Recursive Graphics

- Sierpinski Triangle - Koch Snowflake - Hilbert Curve -

By: Kiana Ross & Nick Goltsos

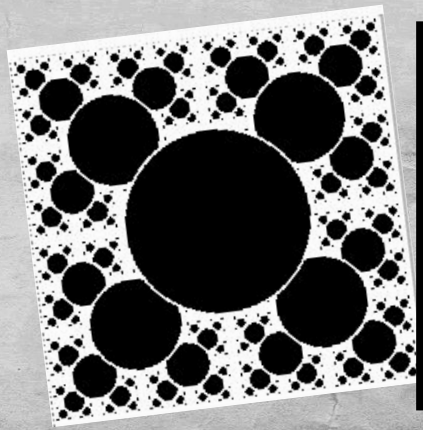
Github repository link

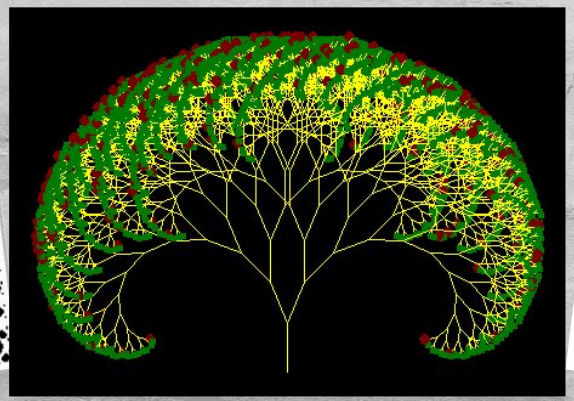
https://github.com/kianaross/CSC212Finalproject

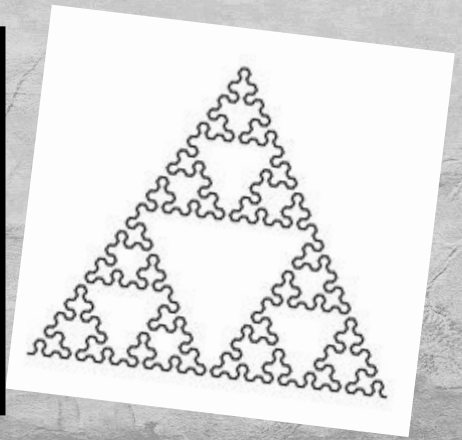
What is a Recursive Fractal?

- A fractal is a never-ending, recursion driven, pattern created by repeating a simple process over and over in an ongoing feedback loop.
- The recursive calls create graphics by drawing the same image in different sizes and angles, layering to create a more complex image.

**below are many examples of recursive graphics*





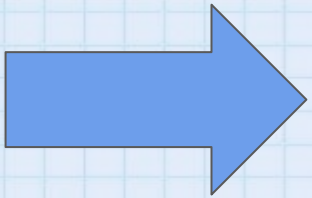


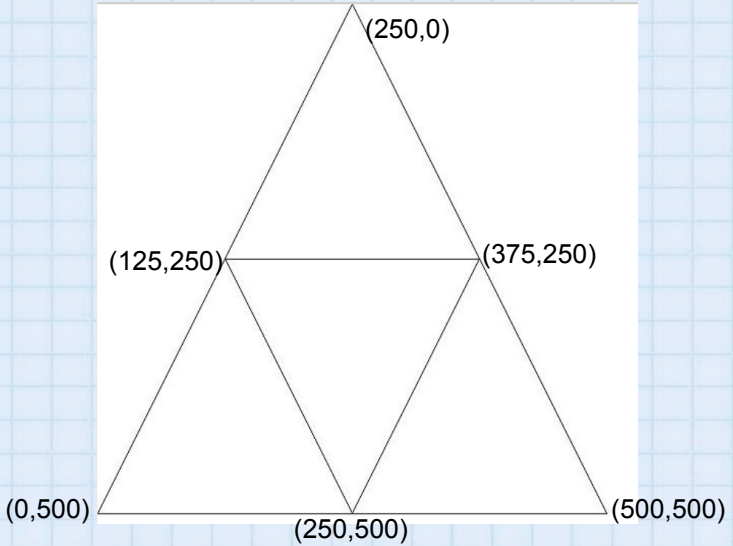
How To Draw Fractals?

