

char getCharAtxy(short int x, short int y);

void ShowConsoleCursor(bool showFlag);

## 9. Complete Code

```
#include <iostream>
#include <windows.h>
#include <fstream>
#include <conio.h>
using namespace std;

HANDLE color = GetStdHandle(STD_OUTPUT_HANDLE);

int option;
int x, y;
char box = 223;
int nickX = 3, nickY = 33;
int elitesX = 3, elitesY = 2;
int lethoX = 182, lethoY = 5;
int deathClawsX = 120, deathClawsY = 12;
int darkGannonX = 65, darkGannonY = 15;

string elitesDirection = "Right";
string lethoDirection = "Down";
string darkGannonDirection = "Right";

int rightSnowX[100];
int rightSnowY[100];
int leftSnowX[100];
int leftSnowY[100];
int rightSnowCount = 0;
int leftSnowCount = 0;

int elitesFireX[100];
int elitesFireY[100];
int lethoFireX[100];
int lethoFireY[100];
int darkGannonFireX[100];
int darkGannonFireY[100];
int elitesFireCount = 0;
int lethoFireCount = 0;
int darkGannonFireCount = 0;

int score = 0;
int nickHealth = 5;
int elitesHealth =  5;
int deathClawsHealth = 5;
int lethoHealth = 5;
int darkGannonHealth = 5;
int lethoCounter = 0;
int darkGannonCounter = 0;
int deathClawsCounter = 0;
int elitesCounter = 0;
int foodCount = 0;
int moneyCount = 0;
int timer = 0;

char nick[6][11] = {{' ', ' ', ' ', ' ', '/', '`', '\\', ' ', ' ', ' ', ' '},
                    {' ', ' ', '_', '(', '^', ' ', '^', ')', '_', ' ', ' '},
                    {'.', '/', '|', ' ', ' ', box, ' ', ' ', '|', '\\', '.'},
                    {' ', ' ', ' ', '\\', '_', box, '_', '/', ' ', ' ', ' '},
                    {' ', ' ', ' ', '|', '|', ' ', '|', '|', ' ', ' ', ' '},
                    {' ', ' ', ' ', '/', '\\', ' ', '/', '\\', ' ', ' ', ' '}};

char elites01[7][26] = {{' ', '(', ',', '"', '"', ',', ')', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ',', ',', '"', ')', ' '},
                        {'(', ',', '"', '.', '"', '-', ',', ',', ',', '-', '"', ',', '-', ',', ',', '-', ',', ' ', ' ', ' ', '"', '"', '-', '-', '"', ' '},
                        {'"', ',', ' ', '(', 'o', ')', ')', ',', '"', ',', '"', ',', '-', '"', '-', ',', '"', '-', ',', ' ', ' ', '"', '-', ',', '"', ','},
                        {' ', '"', ',', '/', '|', ',', '"', ' ', ',', '"', ',', ' ', ',', ',', '-', '"', ' ', '"', '-', '-', '"', '"', ',', ',', '"', ' '},
                        {' ', ' ', ' ', ' ', ' ', ' ', ',', '"', '"', ',', ' ', ',', '"', '"', ',', ',', ')', '"', ')', '"', ' ', ' ', ' ', ' ', ' ', ' '},
                        {' ', ' ', ' ', ' ', ' ', '(', '"', '(', '"', ' ', ' ', ',', ' ', ',', '-', '"', ',', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},
                        {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '(', '"', '(', '"', ' ', '(', '"', '(', '"',' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '}};

char elites02[7][26] = {{' ', ')', ',', '"', '"', ',', '(', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ',', ',', '"', '(', ' '},
                        {')', ',', '"', '.', '"', '-', ',', ',', ',', '-', '"', ',', '-', ',', ',', '-', ',', ' ', ' ', ' ', '"', '"', '-', '-', '"', ' '},
                        {'"', ',', ' ', ')', 'o', '(', '(', ',', '"', ',', '"', ',', '-', '"', '-', ',', '"', '-', ',', ' ', ' ', '"', '-', ',', '"', ','},
                        {' ', '"', ',', '/', '|', ',', '"', ' ', ',', '"', ',', ' ', ',', ',', '-', '"', ' ', '"', '-', '-', '"', '"', ',', ',', '"', ' '},
                        {' ', ' ', ' ', ' ', ' ', ' ', ',', '"', '"', ',', ' ', ',', '"', '"', ',', ',', '(', '"', '(', '"', ' ', ' ', ' ', ' ', ' ', ' '},
                        {' ', ' ', ' ', ' ', ' ', ')', '"', ')', '"', ' ', ' ', ',', ' ', ',', '-', '"', ',', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '},
                        {' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ')', '"', ')', '"', ' ', ')', '"', ')', '"',' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '}};

char letho[4][5] = {{' ', '_', '_', '_', ' '},
                    {'/', '/', '_', '+', '\\'},
                    {'/', '[', '\\', ']', '\\'},
                    {'_', '/', ' ', '|', '_'}};

char deathClaws[7][10] = {{' ', ' ', ' ', '_', '_', '_', ' ', ' ', ' ', ' '},
                          {' ', ' ', '(', '0', ' ', 'o', ')', ' ', ' ', ' '},
                          {' ', ' ', '/', ' ', '0', ' ', '\\', ' ', ' ', ' '},
                          {' ', '/', '/', '|', ' ', '|', '\\', '\\', ' ', ' '},
                          {'(', '`', ')', ' ', '|', '|', ' ', '(', '`', ')'},
                          {' ', ' ', ' ', ' ', '/', '\\', ' ', ' ', ' ', ' '},
                          {' ', ' ', ' ', '_', '|', '|', '_', ' ', ' ', ' '}};

char darkGannon[4][6] = {{' ', ' ', '_', '_', ' ', ' '},
                         {' ', '[', '@', '@', ']', ' '},
                         {'/', '|', '_', '_', '|', '\\'},
                         {' ', 'd', ' ', ' ', 'b', ' '}};

void printDisplay();
void printLogo();
void showMenu();
void printMenu(string);
void showSubMenu();
void printKeys();
void printInstructions();
void printMaze();
bool printGameOver();
void printGameWon();
void printMoney();
void printFood();
void printNickAndPrincess(int x,int y);
void defineHealthAndScore();
void printHealthAndScore();

void printNick();
void printElites();
void printReverseElites();
void printLetho();
void printDeathClaws();
void printDarkGannon();
