Report On

Drawing Eiffel Tower

Submitted in partial fulfillment of the requirements of the Course project in
Semester III of Second Year Artificial Intelligence and Data Science

by
Rohan G Mangaonkar (Roll No. 27)
Tarun Premnaryan Pathak (Roll No. 38)
Aryan Mohan Parab (Roll No. 36)

Supervisor
Prof. Sejal D'mello

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled “Eiffil Tower” is a bonafide work of "Rohan G Mangaonkar (Roll No. 27), Tarun Premnaryan Pathak (Roll No. 38), Aryan Mohan Parab (Roll No. 36) " submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester III of Second Year Artificial Intelligence and Data Science engineering.

Supervisor

Prof. Sejal D’mello

Dr. Tatwadarshi P. N.
Head of Department

Table of Contents

Pg. No

ABSTRACT

This C program uses the Turbo C graphics library to create a decorative pattern on the screen. It utilizes various graphics functions to draw lines, shapes, and decorative elements. The program is structured as follows:

Initialization:
The graphics system is initialized using the initgraph function with automatic detection of the graphics driver. The path to the graphics driver files is specified.
Left and Right Side Drawing:
The program draws intricate decorative patterns on both the left and right sides of the screen. These patterns consist of lines, triangles, rectangles, and various shapes.
Rectangles for Decoration:
Three rectangles are drawn in the middle of the screen to enhance the decorative effect.
Triangle and Vertical Line Decoration:
Triangles and vertical lines are drawn as part of the decoration. Loops are used to create a series of triangles and vertical lines.
Middle Line and Upper Most Decoration:
A vertical line is drawn in the middle of the screen.
Above the middle line, additional decorative elements are drawn using a loop, creating a visually appealing pattern.
Tangent Lines and Rectangle:
Tangent lines are drawn from the upper bases to the middle of the screen.
A rectangle is drawn at the intersection of the tangent lines.
Screen Interaction:
The program uses getch() to wait for user input to keep the graphics window open.
Upon receiving a key press, the program closes the initialized graphics driver using closegraph().
This program showcases graphics programming capabilities in Turbo C, illustrating how various graphics functions can be used to create intricate decorative patterns on the screen.

CODE

TOWER.c > ...

```c
// C program for the above approach
#include <conio.h>
#include <graphics.h>
#include <stdio.h>

// Driver Code
void main()
{
    int gd = DETECT, gm;

    // Initialize of gdriver with
    // DETECT macros
    initgraph(&gd, &gm, "C:\\turboc3\\bgi");

    // Declared Variables
    int a = 390, b = 390, c = 700;

    // Left Side
    // Lower Base
    line(300, 1000, 450, 1000);

    // Inside Decoration
    line(300, 1000, 480, 940);
    line(450, 1000, 330, 940);

    // 1st Base
    line(330, 940, 480, 940);

    // Inside Decoration
    line(330, 940, 510, 880);
    line(480, 940, 360, 880);

    // 2nd Base
    line(360, 880, 510, 880);

    // Inside Decoration
    line(360, 880, 540, 820);
    line(390, 820, 510, 880);

    // 3rd Base
    line(390, 820, 540, 820);

    // Left Tangent
    line(300, 1000, 390, 820);

    // Right Tangent
    line(450, 1000, 540, 820);

    // Joining Line
    line(390, 820, 810, 820);
```

```
51
52      // Half Circle
53      ellipse(600, 900, 15, 165, 90, 80);
54
55      // Right Side
56      // Lower Base
57      line(750, 1000, 900, 1000);
58
59      // Inside Decoration
60      line(750, 1000, 870, 940);
61      line(720, 940, 900, 1000);
62
63      // 1st Base
64      line(720, 940, 870, 940);
65
66      // Inside Decoration
67      line(720, 940, 840, 880);
68      line(870, 940, 690, 880);
69
70      // 2nd Base
71      line(690, 880, 840, 880);
72
73      // Inside Decoration
74      line(690, 880, 810, 820);
75      line(840, 880, 660, 820);
76
77      // 3rd Base
78      line(660, 820, 810, 820);
79
80      // Left Tangent
81      line(750, 1000, 660, 820);
82
83      // Right Tangent
84      line(900, 1000, 810, 820);
85
86      // Rectangles For Decoration
87      rectangle(390, 800, 810, 820);
88      rectangle(380, 780, 820, 800);
89      rectangle(390, 760, 810, 780);
90
91      // Triangle Decoration
92      while (a <= 790) {
93          line(a, 820, a + 10, 800);
94          line(a + 10, 800, a + 20, 820);
95          a = a + 20;
96      }
97
98      // Vertical Line Decoration
99      while (b <= 810) {
100         line(b, 760, b, 780);
```

```
        b = b + 20;
    }

    // Left Side
    // Upper Base
    line(410, 760, 530, 760);

    // Inside Decoration
    line(410, 760, 560, 700);
    line(530, 760, 440, 700);

    // 1st Base
    line(440, 700, 560, 700);

    // Inside Decoration
    line(440, 700, 590, 640);
    line(560, 700, 470, 640);

    // 2nd base
    line(470, 640, 590, 640);

    // Left Tangent
    line(410, 760, 470, 640);

    // Right Tangent
    line(540, 760, 590, 640);

    // Right Side
    // Upper Base
    line(670, 760, 790, 760);

    // Inside Decoration
    line(670, 760, 760, 700);
    line(790, 760, 640, 700);

    // 1st Base
    line(640, 700, 760, 700);

    // Inside Decoration
    line(640, 700, 730, 640);
    line(760, 700, 610, 640);

    // 2nd Base
    line(610, 640, 730, 640);

    // Left Tangent
    line(670, 760, 610, 640);

    // Right Tangent
    line(790, 760, 730, 640);
```

```
    line(790, 760, 730, 640);

    // Joining Line
    line(470, 640, 730, 640);

    // Rectangle For Decoration
    rectangle(460, 620, 740, 640);
    rectangle(470, 600, 730, 620);

    // Redeclaring Variable
    b = 470;

    // Vertical Line Decoration
    while (b <= 730) {
        line(b, 600, b, 620);
        b = b + 10;
    }

    // Redeclaring Variable
    a = 600;
    b = 500;

    // Middle Line
    line(600, 600, 600, 140);

    // Upper Most Decoration
    while (b >= 240) {
        if (b == c)
            break;
        else {
            line(b, a, c, a);
            line(b, a, c - 10, a - 40);
            line(b + 10, a - 40, c, a);
            a = a - 40;
            b = b + 10;
            c = c - 10;
        }
    }

    // Tangent Lines
    line(500, 600, 590, 240);
    line(700, 600, 610, 240);
    rectangle(590, 200, 610, 240);

    // Holding The Screen For A While
    getch();

    // Close the initialized gdriver
    closegraph();
}
```