Option 1: Use coordinate points to mark vertices of the fractals and then draw lines from one point to the other using a python program

A .txt file containing these lines/points……..Would output this image

```
1 250 0 0 500
2 250 0 500 500
3 0 500 500 500
4 125 250 375 250
5 125 250 250 500
6 375 250 250 500
```

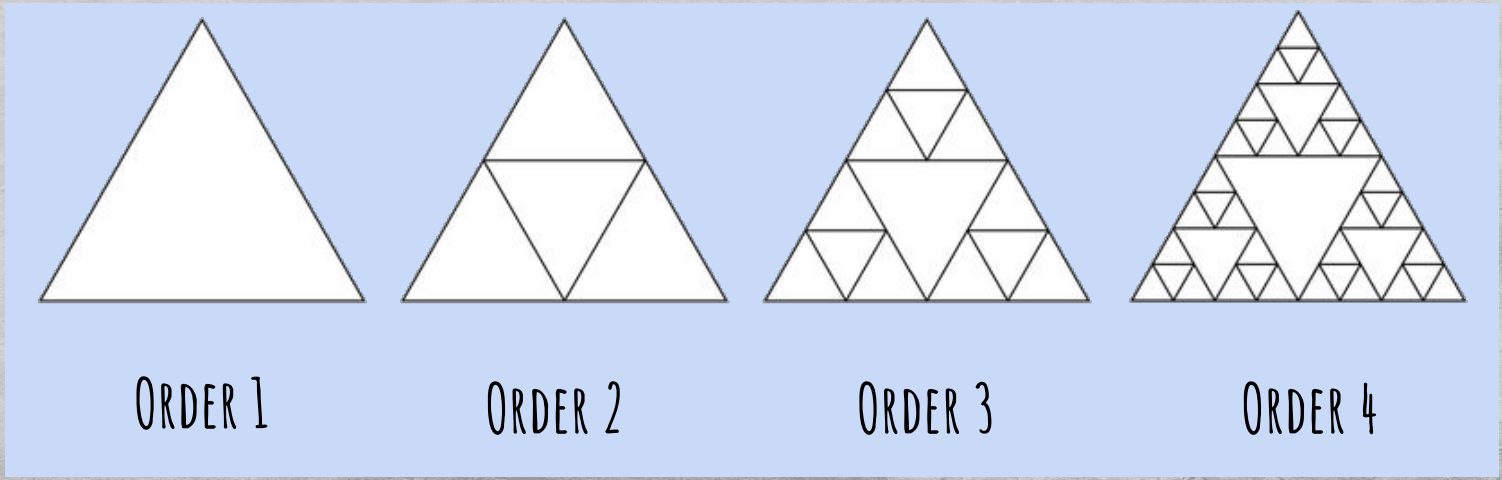




Sierpinski Triangle

HOW TO CREATE (Algorithm)

1. Begin by drawing initial equilateral triangle (with given vertices)
2. Create 3 smaller equilateral triangles inside the original, using the original vertices and their midpoints (ex. (x1+x2)/2, (y1+y2)/2)
3. Repeat process for all right-side up triangles (recursive step)



Sierpinski Triangle - Methods

CODE STRUCTURE

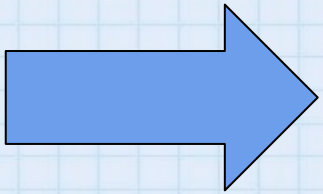
- Use fractal drawing option on the previous slide.
- Main function- 6 command line arguments (x1, y1, x2, y2, x3, y3),the vertices of starting triangle. Will call triangle function.
- **Void Triangle function**(Recursion happens here)
  - Base case- if length of triangle <10, stop
  - Else- Call draw function (x3), pushback 2 points (4 values) (points to draw line).
  - Recursively call triangle function, with 2 new midpoints and 1 prior vertice (3 times to account for each new triangle).
- V**oid Draw function**(called in Triangle Function)
  - takes 4 values and a vector (passed by reference as parameters)
  - set the 4 values to the first 4 values in vector (the 2 points to draw each line)
- Write vector to a txt file, and use a python program to create the PDF image.
- ***Two versions** - slightly altered program and python file to print different colors based on order by tracking order and pushing back into vector as well.

