INTERNATIONAL BURCH UNIVERSITY

FACULTY OF ENGINEERING AND IT

DEPARTMENT OF INFORMATION TECHNOLOGY



**CONSOLE GAME: SNAKE**

PROGRAMMING I TERM PROJECT REPORT

Ferhat Avdic

SARAJEVO

January, 2015

**Problem/Topic**

A typical snake game like we had in old nokia 3310.



**Analysis**

The program includes a two-dimensional array used to show the game instances. In the main function switch is used inside an infinite loop so the user can enter a specific value to perform a desired action.

Various functions are used to perform those actions:

void instructions(); 🡪 Show game instructions

void viewScores(); 🡪 View scoreboard

void deleteScores(); 🡪 Delete all scores

int playGame(); 🡪 The function that makes the game animation possible

For the main menu interface:

void mainMenu(); 🡪 Show main menu of the game

Functions used inside the “playGame” function:

void createField();

🡪 Populates the two-dimensional array with characters so it shows the game borders

void inGameScore(int score, int dead);

🡪 Prints the game score on top of the screen,

🡪 Prints GAME OVER if you lose

void printField();

🡪 Prints the two-dimensional array

void saveScore(int score);

🡪 Asks for user’s nickname and saves both the nickname and the score to a .txt file

\_kbhit()

🡪 A function from conio.h library. Used to determine if a key has been pressed or not. If a key has been pressed then it returns a non-zero value, otherwise it returns zero.

🡪 In this program it is used to animate the snake when key is pressed *and* when it is not pressed

**Design**

1. Define a file pointer and a two-dimensional array as global variables
2. Define constant values for the rows and columns of the two-dimensional array
3. Make the main menu and instructions interfaces
4. Create a function to play the game:

* define all needed variables and arrays
* generate random seed
* generate first worm random position
* make a loop to animate the screen – the loop is the game
* On each loop iteration do following:
  + get snake movement conditions (inside the loop because we need the snake animated even when no key is pressed)
  + take snake body position
  + take worm position
  + take snake head position
  + clear screen and print out the score and the two-dimensional array
  + wipe snake body trace
  + score conditions
  + death conditions
* When snake dies and game is over save the score

1. Make functions to view and delete scores

**Code**

#include<stdio.h>

#include<conio.h>

#include<Windows.h>

#include<stdlib.h>

#include<time.h>

#include<string.h>

#define rowSize 17

#define columnSize 33

char field[rowSize][columnSize];

FILE \*fp;

void mainMenu();

int playGame();

void instructions();

void createField();

void inGameScore(int score, int dead);

void printField();

void saveScore(int score);

void viewScores();

void deleteScores();

int main()

{

char menuChoice = '2';

while (1)

{

mainMenu();

if (menuChoice < 49 || menuChoice > 53)

printf("\n\n\t\t\t\t ERROR: Wrong input. Try again.\n");

menuChoice = getch();

switch (menuChoice)

{

case '1': playGame(); break;

case '2': instructions(); break;

case '3': viewScores(); break;

case '4': deleteScores(); break;

case '5': return 0; break;

default: system("cls"); break;

}

}

}

void mainMenu()

{

system("cls");

printf("\n\n\t\t\t\t ------------------------------------- ");

printf("\n\t\t\t\t| GAME MENU |");

printf("\n\t\t\t\t|-------------------------------------|");

printf("\n\t\t\t\t| |");

printf("\n\t\t\t\t| [1] Play Game |");

printf("\n\t\t\t\t| [2] Instructions |");

printf("\n\t\t\t\t| [3] View Scores |");

printf("\n\t\t\t\t| [4] Delete Scores |");

printf("\n\t\t\t\t| [5] Exit |");

printf("\n\t\t\t\t| |");

printf("\n\t\t\t\t ------------------------------------- ");

}

void instructions()

{

system("cls");

printf("\n\n\t\t\t\t ------------------------------------- ");

printf("\n\t\t\t\t| INSTRUCTIONS | ");

printf("\n\t\t\t\t|-------------------------------------|");

printf("\n\t\t\t\t| |");

printf("\n\t\t\t\t| -> Eat worms to increase score |");

printf("\n\t\t\t\t| -> Don't eat yellow snow |");

printf("\n\t\t\t\t| -> Use W,A,S,D keys to play |");

printf("\n\t\t\t\t| -> Snake dies if you hit the wall |");

printf("\n\t\t\t\t| or if it bites itself |");

printf("\n\t\t\t\t| |");

printf("\n\t\t\t\t ------------------------------------- \n\n\t\t\t\t");

system("Pause");

}

void inGameScore(int score, int dead)

{

printf("\n");

printf("\n\t\t SCORE: %d\t\t %s", score, dead == 1 ? "GAME OVER" : " ");

printf("\n\t\t -----------");

printf("\n\n\t\t");

}

void saveScore(int score)

{

fflush(stdin);

char name[10];

printf("Enter nickname: ");

fgets(name, 10, stdin);

strtok(name, "\n");

fp = fopen("C:/Users/Ferhat/Documents/Visual Studio 2013/Projects/Special/Snake/scores.txt", "a");

fprintf(fp, "%s: %d\n", name, score);

fclose(fp);

