

# Computational Physics Problem Set

Saleh Shamloo Ahmadi  
Student Number: 98100872

September 27, 2021

## 1 Fractal Generation

Fractals are self-similar shapes, so they can be generated by repeatedly applying mappings to a set of points. The fractals in this problem set can all be created with linear iterated function systems (IFS). The transformations for these fractals can be described with affine transformations:

$$\mathbf{v}_{i+1,j} = T_j \mathbf{v}_{i,k} + \mathbf{c}_j \quad (1)$$

$$\mathbf{v}_{i,j} = \begin{pmatrix} x_{i,j} \\ y_{i,j} \end{pmatrix} \quad (2)$$

$$T_j = \begin{pmatrix} a_j & b_j \\ c_j & d_j \end{pmatrix}, \quad \mathbf{c}_j = \begin{pmatrix} e_j \\ f_j \end{pmatrix} \quad (3)$$

There are two ways to define the mappings: relative and absolute. Also, the fractals can be generated randomly with the absolute mappings.

### 1.1 Relative Mapping

This type of mapping is suitable for use with line-based fractals (i.e. the fractal consists of a single curve). In this method, the transformations are applied locally on each line; The origin of the coordinate system is set to on end of the line and the point on the other end of the line is transformed relative to that. Algorithm 1 is the implementation of this method.

### 1.2 Absolute Mapping

This type of mapping is suitable for non-line-based fractals (the fractal isn't only a single curve). In this method, the same transformations are applied to every point

---

**Algorithm 1** Fractal Generation by Relative Mapping

---

```
1: function FRACTAL( $x_{init}, y_{init}, T, \mathbf{c}, steps$ )  $\triangleright T$  and  $\mathbf{c}$  represent the matrix and  
   vector for every transformation  
2:    $x \leftarrow x_{init}$   
3:    $y \leftarrow y_{init}$   
4:   for  $step \leftarrow 1, steps$  do  
5:      $i \leftarrow 1$   
6:     while  $i < N(x)$  do  $\triangleright N(x)$  is the number of  $x$  coordinates (which is  
       the same as the number of points)  
7:        $\begin{pmatrix} x_{rel} \\ y_{rel} \end{pmatrix} \leftarrow \begin{pmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \end{pmatrix}$   
8:       for all  $T_j$  and  $\mathbf{c}_j$  do  
9:          $\begin{pmatrix} x_{new} \\ y_{new} \end{pmatrix} \leftarrow T_j \begin{pmatrix} x_{rel} \\ y_{rel} \end{pmatrix} + \mathbf{c}$   
10:        insert  $x_{new}$  into  $x$  at index  $i$   
11:        insert  $y_{new}$  into  $y$  at index  $i$   
12:         $i \leftarrow i + 1$   
13:      end for  
14:    end while  
15:  end for  
16:  return  $x$  and  $y$   
17: end function
```

---

in each step to generate new points. The origin of the coordinate system is always set at the first point of the fractal. By repeatedly applying the transformations to the fractal in each step, the self-similar shape is created. Algorithm 2 is the implementation of this method.

---

**Algorithm 2** Fractal Generation by Absolute Mapping

---

```

1: function FRACTAL( $x_{init}, y_{init}, T, \mathbf{c}, steps$ )  $\triangleright T$  and  $\mathbf{c}$  represent the matrix and
   vector for every transformation
2:    $x \leftarrow x_{init}$ 
3:    $y \leftarrow y_{init}$ 
4:   for  $step \leftarrow 1, steps$  do
5:     define the  $x_{new}$  empty array
6:     define the  $y_{new}$  empty array
7:     for all  $x_i \in x$  and  $y_i \in y$  do
8:       for all  $T_j$  and  $\mathbf{c}_j$  do
9:          $\begin{pmatrix} x_{gen} \\ y_{gen} \end{pmatrix} \leftarrow T_j \begin{pmatrix} x_{rel} \\ y_{rel} \end{pmatrix} + \mathbf{c}$ 
10:        append  $x_{gen}$  to  $x_{new}$ 
11:        append  $y_{gen}$  to  $y_{new}$ 
12:       end for
13:     end for
14:      $x \leftarrow x_{new}$ 
15:      $y \leftarrow y_{new}$ 
16:   end for
17:   return  $x$  and  $y$ 
18: end function

```

---

### 1.3 Random Fractals

Some fractals can be generated by repeatedly applying random transformations from the fractal. If this is done with enough sample points, and enough steps so that the smallest unit of the shape is smaller than a pixel of the display, the fractal will be perfect. Algorithm 3 is the implementation of this method.

