Discrete Mathematics (ITP30003)

Programming Assignment 2 Report

Team 2

21200439 Seungjin Yang

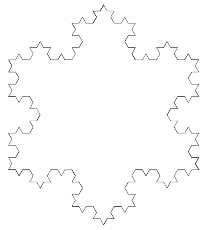
21400738 Seungyun Ji

21500830 Minho Kim

21700581 Jihyun Lee

21700097 Minju Kim

I. Fractal 1



(Pic. 1) Representative Result Image of the Program

i) Description on the Recursion That Solves the Problem

First, draw three lines to form a triangle and call recursive function for each of the lines.
The recursive function behaves as the following:

1) Basis Step: if *depth* is equal to 1, return.

2) Recursive Step: if *depth* is greater than 1, execute the following.

- Given that start-point and end-point of a line, find three points: *A, B, C*.
- A and B are determined by one third point and two third point in the line, respectively.
- x-position of *C* is determined by (x-coordinate of *A* + cos (angle of the line – PI / 3) * (length of the line / 3)).
- y-position of *C* is determined by (y-coordinate of *A* + cos (angle of the line – PI / 3) * (length of the line / 3)).
- Draw the lines that connect the following pairs of pointers.
    - (start-point, *A*), (*A*, *C*), (*C*, *B*), (*B*, end-point)
- Decrease *depth* by 1 and call this function recursively for the four lines that are newly constructed.

ii) Description on the Parameters

1) *x1*: determines the x-coordinate of the start-point of the line in consideration

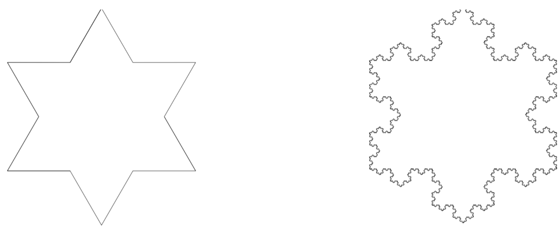
2) *y1*: determines the y-coordinate of the start-point of the line in consideration

3) *x2*: determines the x-coordinate of the end-point of the line in consideration

4) *y2*: determines the y-coordinate of the end-point of the line in consideration

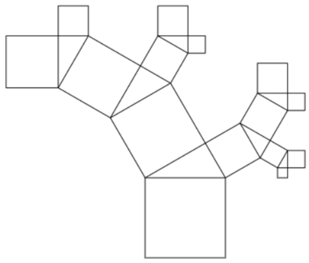
5) *depth*: determines how many times the recursive function is called

iii) Images Generated with Different Parameters

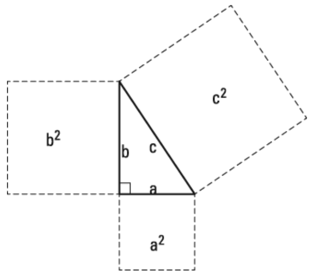


(Pic. 2) Result Images with Different Depths

II. Fractal 2



(Pic. 1) Representative Result Image of the Program



(pic.2) Reference Diagram

i) Description on the Recursion That Solves the Problem

1) Basis Step: if *depth* is equal to 1, execute the same instructions as the recursive steps, except for the recursive calling part.

2) Recursive Step: if *depth* is greater than 1, execute the following.

- Rotate the plane according to the given *angle*.

- Given that a reference point, draw a base square. Reference point represents the left-down vertex of the base square just drawn.

- Draw two rotated squares, *A* and *B*, upon the base square, so that the three squares form a reference diagram (see pic. 2), which demonstrates Pythagorean theorem.

- Decrease *depth* by 1 and call this function recursively for the left-down vertex (reference point) and *size* and *angle* of each of the squares *A* and *B*.

ii) Description on the Parameters

1) *x*: determines the x-coordinate of the left-down vertex of the base square

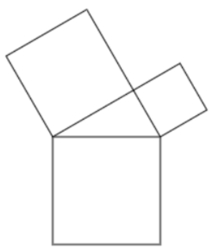
2) *y*: determines the y-coordinate of the left-down vertex of the base square

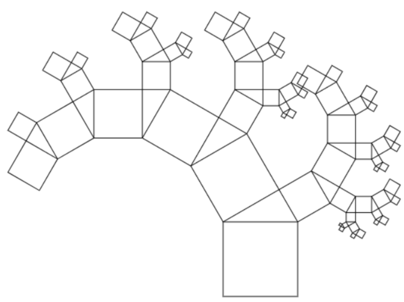
3) *angle*: determines the angle of rotation of plane

4) *size*: determines the length of the side of the base square

5) *depth*: determines how many times the recursive function is called

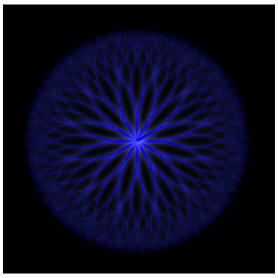
iii) Images Generated with Different Parameters





(Pic. 2) Result Images with Different Depths

III. Fractal Designed by Our Team



(Pic. 1) Result Image of the Program

i) Description on the Recursion That Creates the Fractal

From an origin point, call recursive function for the twelve times with the different parameter values for *angle*: {-150, -120, -90, -60, -30, 0, 30, 60, 90, 120, 150, 180}.
The function behaves as the following:

1) Basis Step: if *depth* is equal to 1, return.