void eraseNick();
void eraseElites();
void eraseLetho();
void eraseDeathClaws();
void eraseDarkGannon();

void moveNickLeft();
void moveNickRight();
void moveNickUp();
void moveNickDown();
void gravityForNick();
void moveElites();
void moveLetho();
void moveDarkGannon();
void moveDeathClaws();
void moveDeathClawsDown();

void printSnowRight(int x, int y);
void eraseSnow(int x, int y);
void generateSnowRight();
void moveSnowRight();
void removeRightSnowFromArray(int index);
void printSnowLeft(int x, int y);
void generateSnowLeft();
void moveSnowLeft();
void removeLeftSnowFromArray(int index);

void printFire(int fireX, int fireY);
void eraseFire(int fireX, int fireY);
void generateElitesFire();
void moveElitesFire();
void generateLethoFire();
void moveLethoFire();
void generateDarkGannonFire();
void moveDarkGannonFire();

void collisionOfNickAndEnemies(int playerX , int playerY , int playerWidth , int playerHeight , int enemyX , int enemyY , int enemyWidth , int enemyHeight);
void collisionOfSnowWithElites();
void collisionOfSnowWithDeathClaws();
void collisionOfSnowWithLetho();
void collisionOfSnowWithDarkGannon();
void collisionOfFireAndNick();
void collisionOfFoodOrMoney();

void storeScoreAndHealth();
void loadScoreAndHealth();
string getfield(string line, int field);

void gotoxy(int x, int y);
char getCharAtxy(short int x, short int y);
void ShowConsoleCursor(bool showFlag);

main()
{
    int count = 0;
    string menu;
    bool flag = true;
    bool mutualFlag;
    bool infoFlag;
    bool gameFlag;
    bool keysFlag;
    bool instructionFlag;
    char alphabet;
    ShowConsoleCursor(false);
    while (flag)
    {
        system("cls");
        printDisplay();
        mutualFlag = true;
        while (mutualFlag)
        {
            menu = "MENU";
            system("cls");
            printLogo();
            printNickAndPrincess(100,13);
            printMenu(menu);
            showMenu();
            cin >> option;
            if (option == 1)
            {
                gameFlag = true;
                system("cls");
                printMaze();
                printNick();
                printReverseElites();
                printLetho();
                printDarkGannon();
                printDeathClaws();
                printHealthAndScore();
                loadScoreAndHealth();
                getch();
                while (gameFlag)
                {
                    if (GetAsyncKeyState(VK_LEFT))
                    {
                        moveNickLeft();
                        gravityForNick();
                    }
                    if (GetAsyncKeyState(VK_RIGHT))
                    {
                        moveNickRight();
                        gravityForNick();
                    }
                    if (GetAsyncKeyState(VK_UP))
                    {
                        moveNickUp();
                    }
                    if (GetAsyncKeyState(VK_DOWN))
                    {
                        moveNickDown();
                        gravityForNick();
                    }
                    if (GetAsyncKeyState('S'))
                    {
                        generateSnowRight();
                    }
                    if (GetAsyncKeyState('A'))
                    {
                        generateSnowLeft();
                    }
                    if (GetAsyncKeyState(VK_SPACE))
                    {
                        gotoxy(2, 42);
                        cout << "Pause";
                        getch();
                        cin >> alphabet;
                        cout << " ";
                    }
                    if (GetAsyncKeyState(VK_ESCAPE))
                    {
                        gameFlag = false;
                    }
                    if (timer == 3)
                    {
```

```
        if(lethoHealth > 0)
        {
            moveLetho();
        }
        if(lethoHealth <= 0 && lethoCounter == 0)
        {
            eraseLetho();
            lethoCounter++;
        }
        if(darkGannonHealth >= 0)
        {
            moveDarkGannon();
        }
        if(darkGannonHealth <= 0 && darkGannonCounter == 0)
        {
            eraseDarkGannon();
            darkGannonCounter++;
        }
        if(deathClawsHealth >= 0)
        {
            moveDeathClaws();
            moveDeathClawsDown();
        }
        if(deathClawsHealth <= 0 && deathClawsCounter == 0)
        {
            eraseDeathClaws();
            deathClawsCounter++;
        }
        timer = 0;
    }
    if(elitesHealth >= 0)
    {
        moveElites();
    }
    if(elitesHealth <= 0 && elitesCounter == 0)
    {
        eraseElites();
        elitesCounter++;
    }
    if (count == 10 && elitesHealth > 0)
    {
        generateElitesFire();
    }
    if (count == 20 && lethoHealth > 0)
    {
        generateLethoFire();
    }
    if (count == 30 && darkGannonHealth > 0)
    {
        generateDarkGannonFire();
    }
    if (count == 50)
    {
        printMoney();
    }
    if (count == 100)
    {
        printFood();
        count = 0;
    }
    moveSnowRight();
    moveSnowLeft();
    moveElitesFire();
    moveLethoFire();
    moveDarkGannonFire();
    collisionOfFireAndNick();
    collisionOfFoodOrMoney();
    collisionOfNickAndEnemies(nickX, nickY, 11, 6, deathClawsX, deathClawsY, 10, 7);