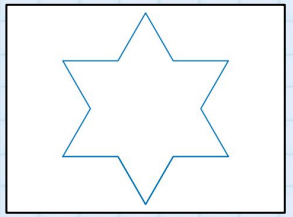
How To Draw Fractals?

Option 2: Use L-Systems, a process of creating instructions which state (F) draw a line (-) turn to the left by some degree (+) turn to the right by some degree. Operates by having a “turtle” walk in a space knowing its current position and orientation, as well as the next instruction. Involves replacement.

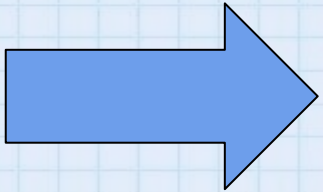
These l-system operations ……..... Print these images

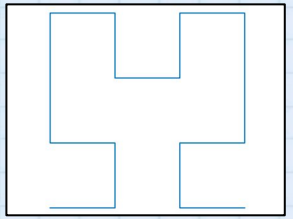
F-F++F-F++F-F++F-F++
F-F++F-F++F-F++F-F++
(with 60 degree turns)





+-F+F+F-F-+F-F-F+F+F-F-F
+-F-F+F+F-+
(with 90 degree turns)

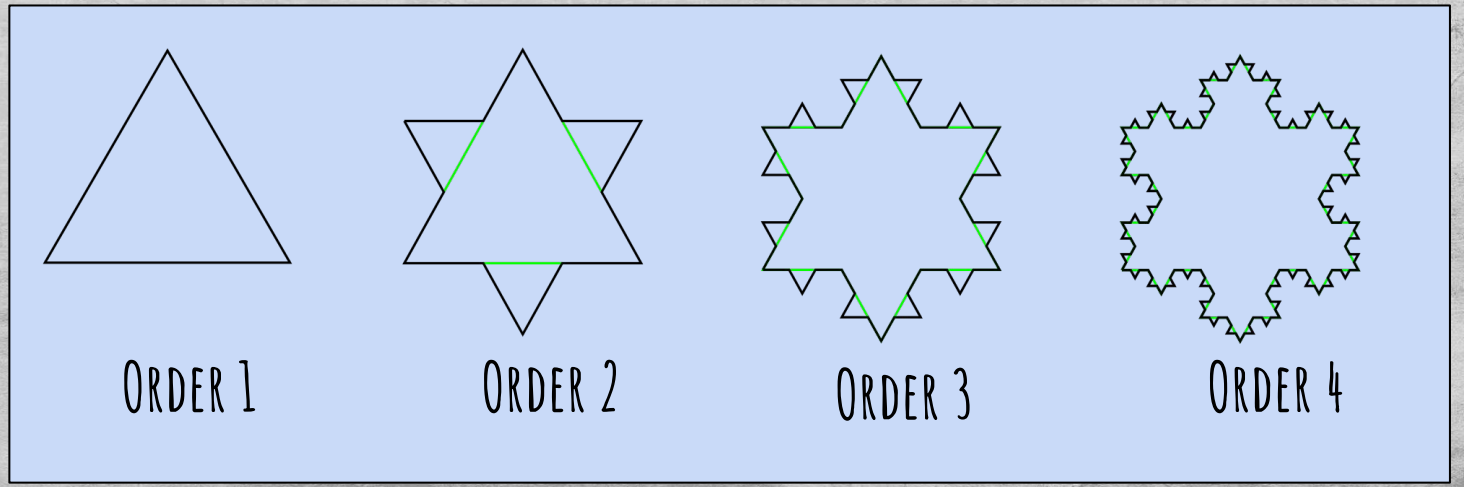




Koch Snowflake

HOW TO CREATE (Algorithm)

1. The Koch Snowflake begins as an equilateral triangle
2. The following iterations are formed by splitting segments into thirds and repeatedly adding smaller triangles in the middle third.
3. The snowflake can be split into 3 equivalent sides, thus we focused on creating 1/3 and then replicating through a 120 degree right hand turn.