### 1.4 Representation of the fractals

The fractals are defined by functions  $\{f_1, f_2, \dots, f_n\}$ . Each of the functions are

$$f_i(\mathbf{v}) = T_i \mathbf{v} + \mathbf{c}_i, \quad (4)$$

---

**Algorithm 3** Fractal Generation by Absolute Mapping

---

```
1: function FRACTAL( $range_x, range_y, T, \mathbf{c}, steps, samples$ )  $\triangleright T$  and  $\mathbf{c}$  represent  
   the matrix and vector for every transformation  
2:   generate random  $x$  values in  $range_x$  ( $N(x) = samples$ )  
3:   generate random  $y$  values in  $range_y$  ( $N(y) = samples$ )  
4:   for all  $x_i \in x$  and  $y_i \in y$  do  
5:     for  $step \leftarrow 1, steps$  do  
6:       choose random  $T_j$  and  $\mathbf{c}_j$   
7:        $\begin{pmatrix} x_i \\ y_i \end{pmatrix} \leftarrow T_j \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \mathbf{c}_j$   
8:     end for  
9:   end for  
10:  return  $x$  and  $y$   
11: end function
```

---

where

$$T = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} e \\ f \end{pmatrix}. \quad (5)$$

So we can represent each fractal by defining the values  $a, b, c, d, e$ .

## 2 Koch Snowflake

transformation	$a$	$b$	$c$	$d$	$e$	$f$
$f_1$	1/3	0	0	1/3	0	0
$f_2$	1/6	$-\sqrt{3}/6$	$\sqrt{3}/6$	1/6	1/3	0
$f_3$	1/6	$-\sqrt{3}/6$	$\sqrt{3}/6$	1/6	1/2	$\sqrt{3}/6$
$f_4$	1/3	0	0	1/3	2/3	0

