

Snake Game

+

PPS PROJECT

By Dhatrish Bali
RA2111001010003



BASICS OF THE GAME

The snake is represented with a **0**(zero) symbol.

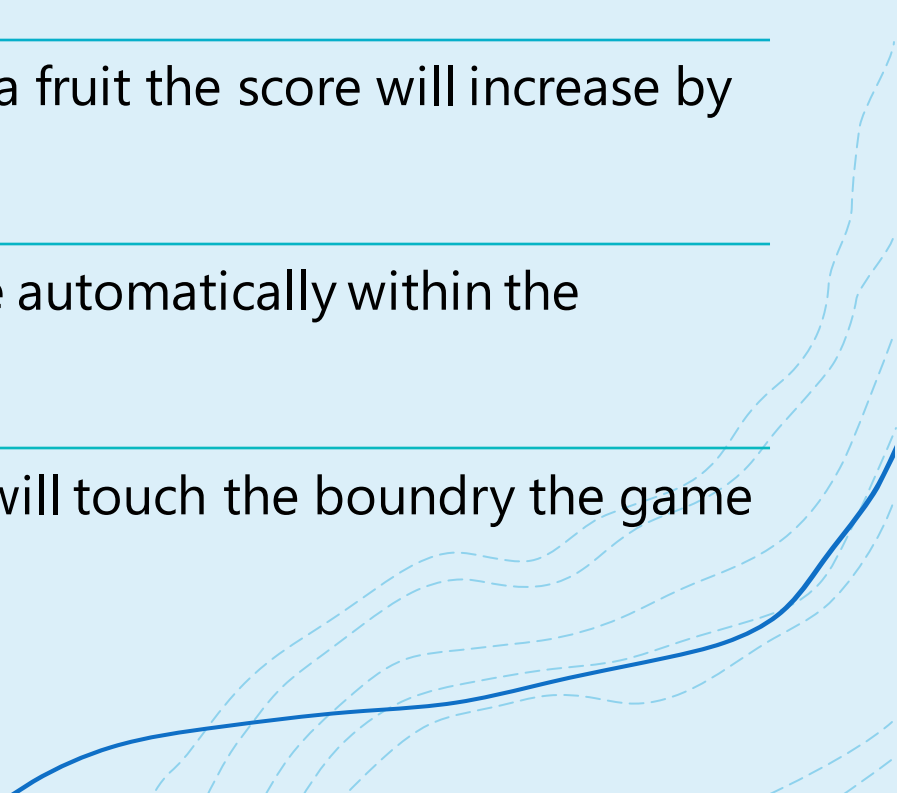
The fruit is represented with an *****(asterisk) symbol.

The snake can move in any direction according to the user with the help of the keyboard (**W**, **A**, **S**, **D** keys).

When the snake eats a fruit the score will increase by 10 points.

The fruit will generate automatically within the boundaries.

Whenever the snake will touch the boundry the game is over.



STEPS

There will be four user-defined functions.

Build a boundary within which the game will be played.

The fruits are generated randomly.

Then increase the score whenever the snake eats a fruit.

USER DEFINED FUNCTIONS

1

Draw(): This function creates the boundary in which the game will be played.

2


Setup(): This function will set the position of the fruit within the boundary.

3

Input(): This function will take the input from the keyboard.

4

Logic(): This function will set the movement of the snake.



```
C #include <conio.h> Untitled-2 ●  
1  #include <conio.h>  
2  #include <stdio.h>  
3  #include <stdlib.h>  
4  #include <unistd.h>  
5  
6  int i, j, height = 20, width = 20;  
7  int gameover, score;  
8  int x, y, fruitx, fruity, flag;  
9
```

The header file and
Variable used in this
programe


```
#####
#           #
#   *       #
#           #
#           #
#           #
#           #
#           #
#           #
#   0       #
#           #
#           #
#           #
#           #
#           #
#           #
#####
score = 0
press X to quit the game
```

```
30 // Function to draw the boundaries
31 void draw()
32 {
33     system("cls");
34     for (i = 0; i < height; i++) {
35         for (j = 0; j < width; j++) {
36             if (i == 0 || i == width - 1
37                 || j == 0
38                 || j == height - 1) {
39                 printf("#");
40             }
41             else {
42                 if (i == x && j == y)
43                     printf("0");
44                 else if (i == fruitx
45                         && j == fruity)
46                     printf("*");
47                 else
48                     printf(" ");
49             }
50         }
51         printf("\n");
52     }
53 }
```

raw(): This function is responsible to build the boundary within which the game will be played

setup(): This function is used to write the code to generate the fruit within the boundary using **rand()** function.

+ Using **rand()%20** because the size of the boundary is **length = 20** and **width = 20** so the fruit will generate within the boundary.

```
12 void setup()
13 {
14     gameover = 0;
15
16     // Stores height and width
17     x = height / 2;
18     y = width / 2;
19 label1:
20     fruitx = rand() % 20;
21     if (fruitx == 0)
22         goto label1;
23 label2:
24     fruity = rand() % 20;
25     if (fruity == 0)
26         goto label2;
27     score = 0;
28 }
```

Input(): In this function, the programmer writes the code to take the input from the keyboard (W, A, S, D, X keys).

```
62 void input()
63 {
64     if (kbhit()) {
65         switch (getch()) {
66             case 'a':
67                 flag = 1;
68                 break;
69             case 's':
70                 flag = 2;
71                 break;
72             case 'd':
73                 flag = 3;
74                 break;
75             case 'w':
76                 flag = 4;
77                 break;
78             case 'x':
79                 gameover = 1;
80                 break;
81         }
82     }
83 }
84
```