    collisionOfNickAndEnemies(nickX, nickY, 11, 6, darkGannonX, darkGannonY, 6, 4);
    collisionOfNickAndEnemies(nickX, nickY, 11, 6, lethoX, lethoY, 5, 4);
    collisionOfNickAndEnemies(nickX, nickY, 11, 6, elitesX, elitesY, 26, 7);
    collisionOfSnowWithElites();
    collisionOfSnowWithDeathClaws();
    collisionOfSnowWithLetho();
    collisionOfSnowWithDarkGannon();
    printHealthAndScore();
    storeScoreAndHealth();
    if(nickHealth <= 0 && (deathClawsHealth >= 0 || elitesHealth >= 0 || darkGannonHealth >= 0 || lethoHealth >= 0))
    {
        printGameOver();
        defineHealthAndScore();
        gotoxy(60,30);
        cin >> alphabet;
        while(alphabet != 'z')
            {
                cin >> alphabet;
                if(alphabet == 'z')
                {
                    gameFlag = false;
                    break;
                }
            }
    }
    if(nickHealth > 0 && (deathClawsHealth <= 0 && elitesHealth <= 0 && darkGannonHealth <= 0 && lethoHealth <= 0))
    {
        system("cls");
        printLogo();
        printGameWon();
        gotoxy(60,30);
        cin >> alphabet;
        if(alphabet == 'z')
        {
            system("cls");
            printLogo();
            printNickAndPrincess(45,13);
            defineHealthAndScore();
            gotoxy(60,30);
            cin >> alphabet;
            while(alphabet != 'z')
            {
```

```
                                cin >> alphabet;
                                if(alphabet == 'z')
                                {
                                    gameFlag = false;
                                    break;
                                }
                            }
                        }
                    }
                    timer++;
                    count++;
                    Sleep(50);

                }
            }
            else if (option == 2)
            {
                infoFlag = true;
                while (infoFlag)
                {
                    menu = "SUBMENU";
                    system("cls");
                    printLogo();
                    printNickAndPrincess(100,13);
                    printMenu(menu);
                    showSubMenu();
                    cin >> option;
                    if (option == 1)
                    {
                        keysFlag = true;
                        while(keysFlag)
                        {
                            menu = "KEYS";
                            system("cls");
                            printLogo();
                            printNickAndPrincess(100,13);
                            printMenu(menu);
                            printKeys();
                            getch();
                            keysFlag = false;
                        }
                    }
                    else if (option == 2)
                    {
                        instructionFlag = true;
                        while(instructionFlag)
                        {
                            menu = "Instuctions";
                            system("cls");
                            printLogo();
                            printNickAndPrincess(100,13);
                            printMenu(menu);
                            printInstructions();
                            getch();
                            instructionFlag = false;
                        }
                    }
                    else if (option == 3)
                    {
                        infoFlag = false;
                    }
                }
            }
            else if (option == 3)
            {
                system("cls");
                mutualFlag = false;
                flag = false;
            }
        }
    }
}
void printDisplay()
{
    SetConsoleTextAttribute(color,9);
    char box = 219;
    string line;
    string display[43];
    fstream file;
    file.open("display.txt", ios ::in);
    int idx = 0;
    while (!file.eof())
    {
        getline(file,line);
        display[idx] = line;
        idx++;
    }
    file.close();
    for (int rows = 0; rows < 43; rows++)
    {
        cout << display[rows];
        cout << endl;
    }
    int x = 38, y = 26;
    gotoxy(x, y);
    for(int idx = 0; idx < 20; idx++)
    {
        cout << box;
        x = x + 1;
        gotoxy(x, y);
        Sleep(100);
    }
    getch();
}
void printLogo()
{
    SetConsoleTextAttribute(color,9);
    string line;
    string logo[43];
    fstream file;
    file.open("logo.txt", ios ::in);
    int idx = 0;
    while (!file.eof())
    {
        getline(file,line);
        logo[idx] = line;
        idx++;
    }
    file.close();
    for (int rows = 0; rows < 43; rows++)
    {
        cout << logo[rows];
        cout << endl;
    }
}
void printMenu(string menu)
{
    SetConsoleTextAttribute(color,6);
    gotoxy(30, 14);
    cout << menu << endl;
    x = 19, y = 15;
    gotoxy(x, y);
    y = y + 1;
    cout << " ----------------------------" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "|********************************|" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << " ----------------------------" << endl;
}
void showMenu()
{
    SetConsoleTextAttribute(color,5);
    x = 24, y = 19;
    gotoxy(x, y);
    y = y + 1;