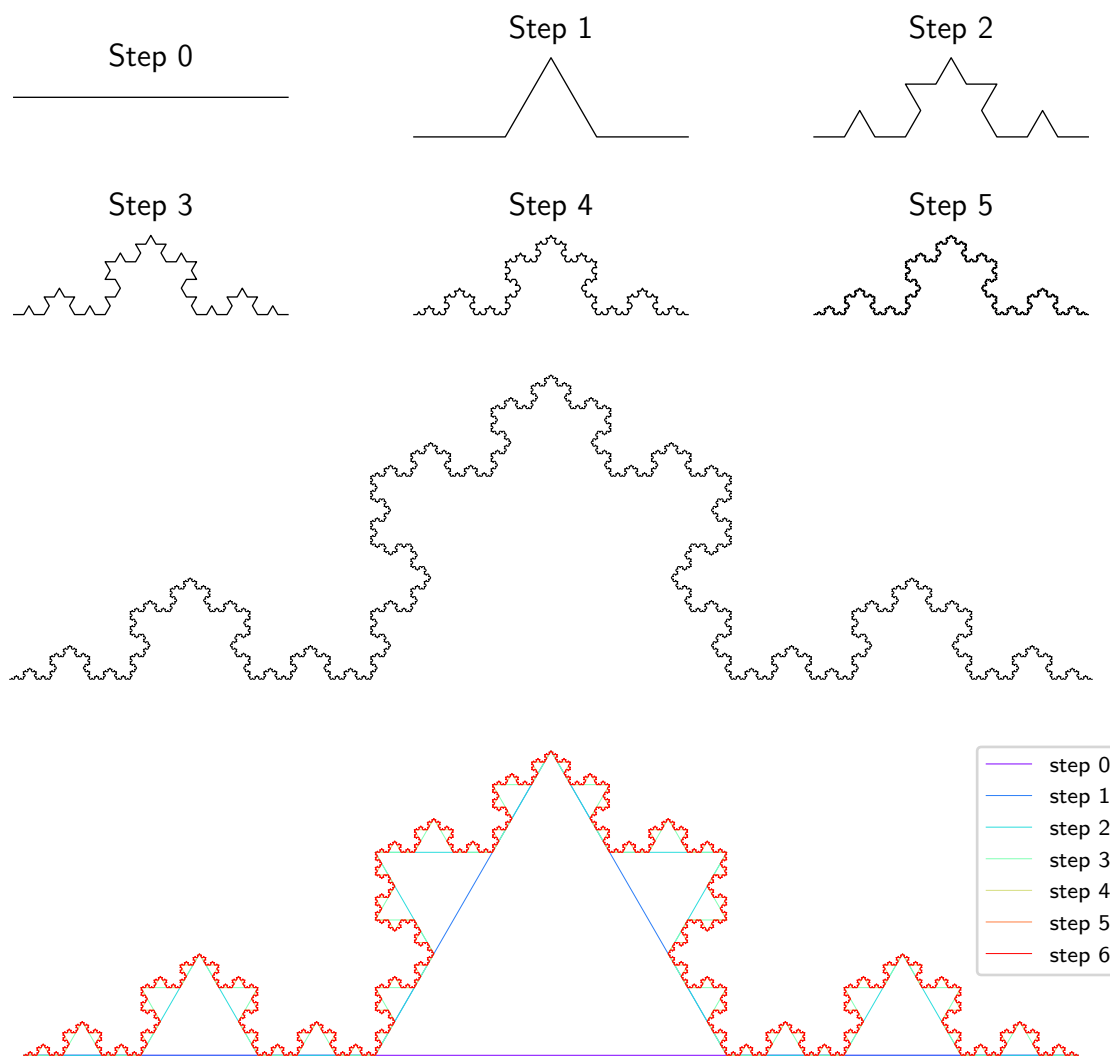


Figure 1: Koch Curve

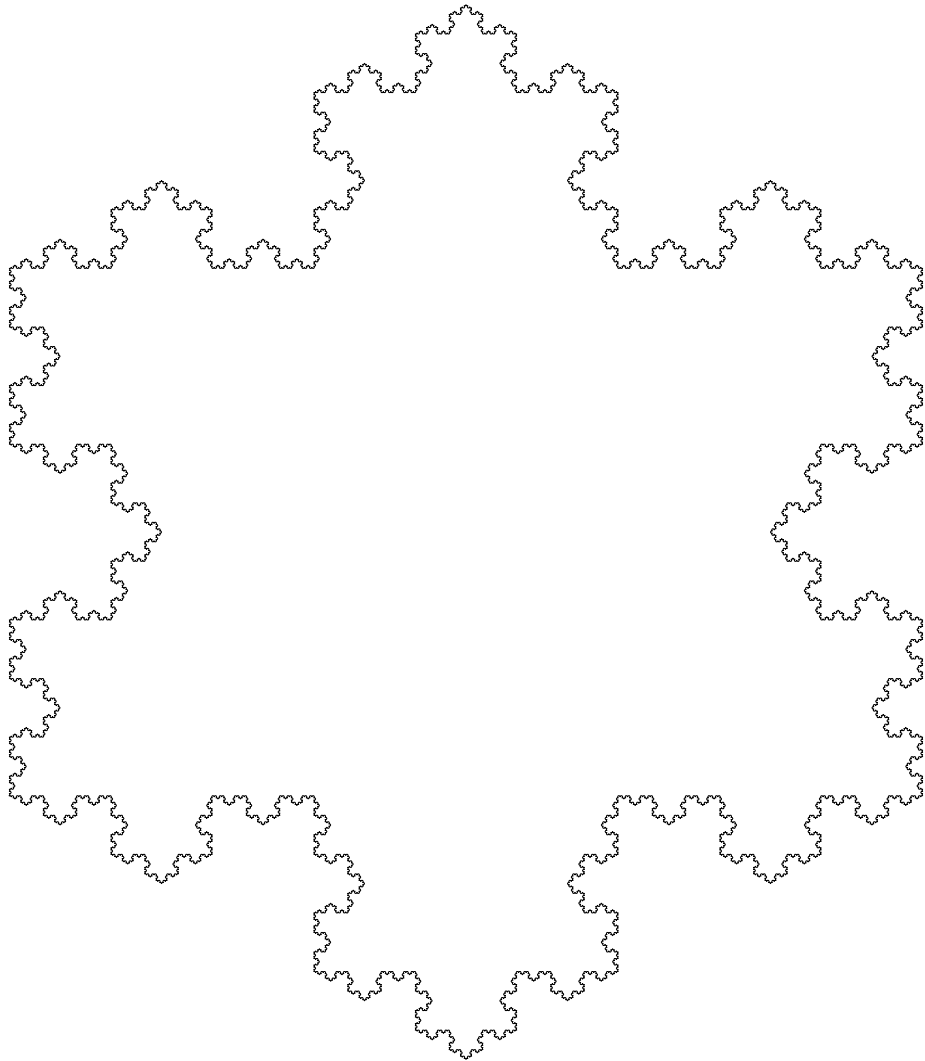


Figure 2: Koch Snowflake

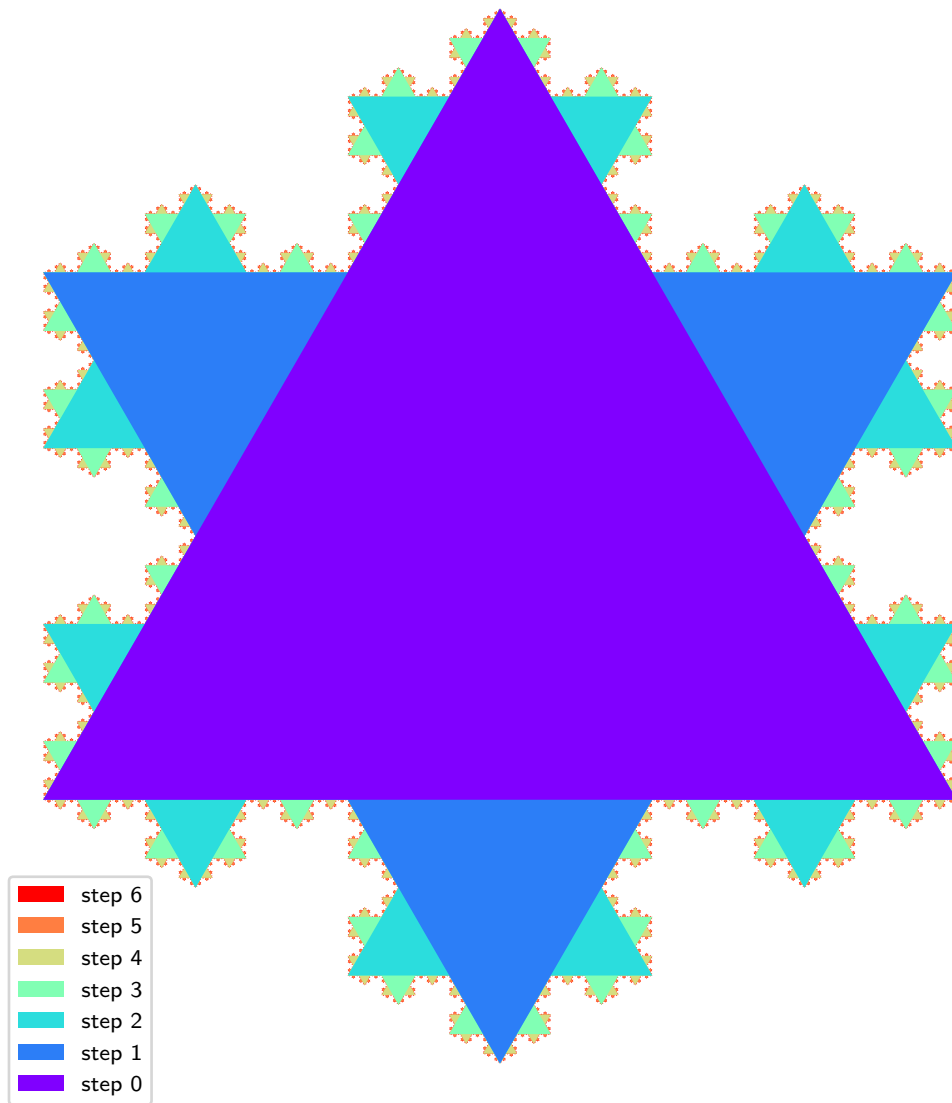


Figure 3: Colored Koch Snowflake

### 3 Highway Dragon

transformation	$a$	$b$	$c$	$d$	$e$	$f$
$f_1$	$1/2$	$1/2$	$-1/2$	$1/2$	$0$	$0$
$f_2$	$-1/2$	$1/2$	$-1/2$	$-1/2$	$1/2$	$-1/2$