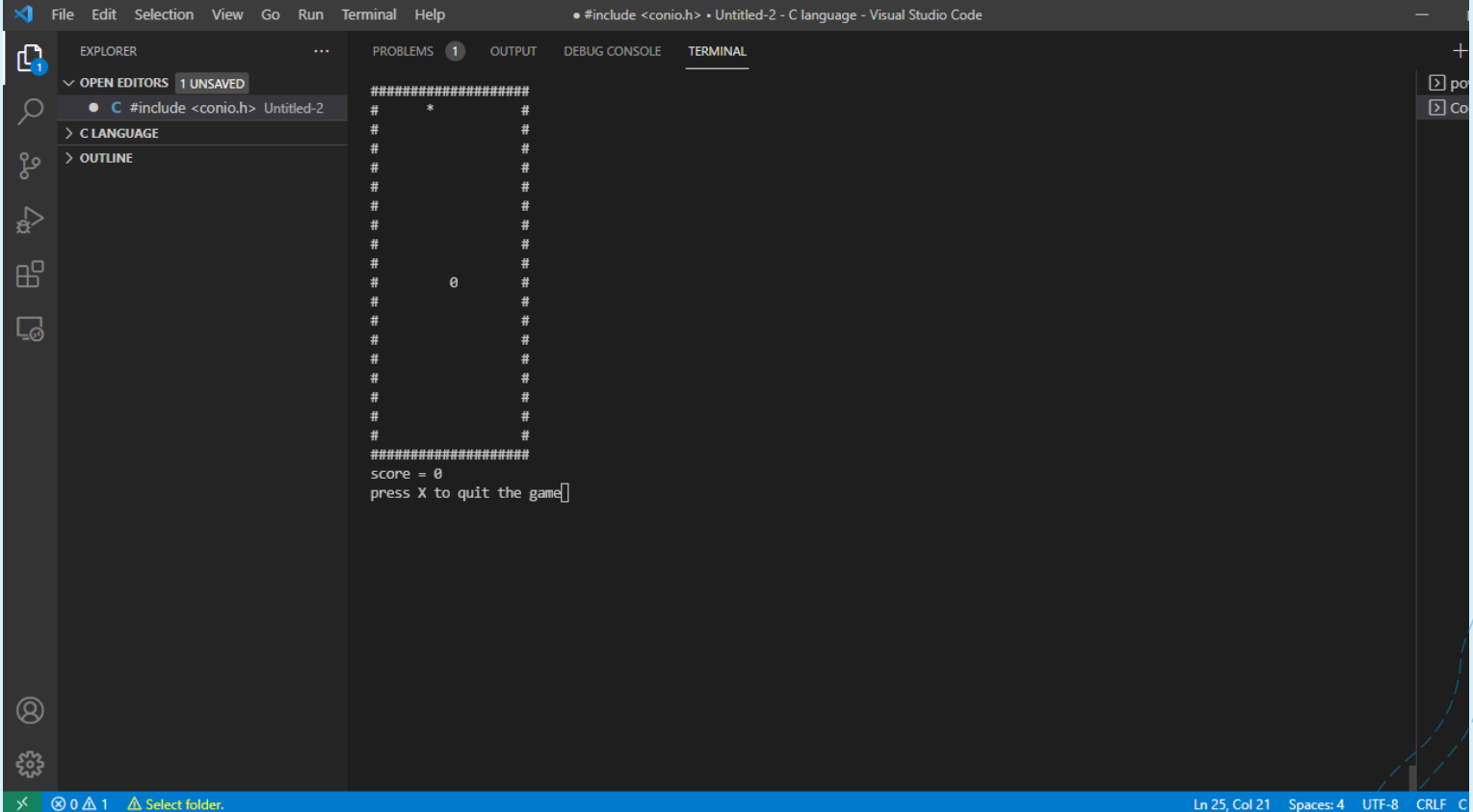

logic(): Here, write all the logic for this program like for the movement of the snake, for increasing the score, when the snake will touch the boundary the game will be over, to exit the game and the random generation of the fruit once the snake will eat the fruit.

```
85 // Function for the logic behind
86 // each movement
87 void logic()
88 {
89     sleep(0.01);
90     switch (flag) {
91     case 1:
92         y--;
93         break;
94     case 2:
95         x++;
96         break;
97     case 3:
98         y++;
99         break;
100    case 4:
101        x--;
102        break;
103    default:
104        break;
105    }
106
107    // If the game is over
108    if (x < 0 || x > height
109        || y < 0 || y > width)
110        gameover = 1;
111
112    // If snake reaches the fruit
113    // then update the score
114    if (x == fruitx && y == fruity) {
115    label3:
116        fruitx = rand() % 20;
117        if (fruitx == 0)
118            goto label3;
119
120    // After eating the above fruit
121    // generate new fruit
122    label4:
123        fruity = rand() % 20;
124        if (fruity == 0)
125            goto label4;
126        score += 10;
127    }
128 }
129
```

sleep(): This function in C is a function that delays the program execution for the given number of seconds. In this code sleep() is used to slow down the movement of the snake so it will be easy for the user to play.

```
130 // Driver Code
131 void main()
132 {
133     int m, n;
134
135     // Generate boundary
136     setup();
137
138     // Until the game is over
139     while (!gameover) {
140
141         // Function Call
142         draw();
143         input();
144         logic();
145     }
146 }
```

THE FINAL PRODUCT



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project with an 'Untitled-2' file. The editor displays a C program that prints a diamond shape of asterisks and a score of 0. The terminal on the right shows the output of the program, which matches the code in the editor.

```
#####  
#      *      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#      #      #  
#####  
score = 0  
press X to quit the game
```