IAE 101 – Introduction to Digital Intelligence
Fall, 2024
Programming Project 02 – Drawing Sierpinski Triangles

Assigned: Friday, October 11th, 2024
Due: **Thursday, October 24th, 2024**

Description:
This project will allow to exercise your skills at computer drawing, using the pygame library, and recursive programming by implementing an algorithm to draw Sierpinski Triangles.

The Sierpinski Triangle is a fractal pattern in which a triangle (usually an equilateral triangle) is inscribed with three smaller triangles. This pattern is repeated within each new set of triangles, creating a complex image. However, if the image is examined closely—if you zoom in on a part of the Triangle—you will see the same image repeated over again.

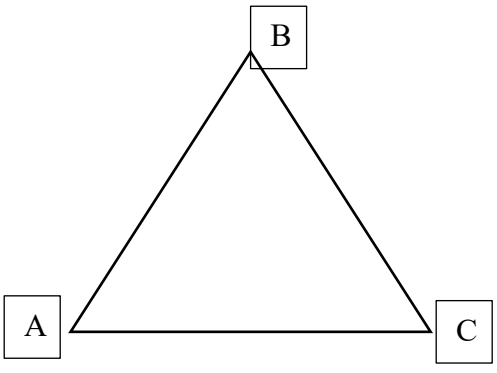
The Sierpinski Triangle, also known as the Sierpinski Sieve or the Siepinski Gasket, is named after Waclaw Sierpinski, a Polish Mathematician (1882 – 1969) who described the pattern, or set of steps, for constructing it in 1915. However, the pattern appears in Western art as early as the 13th Century.

One procedure for constructing a Sierpinski Triangle works like this.

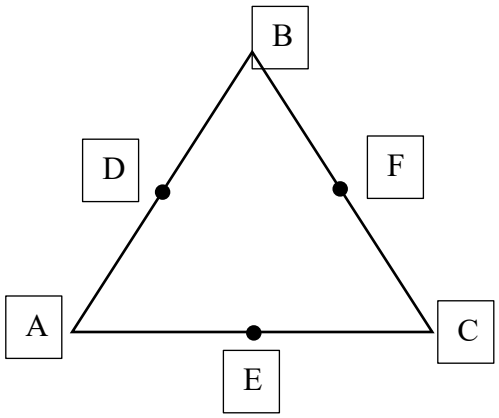
Input: We begin with a triangle.
Step 1: Find the midpoint of each side of the triangle.
Step 2: Draw line between the midpoints to create three new triangles inside the original triangle.
Repeat Steps 1 and 2 for each new triangle you create.

Please consult the diagram below.

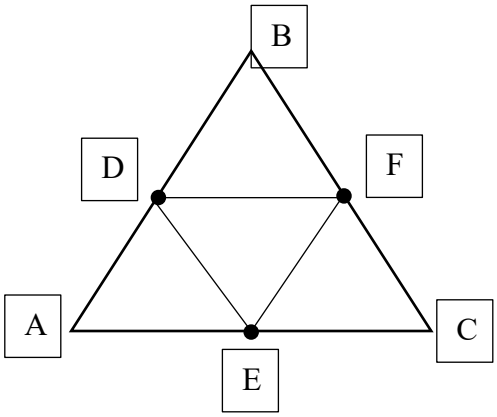
Step 0: Start with a triangle of points: A, B, and C.



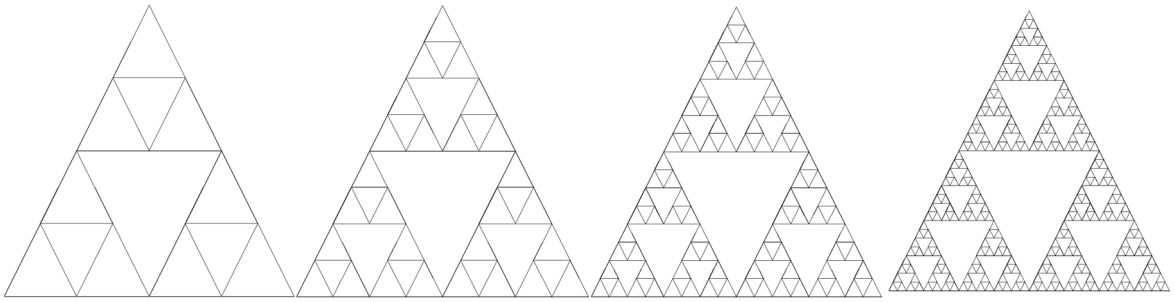
Step 1: Find the midpoint of each side of the triangle: D, E, and F.