    cout << "1. Start" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "2. Option" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "3. Exit" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "Enter option : ";
}
void showSubMenu()
{
    SetConsoleTextAttribute(color,5);
    x = 24, y = 19;
    gotoxy(x, y);
    y = y + 1;
    cout << "1. Keys" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "2. Instructions" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "3. Exit " << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "Enter option: ";
}
void printKeys()
{
    SetConsoleTextAttribute(color,5);
    x = 19, y = 19;
    gotoxy(x, y);
    y = y + 1;
    cout << "1. UP             Go up" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "2. DOWN           Go down " << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "3. LEFT           Go left " << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "4. RIGHT          Go right" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "5. SPACE          Pause" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "6. A              To generate Left Snow" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "5. S              To generate right Snow" << endl;
    gotoxy(x, y);
    y = y + 1;
    cout << "6. ESC            Exit Game" << endl;
}
void printInstructions()
{
    SetConsoleTextAttribute(color,5);
    x = 19, y = 19;
    gotoxy(x, y);
    y = y + 2;
    cout << "* Be aware from ememies." << endl;
    gotoxy(x, y);
    y = y + 2;
    cout << "* Do not stop at one place." << endl;
    gotoxy(x, y);
    y = y + 2;
    cout << "* Continuously throw snow on enemies. " << endl;
}
void printMaze()
{
    SetConsoleTextAttribute(color,9);
    string line;
    string maze[43];
    fstream file;
    file.open("maze.txt", ios ::in);
    int idx = 0;
    while (!file.eof())
    {
        getline(file,line);
        maze[idx] = line;
        idx++;
    }
    file.close();
    for (int idx = 0; idx < 43; idx++)
    {
        cout << maze[idx];
        cout << endl;
    }
}
bool printGameOver()
{
    SetConsoleTextAttribute(color,4);
    int x = 45 ,y = 15;
    gotoxy(x,y);
    cout << "  :::::::      :::     ::::    ::::  :::::::::   :::::::  :::     ::: :::::::::: :::::::::  "<< endl;
    y = y + 1;
    gotoxy(x,y);
    cout << " :+:    :+:   :+: :+:   +:+:+: :+:+:+ :+:         :+:    :+: :+:     :+: :+:        :+:    :+: "<< endl;
    y = y + 1;
    gotoxy(x,y);
    cout << " +:+         +:+   +:+  +:+ +:+:+ +:+ +:+         +:+    +:+ +:+     +:+ +:+        +:+    +:+ "<< endl;
    y = y + 1;
    gotoxy(x,y);
    cout << " :#:        +#++:++#++: +#+  +:+  +#+ +#++:++#    +#+    +:+ +#+     +:+ +#++:++#   +#++:++#:  "<< endl;
    y = y + 1;
    gotoxy(x,y);
    cout << " +#+   +#+# +#+     +#+ +#+       +#+ +#+         +#+    +#+  +#+   +#+  +#+        +#+    +#+ "<< endl;
    y = y + 1;
    gotoxy(x,y);
    cout << " #+#    #+# #+#     #+# #+#       #+# #+#         #+#    #+#   #+#+#+#   #+#        #+#    #+# "<< endl;
    y = y + 1;
    gotoxy(x,y);
    cout << "  ########  ###     ### ###       ### ##########   ########      ###     ########## ###    ### "<< endl;
    gotoxy(x + 10,y + 3);
    cout << "Your Score is : " << score;
    getch();
    return true;
}
void printMoney()
{
    SetConsoleTextAttribute(color,10);
    if(moneyCount == 1)
    {
        gotoxy(8,18);
        cout << "$";
    }
    if(moneyCount == 2)
    {
        gotoxy(81,18);
        cout << "$";
    }
    if(moneyCount == 3)
    {
        gotoxy(112,28);
        cout << "$";
    }
    if(moneyCount == 4)
    {
        gotoxy(25,38);
        cout << "$";
    }
    if(moneyCount == 5)
    {
        gotoxy(127,38);
        cout << "$";
        moneyCount = -1;
    }
    moneyCount++;
}
void printFood()
{
    if(foodCount == 1)
    {
        gotoxy(8,18);
        cout << "@";
    }
    if(foodCount == 2)
    {
        gotoxy(81,18);
        cout << "@";
    }
    if(foodCount == 3)
    {
        gotoxy(112,28);
        cout << "@";
    }
    if(foodCount == 4)
    {
        gotoxy(25,38);
        cout << "@";
    }
    if(foodCount == 5)
    {
        gotoxy(127,38);
        cout << "@";
        foodCount = -1;
    }
    foodCount++;
}
void printNickAndPrincess(int x,int y)
```

```
{
    SetConsoleTextAttribute(color,12);
    string line;
    string picture[27];
    fstream file;
    file.open("nick&Princess.txt", ios ::in);
    int idx = 0;
    while (!file.eof())
    {
        getline(file,line);
        picture[idx] = line;
        idx++;
    }
    file.close();
    gotoxy(x,y);
    for (int rows = 0; rows < 27; rows++)
    {
        cout << picture[rows];
        cout << endl;
        y = y + 1;
        gotoxy(x,y);
    }
}
void printGameWon()
{
    SetConsoleTextAttribute(color,6);
    int x = 35 ,y = 17;
    gotoxy(x,y);
    cout << " :::       :::  ::::::::  ::::    :::    ::::::::::: :::    ::: ::::::::::     ::::::::      :::     ::::    ::::  :::::::::: " << endl;
    y = y + 1;
    gotoxy(x,y);
    cout << " :+:       :+: :+:    :+: :+:+:   :+:        :+:     :+:    :+: :+:           :+:    :+:   :+: :+:   +:+:+: :+:+:+ :+:        " << endl;
    y = y + 1;
    gotoxy(x,y);
    cout << " +:+       +:+ +:+    +:+ :+:+:+  +:+        +:+     +:+    +:+ +:+           +:+         +:+   +:+  +:+ +:+:+ +:+ +:+        " << endl;
