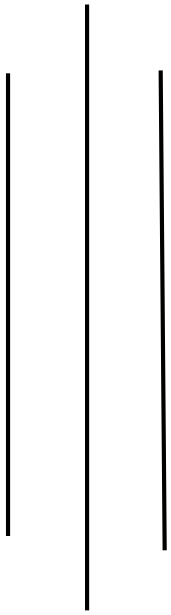
National College of Engineering

(Affiliated to Tribhuvan University)

Talchhikhel, Lalitpur



Submitted By:
Name: Saksham Maharjan
Roll no: 077BCT029
Group: A2

Submitted To:
Department of Electronics
& Computer Engineering

Contents

INTRODUCTION

A header file 'graphics.h' which is a C programming language library is commonly used for creating simple computer graphics applications. It provides a set of functions for drawing basic shapes, colors, and images on the screen. The library also allows to set up a graphics environment, to capture mouse and keyboard events for user to interact with the graphics window and simple animations.

OBJECTIVE

The main objectives of this project are as follows:

- To learn and implement different functions of graphics.
- To interface the application of graphics to the real world.
- To familiarize with graphics and its logical coding.

FUNCTIONS USED:

The functions used in the program are as follows:

1. GRAPHICS.H: GRAPHICS.H is a header file in C that is used to include graphics functions in a C program. It is used for drawing various shapes and as well as animate objects. It is also used to color the objects drawn.

   SYNTAX: *#include<graphics.h>*

2. TIME.H: TIME.H is a C standard library header that provides functions and types for working with time and date-related operations. It allows C programs to access and manipulate time values, handle date and time information, and measure time intervals.

   SYNTAX: *#include<time.h>*

3. WINDOWS.H: WINDOWS.H is a header file in the Microsoft Windows which is primarily used in C and C++ programming languages to access various Windows-specific functionalities and create Windows applications.

   SYNTAX: *#include<windows.h>*

4. initgraph() function: It initializes the graphics system by loading a graphics driver from disk then puts the system into graphics modem. Initgraph() also resets all graphics settings (color, palette, current position, viewport, etc.) to their defaults, and then resets graph result to 0.

   SYNTAX: *void initgraph (int*graphdriver, int *graphmode, char *pathtodriver);*

5. initwindow() function: The function initializes the graphics system by opening a graphics window of the specified size. It has three parameters; width, height and title. The title parameter will be printed at the top of the window.

   SYNTAX: *initwindow(width, height, title);*

6. srand() function: srand() is used to initialize random number generators. The srand() function sets the starting point for producing a series of pseudo-random integers.

   SYNTAX: *srand(time(NULL));*

7. setfillstyle() function: It sets the current fill pattern and fill color. It consists of two parameters.

   SYNTAX: *void setfillstyle(int pattern, int color);*

COLOR	**INT VALUES**
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	10
LIGHTCYAN	11
LIGHTRED	12
LIGHTMAGENTA	13
YELLOW	14
WHITE	15

Pattern Fill	**INT VALUES**
EMPTY_FILL	0
SOLID_FILL	1
LINE_FILL	2
LTSTLASH_FILL	3
SLASH_FILL	4
BKSLASH_FILL	5
LTBKSLASH_FILL	6
HATCH_FILL	7
XHATCH_FILL	8
INTERLEAVE_FILL	9
WIDE_DOT_FILL	10
CLOSE_DOT_FILL	11
USER_FILL	12

8. bar() function: It is used to draw a 2-dimensional, rectangular filled in bar.

   SYNTAX: *void bar(int left, int top, int right, int bottom);*

9. rand() function: The rand() function is used to generate pseudo-random numbers. Pseudo-random numbers are numbers that appear to be random but are actually generated by a deterministic algorithm.

   SYNTAX: *int rand(void);*

10. getpixel() function: getpixel() returns the color of pixel present at location (x, y).

    SYNTAX: *int getpixel(int x, int y);*

11. GetAsyncKeyState() function: This function gives information about the key. i.e. It checks whether a key is pressed or not.

    SYNTAX: *short GetAsynKeyState(int key);*

CODE	MEANING
VK_LSHIFT	LEFT-SHIFT KEY.
VK_RSHIFT	RIGHT-SHIFT KEY.
VK_LCONTROL	LEFT-CONTROL KEY.
VK_RCONTROL	RIGHT-CONTROL KEY.
VK_LMENU	LEFT-MENU KEY.
VK_RMENU	RIGHT-MENU KEY.