WORKING

This program is written in Turbo C and uses the graphics.h library to create a decorative pattern. Here's a step-by-step explanation of the program:
In C graphics, the graphics.h functions are used to draw different shapes like circles, rectangles, etc, display text(any message) in a different format (different fonts and colors)

The first step is to make the left side base of the tower. The left side base is totally built up with a line() function.
On the left side of the base, construct a total of four lines. These lines are tiled from each other. Then join the left side of the lines with a tangent line and the right side with another tangent line. Also, do interior decoration by joining opposite sides of each base with a line. This full work has to be done by using a line() function.

The same has to be done with the right side, similar to the one done on the left side. But the difference is that it is required to tilt bases on the opposite side.
Then join two sides with a line() function.

The next step is to make a half-circle by using an ellipse() function.
Implement three rectangles by using a rectangle() function. All these rectangles will be used in decoration functions.
Among the rectangles, one will be decorated with a continuous triangle which will be implemented by the line() function. These continuous triangle decorations will be done using a while loop.
Another rectangle will be decorated with vertical lines which are separated by the same distance. These vertical lines are implemented by a line() function in another while loop.

Steps followed on the lower base again have to be done here as well. The full method is totally the same here also. But here, we have implemented three bases instead of four.

Join the two sides with a line() function.

Implement two rectangles using the rectangle() function. Between them, the upper one is to be decorated by some vertical lines placed at the same distance from each other. These lines will be implemented by the line() function in a while loop.

Make a while loop that will divide the height of the remaining tower and also create some decoration in it in a single while loop. This whole operation will be implemented by the line() function.
Join the two sides with the line function. Create a rectangle using a rectangle() function on the upper side and a straight line using a line() function.

Function Used:

rectangle(l, t, r, b): A function from graphics.h header file which draws a rectangle from left(l) to right(r) & from top(t) to bottom(b).

line(a1, b1, a2, b2): A function from graphics.h header file which draws a line from (a1, b1) point to (a2, b2) point.

ellipse(int x, int y, int start_angle, int end_angle, int x_radius, int y_radius): A function from graphics.h header file where x, y is the location of the ellipse. x_radius and y_radius decide the radius of the form x and y. start_angle is the starting point of the angle and end_angle is the ending point of the angle. The value of the angle can vary from 0 to 360 degrees.