    y = y + 1;
    gotoxy(x,y);
    cout << " +#+  +:+  +#+ +#+    +:+ +#+ +:+ +#+        +#+     +#++:++#++ +#++:++#      :#:        +#++:++#++: +#+  +:+  +#+ +#++:++#   " << endl;
    y = y + 1;
    gotoxy(x,y);
    cout << " +#+ +#+#+ +#+ +#+    +#+ +#+  +#+#+#        +#+     +#+    +#+ +#+           +#+   +#+# +#+     +#+ +#+       +#+ +#+        " << endl;
    y = y + 1;
    gotoxy(x,y);
    cout << "  #+#+# #+#+#  #+#    #+# #+#   #+#+#        #+#     #+#    #+# #+#           #+#    #+# #+#     #+# #+#       #+# #+#        " << endl;
    y = y + 1;
    gotoxy(x,y);
    cout << "   ###   ###    ########  ###    ####        ###     ###    ### ##########     ########  ###     ### ###       ### ########## " << endl;
    gotoxy(x + 20,y + 3);
    cout << "YOU HAVE SUCCESSFULLY ESCAPE THE PRINCESS";
    getch();
}
void defineHealthAndScore()
{
    score = 0;
    nickHealth = 5;
    elitesHealth =  5;
    deathClawsHealth = 5;
    lethoHealth = 5;
    darkGannonHealth = 5;
    lethoCounter = 0;
    darkGannonCounter = 0;
    deathClawsCounter = 0;
    elitesCounter = 0;
}
void printHealthAndScore()
{
    SetConsoleTextAttribute(color,10);
    char box = 219;
    int x = 2 ,y = 42;
    gotoxy(x, y);
    cout << "            ";
    gotoxy(x, y);
    cout << "Play";
    x = x + 25;
    gotoxy(x, y);
    cout << "Nick Sore: " << score;
    x = x + 25;
    gotoxy(x, y);
    cout << "Nick Health: " << nickHealth;
    gotoxy(x, y + 1);
    for(int count = 0; count < 5; count++)
    {
        cout << "   ";
    }
    gotoxy(x, y + 1);
    for(int count = 0; count < nickHealth; count++)
    {
        cout << box << box << box;
    }
    x = x + 25;
    gotoxy(x, y);
    cout << "Elites Health: " << elitesHealth;
    gotoxy(x, y + 1);
    for(int count = 0; count < 5; count++)
    {
        cout << "   ";
    }
    gotoxy(x, y + 1);
    for(int count = 0; count < elitesHealth; count++)
    {
        cout << box << box << box;
    }
    x = x + 25;
    gotoxy(x, y);
    cout << "DeathClaws Health: " << deathClawsHealth;
    gotoxy(x, y + 1);
    for(int count = 0; count < 5; count++)
    {
        cout << "   ";
    }
    gotoxy(x, y + 1);
    for(int count = 0; count < deathClawsHealth ; count++)
    {
        cout << box << box << box;
    }
    x = x + 25;
    gotoxy(x, y);
    cout << "Letho Health: " << lethoHealth;
    gotoxy(x, y + 1);
    for(int count = 0; count < 5; count++)
    {
        cout << "   ";
    }
    gotoxy(x, y + 1);
    for(int count = 0; count < lethoHealth; count++)
    {
        cout << box << box << box;
    }
    x = x + 25;
    gotoxy(x, y);
    cout << "DarkGannon Health: " << darkGannonHealth;
    gotoxy(x, y + 1);
    for(int count = 0; count < 5; count++)
    {
        cout << "   ";
    }
    gotoxy(x, y + 1);
    for(int count = 0; count < darkGannonHealth; count++)
    {
        cout << box << box << box;
    }
}

void printNick()
{
    SetConsoleTextAttribute(color,1);
    for (int rows = 0; rows < 6; rows++)
    {
        gotoxy(nickX, nickY + rows);
        for (int columns = 0; columns < 11; columns++)
        {
            cout << nick[rows][columns];
        }
    }
}
void eraseNick()
{
    for (int rows = 0; rows < 6; rows++)
    {
        gotoxy(nickX, nickY + rows);
        for (int columns = 0; columns < 11; columns++)
        {
            cout << " ";
        }
    }
}
void printElites()
{
    SetConsoleTextAttribute(color,5);
    for (int rows = 0; rows < 7; rows++)
    {
        gotoxy(elitesX, elitesY + rows);
        for (int columns = 0; columns < 26; columns++)
        {
            cout << elites01[rows][columns];
        }
    }
}
void printReverseElites()
{
    SetConsoleTextAttribute(color,5);
    for (int rows = 0; rows < 7; rows++)
    {
        gotoxy(elitesX, elitesY + rows);
        for (int columns = 25; columns >= 0; columns--)
        {
            cout << elites02[rows][columns];
        }
    }
}
void eraseElites()
{
    for (int rows = 0; rows < 7; rows++)
    {
        gotoxy(elitesX, elitesY + rows);
        for (int columns = 0; columns < 26; columns++)
        {
            cout << " ";
        }
    }
}
void printLetho()
{
    SetConsoleTextAttribute(color,8);
    for (int rows = 0; rows < 4; rows++)
    {
        gotoxy(lethoX, lethoY + rows);
        for (int columns = 0; columns < 5; columns++)
        {
            cout << letho[rows][columns];
        }
    }
}
void eraseLetho()
{
     for (int rows = 0; rows < 4; rows++)
    {
        gotoxy(lethoX, lethoY + rows);
        for (int columns = 0; columns < 5; columns++)
        {
            cout << " ";
        }
    }
}
void printDeathClaws()
{
    SetConsoleTextAttribute(color,4);
    for (int rows = 0; rows < 7; rows++)
    {
        gotoxy(deathClawsX, deathClawsY + rows);
        for (int columns = 0; columns < 10; columns++)
        {
            cout << deathClaws[rows][columns];
        }
    }
}
void eraseDeathClaws()
{
    for (int rows = 0; rows < 7; rows++)
    {
        gotoxy(deathClawsX, deathClawsY + rows);
        for (int columns = 0; columns < 10; columns++)
        {
            cout << " ";
        }
    }
}
void printDarkGannon()
{
    SetConsoleTextAttribute(color,2);
    for (int rows = 0; rows < 4; rows++)
    {
        gotoxy(darkGannonX, darkGannonY + rows);