12. delay() function: The delay() function is used to stop the execution of the program for some period of time. It accepts a time in milliseconds to stop the execution of the program to that period of time.

    SYNTAX: *delay(unsigned int);*

13. cleardevice() function: It clears the screen in graphics mode and sets the current position to (0,0). Clearing the screen consists of filling the screen with current background color.

    SYNTAX: *void cleardevice();*

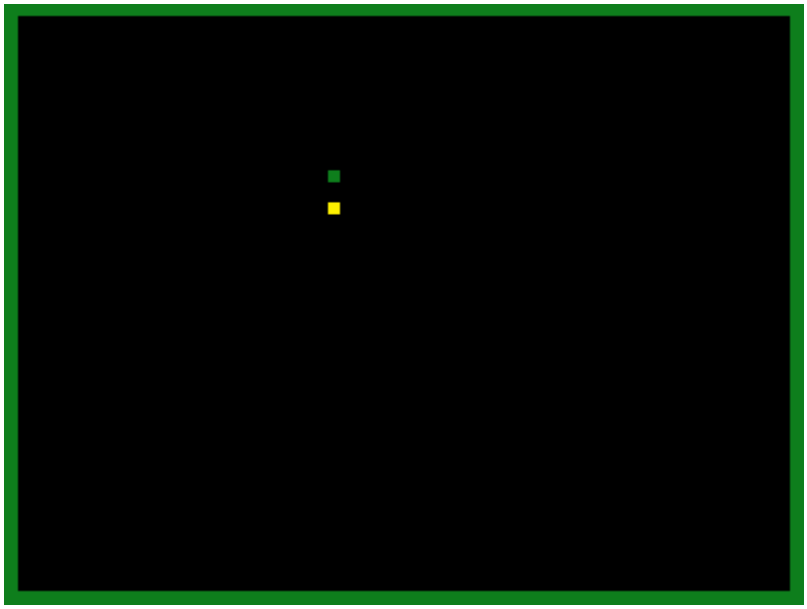
14. settextstyle() function: It is used to change the way in which the text appears. We can modify the size of text, change direction of text, and change the font of the text.

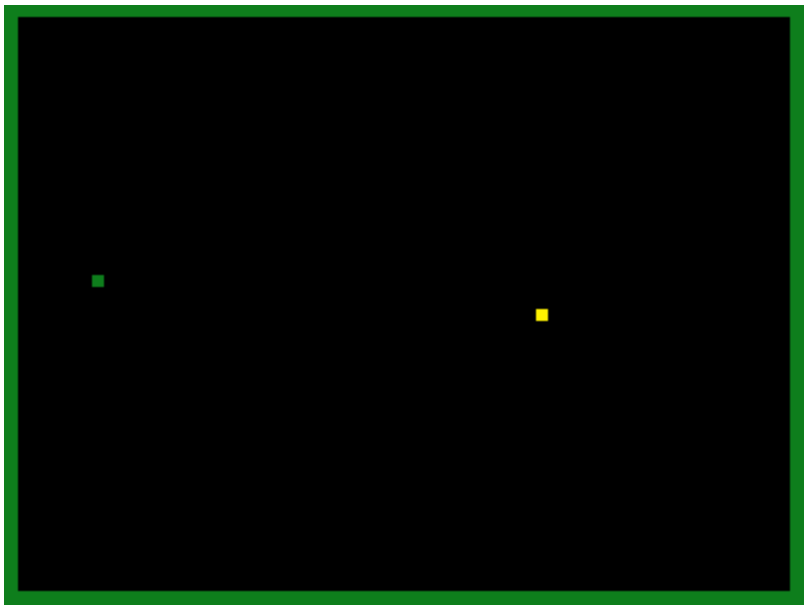
    SYNTAX: *settextstyle(int font, int direction, int charsize);*

15. outtextxy() function: It is used to display the text at specified position (x,y) on the screen.