2) Recursive Step: if *depth* is greater than 1, execute the following.

- Given that *start-point* of a line and *angle*, find the end-point of a line, *A.*
- x-coordinate of *A* is determined by (x-coordinate of *start-point* + cos(*angle* * *depth* * 8.0)
- y-coordinate of *A* is determined by (y-coordinate of *start-point* + sin(*angle* * *depth* * 8.0)
- Draw a line that connects the start-point and *A*.
- Decrease *depth* by 1 and multiply *branchWidth* by 0.6.
- Call this function recursively for the line that are newly constructed. Call it for the two times, with the different parameter values for *angle* (*angle* + 15 and *angle* – 15).

ii) Description on the Parameters

1) *x1*: determines the x-coordinate of the start-point of the line in consideration

2) *y1*: determines the y-coordinate of the start-point of the line in consideration

3) *x2*: determines the x-coordinate of the end-point of the line in consideration

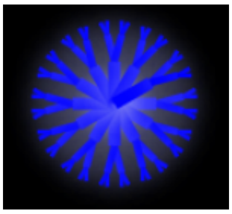
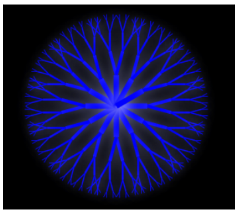
4) *y2*: determines the y-coordinate of the end-point of the line in consideration

5) *angle*: determines the angle of direction of a tree to be drawn

6) *depth*: determines how many times the recursive function is called

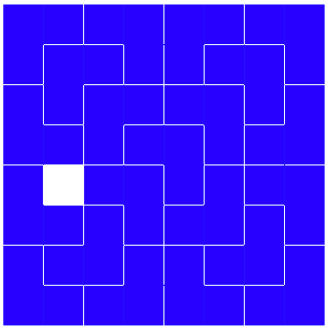
7) *branchWidth:* determines the width of the lines

iii) Images Generated with Different Parameters

(Pic. 2) Result Images with Different Depths

IV. Triomino-Tiling Problem



(Pic. 1) Representative Result Image of the Program

i) Description on the Recursion That Solves the Problem

1) Basis Step: if *n* is equal to 1, return.

2) Recursive Step: if *n* is greater than 1, execute the following.

- Consider dividing the checkerboard in half in horizontal and vertical directions to make four sub-checkerboards.
- Determine which of the four sub-checkerboards contains the missing square. Suppose that we call that sub-checkerboard, *A*.
- Tile the squares from the corner of the center of the checkerboard, each from three sub-checkerboards except for *A*.
- Now, all of four sub-checkerboards have one missing square each.
- Call this function recursively for the four sub-checkerboards with the size *n/2* x *n/2*.

ii) Description on the Parameters

1) *start_x*: determines the starting column index of the checkerboard in consideration

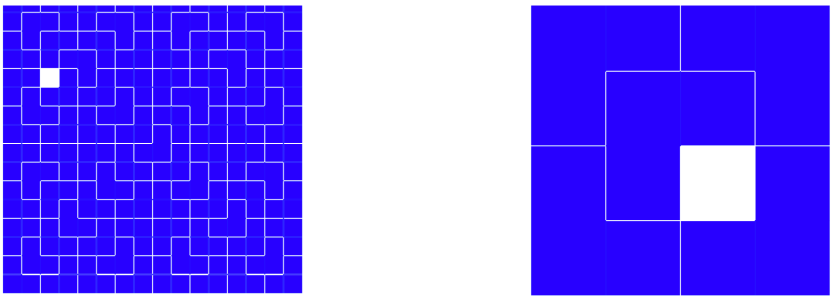
2) *start_y*: determines the starting row index of the checkerboard in consideration

3) *missing_x*: determines the column index of the missing square in consideration

4) *missing_y*: determines the row index of the missing square in consideration

5) *n*: determines the number of rows and columns (*n* x *n*) of the checkerboard in consideration

iii) Images Generated with Different Parameters



(Pic. 2) Result Images with Different Size of Checkerboards and Different Index of the Missing Squares