ALGORITHM

Converting C code to algorithm

Here's the algorithm representation of the given C code:

1. Start the program.
2. Initialize the graphics driver with the DETECT macro.
3. Initialize the graphics mode using the initgraph() function, specifying the graphics driver and the graphics mode.
4. Declare the variables `a`, `b`, and `c` and assign them the values 390, 390, and 700 respectively.
5. Draw the left side of the figure:
- Draw the lower base line from (300, 1000) to (450, 1000).
- Draw the inside decoration lines from (300, 1000) to (480, 940) and from (450, 1000) to (330, 940).
- Draw the first base line from (330, 940) to (480, 940).
- Draw the inside decoration lines from (330, 940) to (510, 880) and from (480, 940) to (360, 880).
- Draw the second base line from (360, 880) to (510, 880).
- Draw the inside decoration lines from (360, 880) to (540, 820) and from (390, 820) to (510, 880).
- Draw the third base line from (390, 820) to (540, 820).
- Draw the left tangent line from (300, 1000) to (390, 820).
- Draw the right tangent line from (450, 1000) to (540, 820).
- Draw the joining line from (390, 820) to (810, 820).
- Draw the half circle at (600, 900) with a radius of 80, starting angle of 15 degrees, and ending angle of 165 degrees.
6. Draw the right side of the figure:
- Draw the lower base line from (750, 1000) to (900, 1000).
- Draw the inside decoration lines from (750, 1000) to (870, 940) and from (720, 940) to (900, 1000).
- Draw the first base line from (720, 940) to (870, 940).
- Draw the inside decoration lines from (720, 940) to (840, 880) and from (870, 940) to (690, 880).
- Draw the second base line from (690, 880) to (840, 880).

- Draw the inside decoration lines from (690, 880) to (810, 820) and from (840, 880) to (660, 820).
- Draw the third base line from (660, 820) to (810, 820).
- Draw the left tangent line from (750, 1000) to (660, 820).
- Draw the right tangent line from (900, 1000) to (810, 820).
- Draw the rectangles for decoration at (390, 800) to (810, 820), (380, 780) to (820, 800), and (390, 760) to (810, 780).
- Draw the triangle decoration with a while loop:
- Start a while loop with the condition `a <= 790`.
- Draw a line from (a, 820) to (a + 10, 800).
- Draw a line from (a + 10, 800) to (a + 20, 820).
- Increment `a` by 20.
- Draw the vertical line decoration with a while loop:
- Start a while loop with the condition `b <= 810`.
- Draw a line from (b, 760) to (b, 780).
- Increment `b` by 20.
7. Draw the left side of the upper base:
- Draw the upper base line from (410, 760) to (530, 760).
- Draw the inside decoration lines from (410, 760) to (560, 700) and from (530, 760) to (440, 700).
- Draw the first base line from (440, 700) to (560, 700).
- Draw the inside decoration lines from (440, 700) to (590, 640) and from (560, 700) to (470, 640).
- Draw the second base line from (470, 640) to (590, 640).
- Draw the left tangent line from (410, 760) to (470, 640).
- Draw the right tangent line from (540, 760) to (590, 640).
8. Draw the right side of the upper base:
- Draw the upper base line from (670, 760) to (790, 760).
- Draw the inside decoration lines from (670, 760) to (760, 700) and from (790, 760) to (640, 700).
- Draw the first base line from (640, 700) to (760, 700).
- Draw the inside decoration lines from (640, 700) to (730, 640) and from (760, 700) to (610, 640).
- Draw the second base line from (610, 640) to (730, 640).
- Draw the left tangent line from (670, 760) to (610, 640).
- Draw the right tangent line from (790, 760) to (730, 640).
- Draw the joining line from (470, 640) to (730, 640).

- Draw the rectangle for decoration at (460, 620) to (740, 640) and (470, 600) to (730, 620).
- Redefine the variable `b` as 470.
- Draw the vertical line decoration with a while loop:
- Start a while loop with the condition `b <= 730`.
- Draw a line from (b, 600) to (b, 620).
- Increment `b` by 10.
9. Redefine the variables `a`, `b`, and `c` as 600, 500, and 390 respectively.
10. Draw the middle line from (600, 600) to (600, 140).
11. Draw the upper most decoration with a while loop:
- Start a while loop with the condition `b >= 240`.
- Check if `b` is equal to `c`, if true, break the loop.
- If not, draw the lines:
- Draw a line from (b, a) to (c, a).
- Draw a line from (b, a) to (c - 10, a - 40).
- Draw a line from (b + 10, a - 40) to (c, a).
- Decrement `a` by 40.
- Increment `b` by 10.
- Decrement `c` by 10.
12. Draw the tangent lines from (500, 600) to (590, 240) and from (700, 600) to (610, 240).
13. Draw the rectangle for decoration at (590, 200) to (610, 240).
14. Wait for a keyboard input using the getch() function.
15. Close the initialized gdriver using the closegraph() function.
16. End the program.

This algorithm represents the steps performed by the given C code. Keep in mind that the algorithm does not include the specific syntax or function details, as it focuses on the logical steps of the code.

RESULT