    SYNTAX: *outtextxy(int x, int y, char *string);*

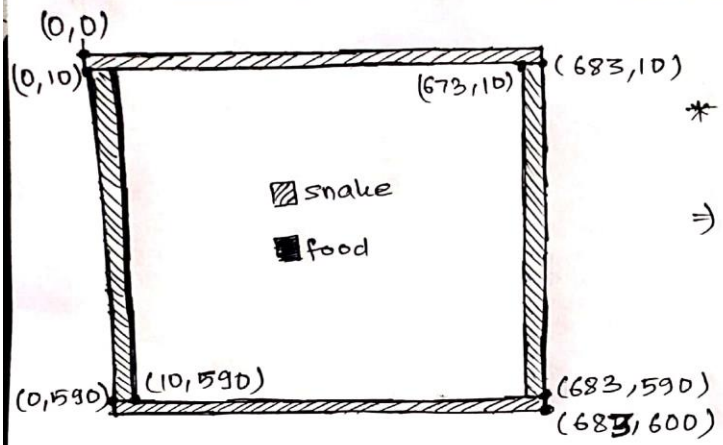
EXPECTED OUTPUT






Game Over

PROCESS



* coordinates for the boundary bar
⇒ width = 10

coordinates for the food: (rx, ry) ⇒ initialized at (200, 200)

// To make food

1. make using random function i.e rand().

// new value of food coordinates:

1. rx = 1 + rand() % 663 ← To avoid generate food in boundary
   ry = 1 + rand() % 580

2. rx = rx / 10;
   rx = rx * 10;
   [rx = 51 then rx = 51/10 = 5.1 ⇒ 5
   rx = 5 * 10 = 50]

   ry = ry / 10;
   ry = ry * 10;
   [ry = 79 then ry = 79/10 = 7.9
   ⇒ 7
   ry = 7 * 10 = 70]

// For movement of snake:

eg. if 'arrow key i.e GetAsyncKeyState(VK_RIGHT) then
= right

d = 1;

// when (d = k = 1), x = x + 10, and (dir = 1 :) → for current direction of snake

[moves the snake about 1 unit right

status of arrow key direction.

SOURCE CODE

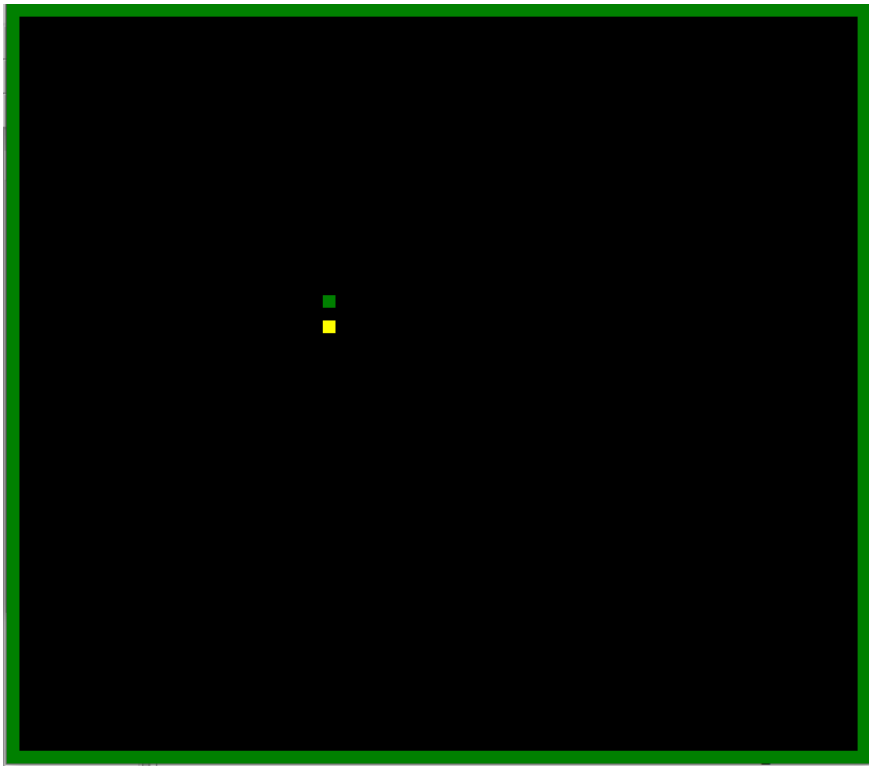
```
#include<stdio.h>
#include<time.h>
#include<windows.h>
#include<graphics.h>
#include<stdlib.h>

void gameover();
int  endfunk(int e);

int main()
{
   int gd=DETECT, gm, x=200, y=200, d=1, dir=1, rx=200, ry= 200, c=0 , fx, fy;
   initgraph(&gd,&gm,"");
   initwindow(683,600,"SNAKEMAN"); //window resolution and name
   delay(1000);
   srand(time(NULL)); //for starting point of food to be random every time
   setfillstyle(1,2);
   for(;;)
   {
      setfillstyle(1,0);//screen clear to black
      bar(0,0,683,600);
      setfillstyle(1,2);//boundary color for snake

      //boundary bars
      bar(0,0,683,10);
      bar(0,590,683,600);
      bar(0,10,10,590);
      bar(673,10,683,590);

//to make food
      if(x == rx && y == ry)
      {
         c = c + 1; //food  counter for  score
         setfillstyle(1,0); //color to erase the previous food
         bar(rx,ry,rx+10,ry+10); //previous food
         do
         {
            rx = (1 + rand() % 663);
```

