Final submission:

Final project:

**Console based Snake Game**

Group: **F2M**

Fatima Ahmed 22F-3159

Mufaza Urfan 23F-0726

Meerab Awais 23F-0515

**About the game:**

We made this game using c++ and is a one player game. The objective is to generate food at random positions and the game continues until the snake is moving, eating the food and not colliding with the boundary or its tail.

**Natural language:**

1. Ask the user to enter their name:
2. Display menu
3. Ask for the user’s choice
4. If the user enters 1 the game initiates
5. Board is displayed and snake is moved according to the keys user press.
6. If the user enters 2 display the rules
7. If the user enters 3 display highscores
8. If the user enters 4 exit the game

**Code:**

#include <iostream>

#include <string>

#include <fstream>

#include <windows.h>

#include <conio.h>

#include <ctime>

using namespace std;

// Constants for special keys

#define KEY\_UP 72

#define KEY\_DOWN 80

#define KEY\_LEFT 75

#define KEY\_RIGHT 77

#define KEY\_W 119

#define KEY\_A 97

#define KEY\_S 115

#define KEY\_D 100

string playerName;

const int width = 70;

const int height = 30;

const int maxHighScores = 10;

bool gameover;

int h, t;

int fruitX, fruitY, score;

int tailX[100], tailY[100];

int prevposX, prevposY, prevpos2X, prevpos2Y;

int length;

pair<string, int> highScores[maxHighScores];

HANDLE console = GetStdHandle(STD\_OUTPUT\_HANDLE);

enum direction { UP, DOWN, LEFT, RIGHT };

direction dir;

void initializeGame() {

gameover = false;

h = width / 2;

t = height / 2;

fruitX = rand() % width;

fruitY = rand() % height;

score = 0;

dir = RIGHT; // Initial direction

}

void setCursorPosition(int x, int y) {

COORD coord;

coord.X = x;

coord.Y = y;

SetConsoleCursorPosition(console, coord);

}

void drawCharacter(char character, int x, int y) {

setCursorPosition(x, y);

cout << character;

}

void drawGameBoard() {

system("cls");

for (int i = 0; i < width; i++)

cout << "#";

cout << endl;

for (int i = 0; i < height; i++) {

for (int j = 0; j < width; j++) {

if (j == 0 || j == width - 1)

cout << "#";

else if (i == t && j == h)

cout << "S";

else if (i == fruitY && j == fruitX)

cout << "X";

else {

bool tail = false;

for (int k = 0; k < length; k++) {

if (tailX[k] == j && tailY[k] == i) {

cout << "s";

tail = true;

}

}

if (!tail)

cout << " ";

}

}

cout << endl;

}

for (int i = 0; i < width; i++)

cout << "#";

cout << endl;

cout << "score: " << score << endl;

}

void input() {

if (\_kbhit()) {

int ch = \_getch();

switch (ch) {

case KEY\_UP:

case KEY\_W:

if (dir != DOWN)

dir = UP;

break;

case KEY\_DOWN:

case KEY\_S:

if (dir != UP)

dir = DOWN;

break;

case KEY\_RIGHT:

case KEY\_D:

if (dir != LEFT)

dir = RIGHT;

break;

case KEY\_LEFT:

case KEY\_A:

if (dir != RIGHT)

dir = LEFT;

break;

}

}

}

void moveSnake() {

prevposX = tailX[0];

prevposY = tailY[0];

tailX[0] = h;

tailY[0] = t;

for (int i = 1; i < length; i++) {

prevpos2X = tailX[i];

prevpos2Y = tailY[i];

tailX[i] = prevposX;

tailY[i] = prevposY;

prevposX = prevpos2X;

prevposY = prevpos2Y;

}

// Automatic movement based on direction

switch (dir) {

case UP:

t--;

break;

case DOWN:

t++;

break;

case LEFT:

h--;

break;

case RIGHT:

h++;

break;

}

// Check for collisions with borders

if (h <= 0 || h >= width - 1 || t <= 0 || t >= height)

gameover = true;

for (int i = 0; i < length; i++) {

if (tailX[i] == h && tailY[i] == t)

gameover = true;

}

}

void collisionWithFood() {

if (h == fruitX && t == fruitY) {

score += 5;

fruitX = rand() % width;

fruitY = rand() % height;

length++;

}

}

void saveScore(const string& playerName, int playerScore) {

ofstream outfile("highscores.txt", ios::app);

if (outfile.is\_open()) {

outfile << playerName << " " << playerScore << endl;

outfile.close();

}

}

void displayHighScores() {

ifstream infile("highscores.txt");

if (infile.is\_open()) {

cout << "High Scores:" << endl;

string playerName;

int playerScore;

while (infile >> playerName >> playerScore) {

cout << playerName << " - " << playerScore << endl;

}

infile.close();

}

}

void printRules() {

cout << "Rules:" << endl;

cout << "1. Use arrow keys or WASD to control the snake." << endl;

cout << "2. The snake grows longer when it eats the 'X' fruit." << endl;

cout << "3. Avoid colliding with the walls and your own tail." << endl;

cout << "4. The game ends when you collide, and your score is displayed." << endl;

cout << "5. You can save your score with your name in the highscores." << endl;

}

int main() {

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_BLUE);

int Choice;

cout << "Enter your name: ";

getline(cin, playerName);

do {

cout << "Snake Game" << endl;

cout << "=====================" << endl;

cout << "1. Start Game" << endl;

cout << "2. Read Rules" << endl;

cout << "3. See Highscores" << endl;

cout << "4. Exit" << endl;

cout << "=====================" << endl;

cout << "Enter your choice: ";

cin >> Choice;

switch (Choice) {

case 1:

while (true) {

initializeGame();

while (!gameover) {

drawGameBoard();

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_RED);

input();

moveSnake();

collisionWithFood();

Sleep(100);

}

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_RED);

system("cls");

cout << "Game Over!" << endl;

saveScore(playerName, score);

cout << "Score: " << score << endl;

cout << "Player: " << playerName << endl;

displayHighScores();

cout << "Powered by M2F" << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 15); // Set console color back to white

// Ask user if they want to play again

cout << "Do you want to play again? (1 for Yes, 0 for No): ";

int playAgain;

cin >> playAgain;

if (playAgain != 1)

break;

}

break;

case 2:

printRules();

break;

case 3:

displayHighScores();

break;

case 4:

cout << "Exiting the game. Goodbye!" << endl;

cout << "Powered by M2F" << endl;

break;

default:

cout << "Invalid choice. Please try again." << endl;

}

system("pause");

system("cls");

} while (Choice != 4);

return 0;

}