}

void viewScores()

{

system("cls");

int c;

fp = fopen("C:/Users/Ferhat/Documents/Visual Studio 2013/Projects/Special/Snake/scores.txt", "r");

// Algorithm to read a character from the txt file, and print it right after reading

if (fp) {

while ((c = getc(fp)) != EOF)

putchar(c);

fclose(fp);

}

system("Pause");

system("cls");

}

void deleteScores()

{

system("cls");

fp = fopen("C:/Users/Ferhat/Documents/Visual Studio 2013/Projects/Special/Snake/scores.txt", "w");

fclose(fp);

printf("Scores successfully erased.\n");

system("Pause");

system("cls");

}

void createField()

{

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

{

for (int j = 0; j < columnSize; j++)

{

if (i == 0 && j == 0) // top left corner

field[i][j] = 254;// 201;

else if (i == 0 && j == columnSize - 1) // top right corner

field[i][j] = 254; //187;

else if (i == rowSize - 1 && j == 0) // bottom left corner

field[i][j] = 254;// 200;

else if (i == rowSize - 1 && j == columnSize - 1) // bottom right corner

field[i][j] = 254; // 188;

else if (i == 0 || i == rowSize - 1) // top and bottom row

field[i][j] = 254; // 205;

else if (j == 0 || j == columnSize - 1) // sides

field[i][j] = 254; // 186;

else

field[i][j] = ' ';

}

}

}

void printField()

{

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

{

for (int j = 0; j < columnSize; j++)

{

printf(" %c", field[i][j]);

}

printf("\n\t\t");

}

}

int playGame()

{

system("cls");

int score = 0;

int snakeX[500];

int snakeY[500];

snakeX[0] = columnSize / 2;

snakeY[0] = rowSize / 2 + 1;

int snakeBody = 2;

int dead = 0;

int move = 0;

int wormX;

int wormY;

int i; // just a loop control variable initialized

// variable to store control keys

char prevControl;

char control = 'w';

// generate random seed

time\_t t;

srand((unsigned)time(&t));

createField();

// generate first worm

wormY = 1 + (rand() % (rowSize - 2));

wormX = 1 + (rand() % (columnSize - 2));

while (dead == 0)

{

// snake body follows snake head x,y

for (i = snakeBody; i > 0; i--)

{

snakeY[i] = snakeY[i - 1];

snakeX[i] = snakeX[i - 1];

}

Sleep(100);

if (\_kbhit())

{

prevControl = control;

control = \_getch();

if (control == 'w' || control == 'W') snakeY[0]--;

if (control == 's' || control == 'S') snakeY[0]++;

if (control == 'a' || control == 'A') snakeX[0]--;

if (control == 'd' || control == 'D') snakeX[0]++;

if (control == 27) return 0;

// Movement restrictions

if (control == 'w' && prevControl == 's' || control == 'W' && prevControl == 'S') control = prevControl;

if (control == 's' && prevControl == 'w' || control == 'S' && prevControl == 'W') control = prevControl;

if (control == 'a' && prevControl == 'd' || control == 'A' && prevControl == 'D') control = prevControl;

if (control == 'd' && prevControl == 'a' || control == 'D' && prevControl == 'A') control = prevControl;

// Disabling other keys

if (control != 'w' && control != 'W' &&

control != 'a' && control != 'A' &&

control != 's' && control != 'S' &&

control != 'd' && control != 'D' &&

control != 27) control = prevControl;

}

else

{

if (control == 'w' || control == 'W') snakeY[0]--;

if (control == 's' || control == 'S') snakeY[0]++;

if (control == 'a' || control == 'A') snakeX[0]--;

if (control == 'd' || control == 'D') snakeX[0]++;

if (control == 27) return 0;

}

// create snake body

for (i = 1; i < snakeBody; i++)

field[snakeY[i]][snakeX[i]] = '\*';

// create worm

field[wormY][wormX] = '~';

// create snake head

field[snakeY[0]][snakeX[0]] = 248;

system("cls");

inGameScore(score, dead);

printField();

// wipe body trace

field[snakeY[snakeBody - 1]][snakeX[snakeBody - 1]] = ' ';

// increase score and snake when snake eats worms

if (snakeY[0] == wormY && snakeX[0] == wormX)

{

snakeBody++;

score++;

// generate worm

wormY = 1 + (rand() % (rowSize - 2));

wormX = 1 + (rand() % (columnSize - 2));

}

// death conditions

// hit the wall

if (snakeY[0] == 0) dead = 1;

else if (snakeY[0] == rowSize - 1) dead = 1;

if (snakeX[0] == 0) dead = 1;

else if (snakeX[0] == columnSize - 1) dead = 1;

// bite itself

for (i = 1; i < snakeBody; i++)

if (snakeY[0] == snakeY[i + 1] && snakeX[0] == snakeX[i + 1]) dead = 1;

move++;

}

system("cls");

inGameScore(score, dead);

printField();

saveScore(score);

printf("\n\t\t");

system("Pause");

return 0;

}