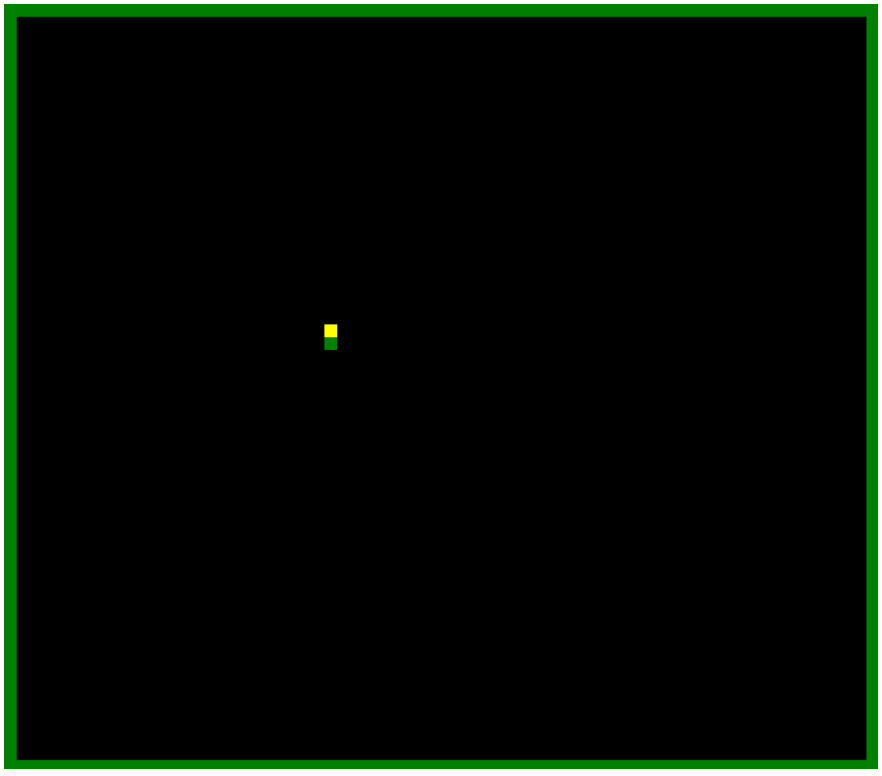
```
      ry = (1 + rand() % 580);
   }while(getpixel(rx,ry) != 0 && rx > 10 && ry > 10);
   rx = rx / 10;
   rx = rx * 10;
   ry = ry / 10;
   ry = ry * 10;
   setfillstyle(1,14); // color for when snake reach the food
}
setfillstyle(1,14); // color for when new food is displayed
bar(rx,ry,rx+10,ry+10); //new food
setfillstyle(1,2);
//arrow keys
if(GetAsyncKeyState(VK_RIGHT))
{
   d = 1;
}
else if(GetAsyncKeyState(VK_LEFT))
{
   d = 2;
}
else if(GetAsyncKeyState(VK_UP))
{
   d = 3;
}
else if(GetAsyncKeyState(VK_DOWN))
{
   d = 4;
}
else
{
   d = 0;
}
switch(d)
{
case 0: //when no arrow key is pressed
   if(dir == 1)
   {
      x = x+10;
   }
   else if(dir == 2)
   {
```

```
      x = x-10;
   }
   else if(dir == 3)
   {
      y = y - 10;
   }
   else if(dir == 4)
   {
      y = y + 10;
   }

   else
   {
      d = 0;
   }
   break;

case 1: //right key
   x = x + 10;
   dir = 1;
   break;

case 2: //left key
   x = x - 10;
   dir = 2;
   break;

case 3: //up key
   y = y - 10;
   dir = 3;
   break;

case 4: //down key
   y = y + 10;
   dir = 4;
   break;

}
bar(x,y,x+10,y+10); //next move of snake
delay(100);
if(x >= 683 || x <= 0 || y <= 0 || y >= 600)// when snake cross the boundary
```

