// 1o

// Characters defined for trapping the keys for playing the game

#include <conio.h>

#include <iostream.h>

#include <stdlib.h>

#include <stdio.h>

#include <dos.h>

int Up= 72;

int Down = 80;

int Left = 75;

int Right= 77;

unsigned Enter= '\013';

int Escape= 27;

#define maxLength 15

int minLength = 5;

int tdelay = 10000;

int curLength = 10;

int objects = 0;

int scoreRow = 23, scoreCol = 60;

class Nibbles

{

int snake[maxLength][2];

int treasure[10][2];

int obstacle[10][2];

int score;

char dirn;

int objectsDone;

public:

int gameEnd;

void init()

{

int r;

for(r = 0; r < 10; r++)

{ treasure[r][0] = treasure[r][1] = -1;

  obstacle[r][0] = obstacle[r][1] = -1;

}

score = 0;

objectsDone = 0;

gameEnd = 0; // 0 indicates the game has not ended. 1 says Ended

dirn='R';

}

void show()

{

int r, c;

gotoxy(snake[0][0], snake[0][1]);

cout << "@";

for(r = 1; r < curLength; r++)

{ gotoxy(snake[r][0], snake[r][1]);

 cout << "X";

}

}

void showPanel(int level)

{

// displays an outer wall between 5,2 and 75,22. Treasures and obstacles within that space

if(level == 1)

{ objects = 5; }

else if(level == 2)

{ objects = 7; }

else if(level == 2)

{ objects = 10; }

clrscr();

int r, c;

int col = 11;

for(r = 0; r < curLength; r++)

{ snake[r][0] = col--;

  snake[r][1] = 12;

}

for(r = 0; r < objects; r++)

   { int rowPos, colPos;

rowPos = randBetween(3,20);

colPos = randBetween(12, 74);

treasure[r][0] = colPos;

treasure[r][1] = rowPos;

rowPos = randBetween(3,20);

colPos = randBetween(12, 74);

obstacle[r][0] = colPos;

obstacle[r][1] = rowPos;

   }

// Display outer boundary

for(r = 2; r <= 22; r++)

{ gotoxy(5,r); cout << "#";

 gotoxy(75,r); cout << "#"; }

for(c = 5; c <= 75; c++)

{ gotoxy(c,2); cout << "#";

 gotoxy(c,22); cout << "#"; }

for(r = 0; r < objects; r++)

{ gotoxy(treasure[r][0], treasure[r][1]); cout << "$";

  gotoxy(obstacle[r][0],obstacle[r][1]); cout << "#";

}

}

void move(int gk)

{

 if(gk == Up) dirn = 'U';

if(gk == Down) dirn = 'D';

if(gk == Left) dirn = 'L';

if(gk == Right) dirn = 'R';

gotoxy(snake[curLength-1][0], snake[curLength-1][1]); cout << " ";

for(int r = curLength-1; r >= 1; r--)

   { snake[r][0] = snake[r-1][0];

     snake[r][1] = snake[r-1][1]; }

if(dirn == 'U') snake[0][1]--;

if(dirn == 'D') snake[0][1]++;

if(dirn == 'L') snake[0][0]--;

if(dirn == 'R') snake[0][0]++;

// Check whether to award or punish

for(r = 0; r < 10; r++)

{ if(snake[0][0] == treasure[r][0] && snake[0][1] == treasure[r][1])

{score += 100; showScore(); objectsDone++;}

  if(snake[0][0] == obstacle[r][0] && snake[0][1] == obstacle[r][1])

gameEnd = 1;

}

// check if the snake curls up and touches itself

for(r = 1; r < curLength; r++)

{ if(snake[0][0] == snake[r][0] && snake[0][1] == snake[r][1])

gameEnd = 1;

}

// check if the snake touches the wall

if(snake[0][0] == 5 || snake[0][0] == 75)

gameEnd = 1;

if(snake[0][1] == 2 || snake[0][1] == 22)

gameEnd = 1;

}

int play()

{

while(1)

{

int gk;

if(kbhit())

gk = getch();

move(gk);

show();

delay(250);

if(gk == Escape) break;

if(gameEnd == 1) break;

if(objectsDone == objects) break;

}

}

void showScore()

{

 gotoxy(scoreCol, scoreRow); cout << "Score : " << score;

}

};

void dispStart();

void main()

{

clrscr();

randomize(); // Start a new Random seed

int level = 1;

dispStart();

Nibbles n;

while(1)

{

unsigned c = ' ';

n.init();

n.showPanel(level);

c = n.play();

if(c == Escape)

   break;

level++;

if(level == 4)

   break;

if(n.gameEnd == 1)

   break;

}

clrscr();

if(n.gameEnd == 1) cout << "You CRASHED..."; else cout << "Game ENDED";

n.showScore();

}

void dispStart()

{

gotoxy(20,10);

cout << "Nibbles ver 1.0";

gotoxy(15,12);

cout << "Created by Himanshu\n\n";

cout << "This game is to be played at three levels\n";

cout << "Each level is with an increased level of difficulty\n";

cout << "Press Up, Down, Left, Right keys to move the Snake\n";

cout << "If the snake touches a treasure chest ($), you get 100 points\n";

cout << "If the snake touches the outer boundary, or an obstacle (#), the game ends\n";

cout << "Press Escape any time to End the Game\n";

getch();

clrscr();

}

int randBetween(int lo, int hi)

{

int r;

do

{

r = random(hi);

} while (r < lo);

return r;

}