        for (int columns = 0; columns < 6; columns++)
        {
            cout << darkGannon[rows][columns];
        }
    }
}

void eraseDarkGannon()
{
    for (int rows = 0; rows < 4; rows++)
    {
        gotoxy(darkGannonX, darkGannonY + rows);
        for (int columns = 0; columns < 6; columns++)
        {
            cout << " ";
        }
    }
}
void moveNickLeft()
{
    char next = getCharAtxy(nickX - 2, nickY + 3);
    if (next == ' ')
    {
        eraseNick();
        nickX = nickX - 2;
        printNick();
    }
}
void moveNickRight()
{
    char next = getCharAtxy(nickX + 13, nickY + 3);
    if (next == ' ')
    {
        eraseNick();
        nickX = nickX + 2;
        printNick();
    }
}
void moveNickUp()
{
    char next = getCharAtxy(nickX + 5, nickY - 2);
    if (next == '`')
    {
        eraseNick();
        nickY = nickY - 10;
        printNick();
    }
}
void moveNickDown()
{
    char next01 = getCharAtxy(nickX , nickY + 6);
    char next02 = getCharAtxy(nickX , nickY + 8);
    if (next01 == '_' && next02 == '`')
    {
        eraseNick();
        nickY = nickY + 10;
        printNick();
    }
}
void gravityForNick()
{
    bool flag = true;
    char space01;
    char space02;
    while (flag)
    {
        space01 = getCharAtxy(nickX, nickY + 6);
        space02 = getCharAtxy(nickX + 10, nickY + 6);
        if (space01 == ' ' && space02 == ' ')
        {
            eraseNick();
            nickY = nickY + 1;
            printNick();
        }
        else
        {
            flag = false;
        }
    }
}
void moveElites()
{

    if (elitesDirection == "Right")
    {
        char next = getCharAtxy(elitesX + 26, elitesY);
        if ((next == ' ') && elitesX < 83)
        {
            eraseElites();
            elitesX++;
            printReverseElites();
        }
        else
        {
            elitesDirection = "Left";
        }
    }
    if (elitesDirection == "Left")
    {
        char next01 = getCharAtxy(elitesX, elitesY);
        char next02 = getCharAtxy(elitesX - 2, elitesY);
        if (next01 == ' ')
        {
            eraseElites();
            elitesX--;
            printElites();
        }
        if ((next01 == '|') || (next02 != ' '))
        {
            elitesDirection = "Right";
        }
    }
}
void moveLetho()
{

    if (lethoDirection == "Down")
    {
        char next = getCharAtxy(lethoX, lethoY + 4);
        if ((next == '_') && lethoY < 40)
        {
            eraseLetho();
            lethoY = lethoY + 10;
            printLetho();
        }
        else
        {
            lethoDirection = "Up";
        }
    }
    if (lethoDirection == "Up")
    {
        char next = getCharAtxy(lethoX, lethoY - 4);
        if (next == '`')
        {
            eraseLetho();
            lethoY = lethoY - 10;
            printLetho();
        }
        else
        {
            lethoDirection = "Down";
        }
    }
}
void moveDarkGannon()
{
    if (darkGannonDirection == "Right")
    {
        char next = getCharAtxy(darkGannonX + 6, darkGannonY);
        if ((next == ' ') && darkGannonX < 120)
        {
            eraseDarkGannon();
            darkGannonX++;
            printDarkGannon();
        }
        else
        {
            darkGannonDirection = "Down";
        }
    }
    else if (darkGannonDirection == "Down")
    {
        char next = getCharAtxy(darkGannonX, darkGannonY + 4);
        if ((next == '_') && darkGannonY < 40)
        {
            eraseDarkGannon();
            darkGannonY = darkGannonY + 10;
            printDarkGannon();
        }
        else
        {
            darkGannonDirection = "Left";
        }
    }
    else if (darkGannonDirection == "Left")
    {
        char next = getCharAtxy(darkGannonX, darkGannonY);
        if ((next == ' ') && darkGannonX > 87)
        {
            eraseDarkGannon();
            darkGannonX--;
            printDarkGannon();
        }
        else
        {
            darkGannonDirection = "Up";
        }
    }
    else if (darkGannonDirection == "Up")
    {
        char next = getCharAtxy(darkGannonX, darkGannonY - 4);
        if ((next == '`') && darkGannonY < 19)
        {
            eraseDarkGannon();
            darkGannonY = darkGannonY - 9;
            printDarkGannon();
        }
        else
        {
            darkGannonDirection = "Right";
        }
    }
}
void moveDeathClaws()
{
    int chaseX = nickX - deathClawsX;
    int chaseY = nickY - deathClawsY;
    // for horizontal movement
    // To move right
    if(chaseX > 0)
    {
        eraseDeathClaws();
        deathClawsX = deathClawsX + 1;
        printDeathClaws();
    }
    // To move left
    if(chaseX < 0)
    {
        eraseDeathClaws();
        deathClawsX = deathClawsX - 1;
        printDeathClaws();
    }
    // For vertical movement
    // to move down
    if(chaseY >= 9)
    {
        eraseDeathClaws();
        deathClawsY = deathClawsY + 10;
        printDeathClaws();
    }
    // To move up
    if(chaseY < 0)
    {
        char next = getCharAtxy(deathClawsX + 4 , deathClawsY - 1);
        chaseY = (-1) * chaseY;
        if(chaseY >= 9 && next == '`')
        {
            eraseDeathClaws();
            deathClawsY = deathClawsY - 10;
            printDeathClaws();
        }
    }

}
void moveDeathClawsDown()
{
    bool flag = true;