Step 2: Connect the midpoints: D, E, and F to create three new triangles: (A, D, E), (D, B, F), and (E, F, C).



This is a *degree* 1 Sierpinski Triangle. The degree describes the number of levels of recursion used when constructing the Triangle. In this example, we drew one set of triangles inside the original triangle, so this is a *degree* 1 Sierpinski Triangle. To get a *degree* 2 Sierpinski Triangle, we must recursively draw new triangles inside the each of the three triangles we just created repeating the exact same procedure. Examples of higher degree Sierpinski triangles are given below.



Installing Pygame:

This command must be run from the command line (not the Idle Interactive Shell).

On Windows, hit the windows key and then enter "CMD". This will open the windows command prompt, where you can enter the installation command.

```
py -3 -m pip install pygame --user
```

On Mac, use the application search to find the "terminal" application. The terminal gives you a command line, where you can enter the installation command.

```
python3 -m pip install pygame --user
```

After installing Pygame, you should be able to import it without any trouble.

Assignment:

Your assignment is to write the recursive function to construct a Sierpinski Triangle of a given degree.

Inside the file sierpinski.py is a function called `sierpinski()`. The function takes several inputs.

- The first parameter, **degree**, is the degree of the Sierpinski Triangle—how many further levels of recursion will be applied when drawing the Triangle.
- The next three parameters, **p1**, **p2**, **p3**, are the points that will be used to draw the next triangle.
- The fourth parameter, **color**, is the initial color that will be used to draw the Triangle.
- The fifth parameter, **line_width**, is used to specify the width of the line used to draw the Triangle. This should be a positive integer. If it is set to 0, then the Triangle will be filled with the selected color when drawn.
- The final parameter, **screen**, is the pygame surface object upon which the Triangle will be drawn. You should not have to change the value passed to this parameter.

You are given a function, `draw_triangle()`, that will draw a triangle on the given surface at the given points in the given color.

You must implement the function `find_midpoint()`, which takes two points as input, and which returns the point that is midway between the two input points.

Students should use the `draw_triangle()` and `find_midpoint()` functions in order to implement the `sierpinski()` function. The iustmplementation m be recursive. The `sierpinski()` function will call itself. Think about how many recursive calls must be made

in each call to the `sierpinski()` function. Think about when and how to stop the recursion so that you construct a Sierpinski Triangle of the given degree.

Experiment with color, line width, etc. to see what sort of effects you can produce when drawing the Sierpinski Triangle.

Submission:

Turn in the sierpinski.py file with the completed `find_midpoint()` and `sierpinski()` functions through Blackboard.

A diversion: fractal dimension:
In grade school, we learn that the dimension of a line segment is one, the dimension of a square is two, and the dimension of a cube is three. But you probably didn't learn what is really meant by *dimension*. How can we express what it means mathematically or computationally?

Formally, we can define the *Hausdorff dimension* or *similarity dimension* of a self-similar figure by partitioning the figure into a number of self-similar pieces of smaller size. We define the dimension to be the log (# self-similar pieces) / log (scaling factor in each spatial direction). For example, we can decompose the unit square into 4 smaller squares, each of side length 1/2; or we can decompose it into 25 squares, each of side length 1/5. Here, the number of self-similar pieces is 4 (or 25) and the scaling factor is 2 (or 5). Thus, the dimension of a square is 2 since log (4) / log(2) = log (25) / log (5) = 2. We can decompose the unit cube into 8 cubes, each of side length 1/2; or we can decompose it into 125 cubes, each of side length 1/5. Therefore, the dimension of a cube is log(8) / log (2) = log(125) / log(5) = 3.

We can also apply this definition directly to the Sierpinski Triangle. We can decompose the unit Sierpinski Triangle into 3 Sierpinski Triangles, each of side length 1/2. Thus, the dimension of a Sierpinski Triangle is log (3) / log (2) ≈ 1.585. Its dimension is fractional—more than a line segment, but less than a square! With Euclidean geometry, the dimension is always an integer; with fractal geometry, it can be something in between. Fractals are similar to many physical objects; for example, the coastline of Britain resembles a fractal, and its fractal dimension has been measured to be approximately 1.25.