Assignment No 10

Name → Sameer Manoj Bramhecha

Roll No → 21115

Batch → E-1

Date of Performance: 10-12-2021.

Date of Submission: 15-12-2021

Title:- Curves and Fractals

Problem Statement:-

Write C++ program to generate fractal patterns by using koch curves.

Learning Objectives:

To study curves and fractals

Learning Outcomes:

~~It~~ Students will be able to implement koch curves and fractals.

S/W and H/W Requirements:-

1.) Windows 10 OS

2.) Open source C++ programming tool like G++/ GCC

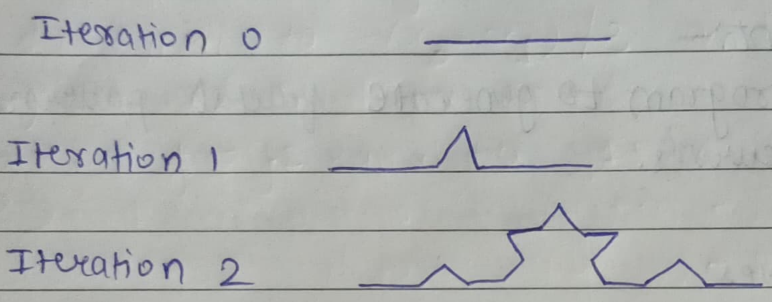
3.) QT creator.

Theory:-

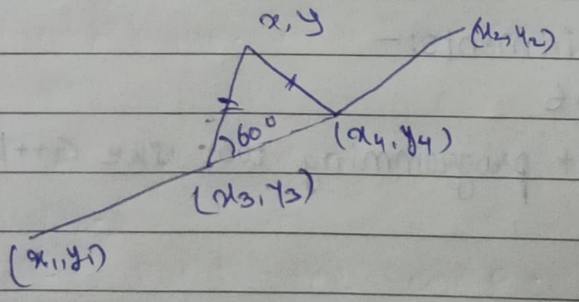
Koch Curve:→ The koch curve fractal was first introduced by Helge von koch in 1904. It was one of the first fractal objects to be described. To create a koch curve:

1. Create a line and divide it into 3 parts.
2. The second part is now rotated by 60°.
3. Add another part which goes from the end of part 2 to the beginning of part 3.
4. Repeat step 1 to step 3 with each part of the line.

We will get following koch curve as number of iteration goes on increasing.



Step 1 :→ In Iteration 0, we have a horizontal line.
Step 2 :→. In Iteration 1, line is divided into 3 equal parts. middle part of a line is rotated in 60° because if it forms a perfect equilateral triangle as shown below.



Here (x_1, y_1) and (x_2, y_2) is accepted from user. Now, we ian see line is divided into 3 equal ~~parts~~ segmental ((x_1, y_1), (x_3, y_3)), segment ((x_4, y_4), (x_2, y_2)), segment ((x_3, y_3), (x_4, y_4)) in above figure.

Coordinates of middle two points will be calculated as follows:

$x_3 = (2 * x_1 + x_2) / 3;$

$y_3 = (2 * y_1 + y_2) / 3;$

$x_4 = (x_1 + 2 * x_2) / 3;$

$y_4 = (y_1 + 2 * y_2) / 3;$

In our curve, middle segment $((x_3,y_3),(x_4,y_4))$ will not be drawn. Now, in order to find out coordinates of the top vertex (x, y) of equilateral triangle, we have rotate point (x_4, y_4) with respect to arbitary point (x_3, y_3) by angle of ~~6~~ 60 degree in anti-clockwise dirn. After performing this rotation, we will get rotated coordinates (x,y) as:

$x = x_3 + (x_4 - x_3) * \cos\theta + (y_4 - y_3) * \sin\theta$

$y = y_3 - (x_4 - x_3) * \sin\theta + (y_4 - y_3) * \cos\theta.$

Step 3: In iteration 2, you will repeat step 2 for every segment obtained in iteration 1.

In this way, you can generate koch curve for any no. of iterations.

* Snowflake Curve:

Snowflake curve is drawn using koch curve iterations. In koch curve, we just have a single line in the starting iteration and in snowflake curve, we have an equilateral triangle. Draw an equilateral triangle and repeat the steps of koch curve generation for all three segments of an equilateral triangle.

Algorithm :

① Koch Curve :→ koch() function

```
1.) Start.
2.) Pass float(it), (float) $x_1$, (float) $x_5$, (float) y, and ~~float~~ (float y5) to the function koch() as arguements.
3.) Declare int x2, x3, x4, y2, y3, y4, dx and dy.
4.) If it is equal to 0, display line btw (x1, y1) and (x5, y5).
5.) Else if it ~~>0~~ is greater than 0, define dx equal to $(x_5 - x_1)/3$ and $dy = (y_5 - y_1)/3$.
6.) Set x2 equal to $x_1 + dx$
7.) Set ~~y2~~ y2 equal to $y_1 + dx$.
8.) Set x3 equal to $(0.5(x_1 + x_5) + \sqrt{3}(y_1 - y_5)/6)$
9.) Set y3 equal to $(0.5(y_1 + y_5) + \sqrt{3}(x_5 - x_1)/6)$
10.) Set x4 equal to $(2 * dx + x_1)$
11.) Set y4 equal to $(2 * dy + y_1)$
12.) Call koch() function and pass (it-1, x1, y1, x2, y2) ~~as~~ as arguements.
13.) Call koch() function and pass (it-1, x2, y2, x3, y3) as arguements
14.) Call koch() function and pass (it-1, x3, y3, x4, y4) as arguements.
15.) Call koch() function and pass (it-1, x4, y4, x5, y5) as arguements.
16.) Else, exit the code.
17.) Stop.
```

Algorithm for Calling function :→

```
1.) Start.
```

2.) Accept no. of iterations from user.

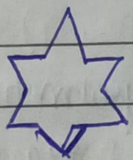
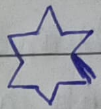
3.) Call koch function: koch (it, 150, 20, 20, 280);

4.) Call koch function: koch (it, 280, 280, 150, 20);

5.) Call koch function: koch (it, 20, 280, 280, 280);

6.) Stop.

Test Cases:→.

Test cases No.	Description	Input.	Expected Output	Actual Output	Result.
1.	Case. 1.	it = 1.			Pass
2.	Case 2.	it = ~~2~~'-1'	Code Exited	Code Exited	Pass

Conclusion: Hence, we have ~~tene~~ learnt koch curve and fractals using it.