    char space01;
    char space02;
    while (flag)
    {
        space01 = getCharAtxy(deathClawsX, deathClawsY + 7);
        space02 = getCharAtxy(deathClawsX + 9, deathClawsY + 7);
        if (space01 == ' ' && space02 == ' ')
        {
            eraseDeathClaws();
            deathClawsY = deathClawsY + 1;
            printDeathClaws();
        }
        else
        {
            flag = false;
        }
    }
}

void printSnowRight(int x, int y)
{
    SetConsoleTextAttribute(color,7);
    gotoxy(x, y);
    cout << "s";
}
void eraseSnow(int x, int y)
{
    gotoxy(x, y);
    cout << " ";
}
void generateSnowRight()
{
    rightSnowX[rightSnowCount] = nickX + 11;
    rightSnowY[rightSnowCount] = nickY + 2;
    gotoxy(nickX + 11, nickY + 2);
    cout << "s";
    rightSnowCount++;
    score++;
}
void removeRightSnowFromArray(int index)
{
    for (int x = index; x < rightSnowCount - 1; x++)
    {
        rightSnowX[x] = rightSnowX[x + 1];
        rightSnowY[x] = rightSnowY[x + 1];
    }
    rightSnowCount--;
}
void moveSnowRight()
```

```
{
    for (int x = 0; x < rightSnowCount; x++)
    {
        char next = getCharAtxy(rightSnowX[x] + 1, rightSnowY[x]);
        if (next != ' ')
        {
            eraseSnow(rightSnowX[x], rightSnowY[x]);
            removeRightSnowFromArray(x);
        }
        else
        {
            eraseSnow(rightSnowX[x], rightSnowY[x]);
            rightSnowX[x] = rightSnowX[x] + 1;
            printSnowRight(rightSnowX[x], rightSnowY[x]);
        }

    }
}
void generateSnowLeft()
{
    leftSnowX[leftSnowCount] = nickX - 1;
    leftSnowY[leftSnowCount] = nickY + 2;
    gotoxy(nickX - 1, nickY + 2);
    cout << "S";
    leftSnowCount++;
    score++;
}
void printSnowLeft(int x, int y)
{
    SetConsoleTextAttribute(color,7);
    gotoxy(x, y);
    cout << "S";
}
void removeLeftSnowFromArray(int index)
{
    for (int x = index; x < leftSnowCount - 1; x++)
    {
        leftSnowX[x] = leftSnowX[x + 1];
        leftSnowY[x] = leftSnowY[x + 1];
    }
    leftSnowCount--;
}
void moveSnowLeft()
{
    for (int x = 0; x < leftSnowCount; x++)
    {
        char next = getCharAtxy(leftSnowX[x] - 1, leftSnowY[x]);
        if (next != ' ')
        {
            eraseSnow(leftSnowX[x], leftSnowY[x]);
            removeLeftSnowFromArray(x);
        }
        else
        {
            eraseSnow(leftSnowX[x], leftSnowY[x]);
            leftSnowX[x] = leftSnowX[x] - 1;
            printSnowLeft(leftSnowX[x], leftSnowY[x]);
        }

    }
}

void printFire(int fireX, int fireY)
{
    SetConsoleTextAttribute(color,4);
    gotoxy(fireX, fireY);
    cout << "F";
}
void eraseFire(int fireX, int fireY)
{
    gotoxy(fireX, fireY);
    cout << " ";
}
void generateElitesFire()
{
    if(elitesDirection == "Left")
    {
        elitesFireX[elitesFireCount] = elitesX + 4;
        elitesFireY[elitesFireCount] = elitesY + 4;
        printFire(elitesFireX[elitesFireCount], elitesFireY[elitesFireCount]);
        elitesFireCount++;
    }
    if(elitesDirection == "Right")
    {
        elitesFireX[elitesFireCount] = elitesX + 22;
        elitesFireY[elitesFireCount] = elitesY + 4;
        printFire(elitesFireX[elitesFireCount], elitesFireY[elitesFireCount]);
        elitesFireCount++;
    }
}
void moveElitesFire()
{
    char next01 , next02 ,next03;
    for(int idx = 0; idx < elitesFireCount; idx++)
    {
        next01 = getCharAtxy(elitesFireX[idx] , elitesFireY[idx] + 1);
        next02 = getCharAtxy(elitesFireX[idx] , elitesFireY[idx] + 2);
        next03 = getCharAtxy(elitesFireX[idx] , elitesFireY[idx] + 3);
        if((next01 == '_' || next01 == '.') && next02 == '*' && next03 == '`')
        {
            eraseFire(elitesFireX[idx] , elitesFireY[idx]);
            elitesFireY[idx] = elitesFireY[idx] + 4;
            printFire(elitesFireX[idx] , elitesFireY[idx]);
        }
        if(next01 == ' ')
        {
            eraseFire(elitesFireX[idx] , elitesFireY[idx]);
            elitesFireY[idx] = elitesFireY[idx] + 1;
            printFire(elitesFireX[idx] ,  elitesFireY[idx]);
        }
        else
        {
            eraseFire(elitesFireX[idx] , elitesFireY[idx]);
        }
    }
}
void generateLethoFire()
{
    lethoFireX[lethoFireCount] = lethoX - 1 ;
    lethoFireY[lethoFireCount]  = lethoY + 1;
    printFire(lethoFireX[lethoFireCount] , lethoFireY[lethoFireCount] );
    lethoFireCount++;
}
void moveLethoFire()
{
    char next;
    for(int idx = 0; idx < lethoFireCount; idx++)
    {
        next = getCharAtxy(lethoFireX[idx] - 1 , lethoFireY[idx]);
        next = getCharAtxy(lethoFireX[idx] - 1 , lethoFireY[idx]);
        if(next == ' ')
        {
            eraseFire(lethoFireX[idx],lethoFireY[idx]);
            lethoFireX[idx] = lethoFireX[idx] - 1;
            printFire(lethoFireX[idx], lethoFireY[idx]);
        }
        else
        {
            eraseFire(lethoFireX[idx],lethoFireY[idx]);
        }
    }
}
void generateDarkGannonFire()
{
    darkGannonFireX[darkGannonFireCount] = darkGannonX - 1;
    darkGannonFireY[darkGannonFireCount] = darkGannonY + 2;
    printFire(darkGannonFireX[darkGannonFireCount], darkGannonFireY[darkGannonFireCount]);
    darkGannonFireCount++;
}
void moveDarkGannonFire()
{
    char next;
    for(int idx = 0; idx < darkGannonFireCount; idx++)
    {
        next = getCharAtxy(darkGannonFireX[idx] - 1 , darkGannonFireY[idx]);