Koch Snowflake - Methods

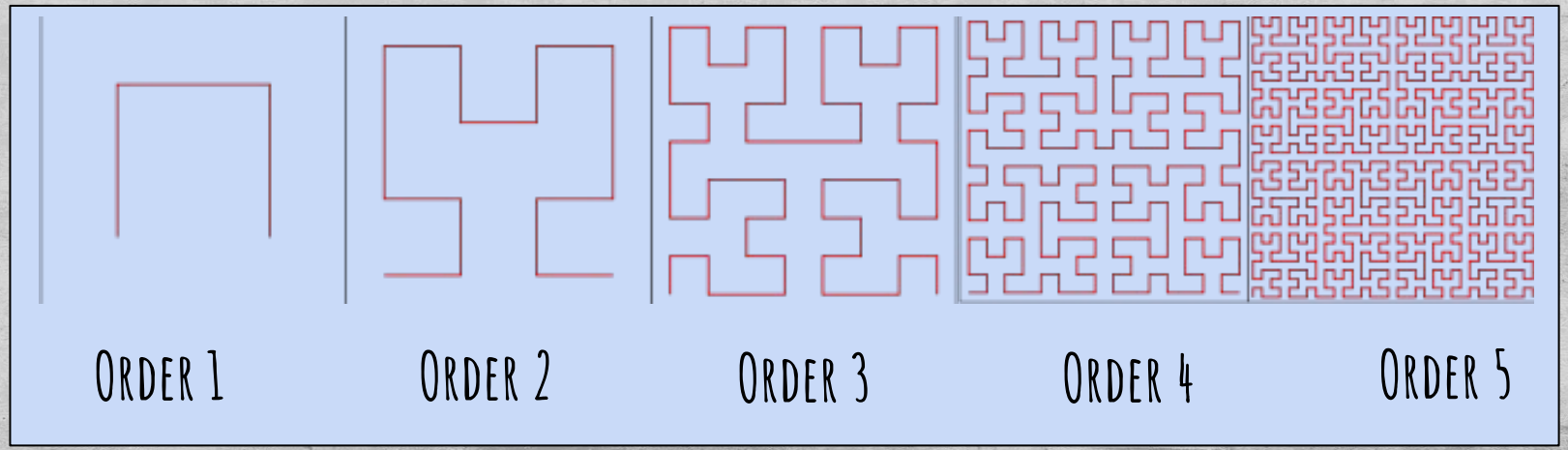
CODE STRUCTURE

- Created using the L-systems.
- The L-system initial string used F->F-F++F-, after each iteration rewrite with F as F string.
- Main function- receives order from user, calls string koch_snowflake function and writes result to output file.
- **String koch_snowfake function-** Uses a loop to draw the 3 sides of the snowflake, calling snowflake() and making a 120 degree right turn after each iteration.
- **Void snowflake**(recursion happens here)-take order and string passed by reference as parameters
  - Base case- if order of 0, return
  - Else- append the commands from string above. Before each (“F”), call snowflake recursively for order-1.
- PDF image created using python file.

Hilbert Curve

HOW TO CREATE (Algorithm)

1. The Hilbert Curve initially begins as an open box (open down in this case)
2. This box is then replicated 3 ways so there is a box in each quadrant
   a. Replicate across y axis (horizontal) (that of the closed sides)
   b. Replicate across the x-axis (vertical) and turned right 90 degrees
   c. Replicate cross x-axis (vertical) , moved right and turned left 90 degrees
3. These replicas are connected and the process is then repeated recursively



Hilbert Curve - Methods

CODE STRUCTURE

- Created using L-systems.
- Unlike Koch snowflake, where we only use one string of commands, Hilbert curve requires 2 because of its complexity. We will use string replacement and box will draw from lower right.
  - A->+BF-AFA-FB+
  - B->-AF+BFB+FA-
- Main function- receives order from user, calls string hilbertcurve function and writes result to output file.
- **String hilbertcurve function-** adds moves for A to a string, calling astring function and bstring function where an A or B appears.
- **Void astring & bstring -** take order and string passed by reference as parameters
  - Base case- order = 1, return (result would +F-F-F+)
  - Else- append commands for the string, calling astring and bstring functions accordingly for order-1.
  - PDF image created using python file where user enters degree (90).

L-System vs. Points- *Which method is better?*

L-systems

Easier to visualize and work with.

Can look at text file and clearly determine how the drawing is created (F, -, +)

Less control over how information is printed (can't specify points to draw at)

Point Systems

Are more complex to visualize and implement as it is not a continuous line approach like L-systems

While more complex offers control over how the information is printed

Can be paired with color changing based on order level

Now we'll run our programs so you can see for yourself