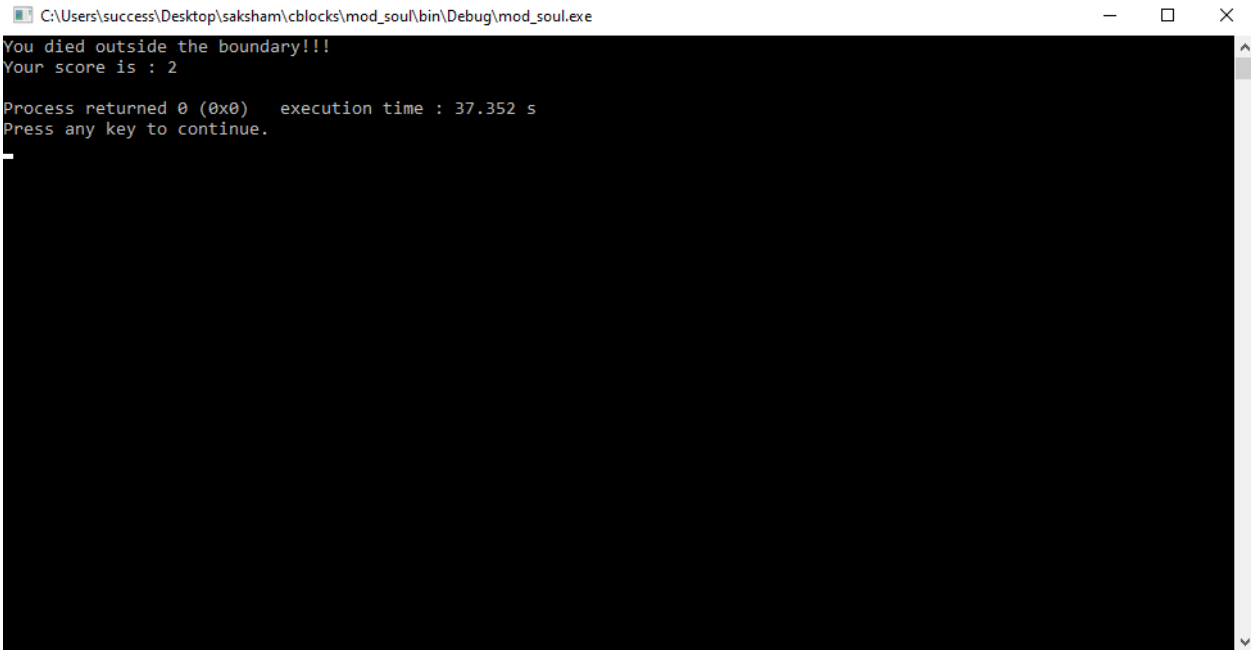
```
      {
         cleardevice();
         gameover();
         delay(2000);
         endfunk(c);
         break;
      }
   }
}

void gameover()
{
   setfillstyle(1,WHITE);
   settextstyle(3,0,5);
   outtextxy((getmaxx()/2)-130,(getmaxy()/2)-50,"Game Over");
}

int endfunk(int e)
{
   e=e-1;
   system("cls");  //to clear the console window
   printf("You died outside the boundary!!!\n");
   printf("Your score is : %d\n", e);
}
```

OUTPUT






SNAKEMAN
Game Over


C:\Users\success\Desktop\saksham\cblocks\mod_soul\bin\Debug\mod_soul.exe
You died outside the boundary!!!
Your score is : 2
Process returned 0 (0x0)   execution time : 37.352 s
Press any key to continue.

CONCLUSION

Thus, using various inbuilt graphics functions of header file graphics.h, we were able to create a snake game. We learnt about these functions, implemented them in a C program and successfully operated the program to get a graphics image as output. Many problems that arises during execution of the program were fixed by thorough debugging. In this way, we got familiarized with various graphics functions and learnt about their uses and implementation.