        if(next == ' ')
        {
            eraseFire(darkGannonFireX[idx],darkGannonFireY[idx]);
            darkGannonFireX[idx] = darkGannonFireX[idx] - 1;
            printFire(darkGannonFireX[idx], darkGannonFireY[idx]);
        }
        else
        {
            eraseFire(darkGannonFireX[idx],darkGannonFireY[idx]);
        }
    }
}

void collisionOfNickAndEnemies(int playerX , int playerY , int playerWidth , int playerHeight , int enemyX , int enemyY , int enemyWidth , int enemyHeight)
{
    int height = playerHeight - enemyHeight;
    int width = playerWidth - enemyWidth;
    if((playerX + playerWidth == enemyX && playerY + height == enemyY) || (enemyX + enemyWidth == playerX && playerY + height == enemyY) || (enemyY + enemyHeight == playerY && playerX + width == enemyX) || (playerY + playerHeight == enemyY  && playerX + width == enemyX))
    {
        eraseNick();
        nickX = 3;
        nickY = 33;
        printNick();
        if(nickHealth>=0)
        nickHealth--;
    }
}
void collisionOfSnowWithElites()
{
    char snow01 , snow02;
    snow01 = getCharAtxy(elitesX - 1 , elitesY + 3);
    snow02 = getCharAtxy(elitesX + 26  , elitesY + 3);
    if(snow01 == 'S' || snow01 == 's' || snow02 == 'S' || snow02 == 's')
    {
        cout << " ";
        if(elitesHealth>=0)
        elitesHealth--;
    }
}
void collisionOfSnowWithDeathClaws()
{
    char snow01 , snow02;
    snow01 = getCharAtxy(deathClawsX - 1 , deathClawsY + 3);
    snow02 = getCharAtxy(deathClawsX + 10  , deathClawsY + 3);
    if(snow01 == 'S' || snow01 == 's' || snow02 == 'S' || snow02 == 's')
    {
        cout << " ";
        if(deathClawsHealth>=0)
        deathClawsHealth--;
    }
}
void collisionOfSnowWithLetho()
{
    char snow01 , snow02;
    snow01 = getCharAtxy(lethoX - 1, lethoY );
    snow02 = getCharAtxy(lethoX + 5 , lethoY);
    if(snow01 == 'S' || snow01 == 's' || snow02 == 'S' || snow02 == 's')
    {
        cout << " ";
        if(lethoHealth>=0)
        lethoHealth--;
    }
}
void collisionOfSnowWithDarkGannon()
```

```
{
    char snow01 , snow02;
    snow01 = getCharAtxy(darkGannonX - 1, darkGannonY );
    snow02 = getCharAtxy(darkGannonX + 6 , darkGannonY);
    if(snow01 == 'S' || snow01 == 's' || snow02 == 'S' || snow02 == 's')
    {
        cout << " ";
        if(darkGannonHealth>=0)
        darkGannonHealth--;
    }
}
void collisionOfFireAndNick()
{
    char fire;
    fire = getCharAtxy(nickX + 11, nickY + 3);
    if(fire == 'F' )
    {
        if(nickHealth>=0)
        nickHealth--;
    }
    fire = getCharAtxy(nickX + 11, nickY + 4);
    if(fire == 'F' )
    {
        if(nickHealth>=0)
        nickHealth--;
    }
    fire = getCharAtxy(nickX + 4, nickY);
    if(fire == 'F' )
    {
        if(nickHealth>=0)
        nickHealth--;
    }
    fire = getCharAtxy(nickX + 5, nickY);
    if(fire == 'F' )
    {
        if(nickHealth>=0)
        nickHealth--;
    }
    fire = getCharAtxy(nickX + 6, nickY);
    if(fire == 'F' )
    {
        if(nickHealth>=0)
        nickHealth--;
    }
    cout << " ";

}
void collisionOfFoodOrMoney()
{
    char next01 , next02;
    next01 = getCharAtxy(nickX - 1 , nickY + 5);
    next02 = getCharAtxy(nickX + 11  , nickY + 5);
    if(next01 == '@' || next02 == '@')
    {
        cout << " ";
        score = score + 5;
    }
    if(next01 == '$' || next02 == '$')
    {
        cout << " ";
        score = score + 10;
    }
}

void storeScoreAndHealth()
{
    fstream file;
    file.open("scoreAndHealth.txt", ios :: out);
    int idx = 0;
    file << score << ",";
    file << nickHealth << ",";
    file << elitesHealth << ",";
    file << deathClawsHealth << ",";
    file << lethoHealth << ",";
    file << darkGannonHealth << ",";
    file.close();
}
void loadScoreAndHealth()
{
    string line;
    fstream file;
    file.open("scoreAndHealth.txt", ios :: in);
    while(getline(file,line))
    {
        score = stoi(getfield(line,1));
        nickHealth = stoi(getfield(line,2));
        elitesHealth = stoi(getfield(line,3));
        deathClawsHealth = stoi(getfield(line,4));
        lethoHealth = stoi(getfield(line,5));
        darkGannonHealth = stoi(getfield(line,6));
    }
    file.close();
}
string getfield(string line, int field)
{
    int commaCount = 1;
    string item = "";
    for(int idx = 0; idx < line.length(); idx++)
    {
        if(line[idx] == ',')
        {
            commaCount++;
        }
        else if(commaCount == field)
        {
            item = item + line[idx];
        }
    }
    return item;
}
```

```

char getCharAtxy(short int x, short int y)
{
    CHAR_INFO ci;
    COORD xy = {0, 0};
    SMALL_RECT rect = {x, y, x, y};
    COORD coordBufSize;
    coordBufSize.X = 1;
    coordBufSize.Y = 1;
    return ReadConsoleOutput(GetStdHandle(STD_OUTPUT_HANDLE), &ci, coordBufSize, xy, &rect) ? ci.Char.AsciiChar : ' ';
}
void gotoxy(int x, int y)
{
    COORD coordinates;
    coordinates.X = x;
    coordinates.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coordinates);
}
void ShowConsoleCursor(bool showFlag)
{
    HANDLE out = GetStdHandle(STD_OUTPUT_HANDLE);

    CONSOLE_CURSOR_INFO cursorInfo;

    GetConsoleCursorInfo(out, &cursorInfo);
    cursorInfo.bVisible = showFlag; // set the cursor visibility
    SetConsoleCursorInfo(out, &cursorInfo);
}
```