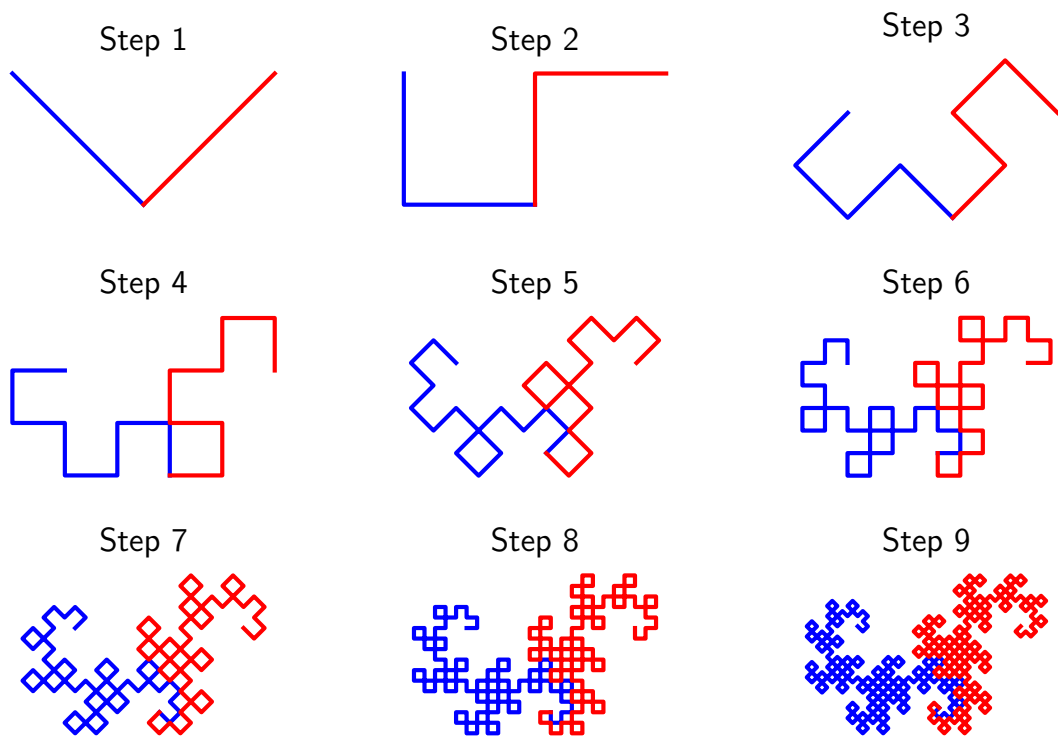


Figure 4: Heighway Dragon

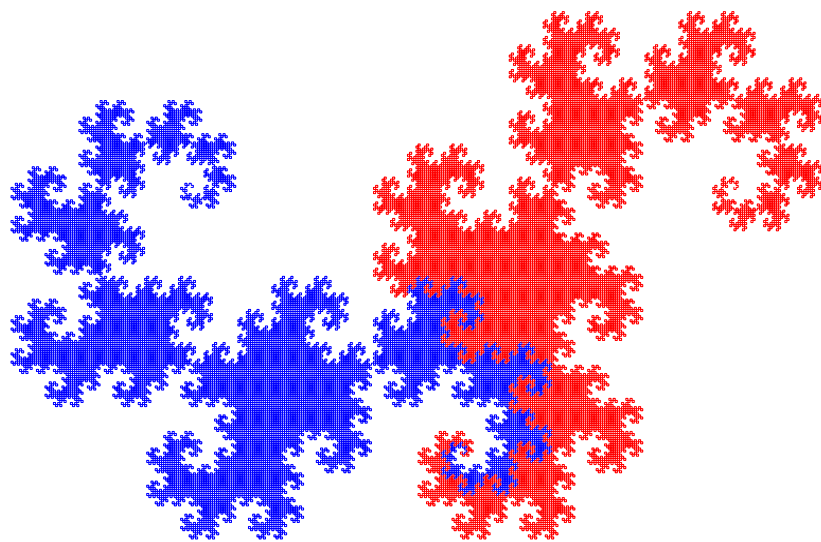
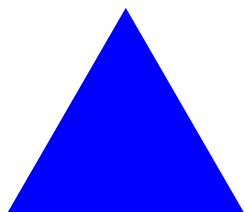


Figure 5: Heighway Dragon

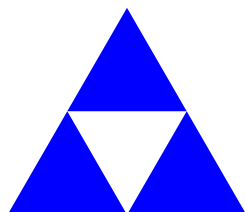
## 4 Sierpiński Triangle

transformation	$a$	$b$	$c$	$d$	$e$	$f$
$f_1$	$1/2$	$0$	$0$	$1/2$	$0$	$0$
$f_2$	$1/2$	$0$	$0$	$1/2$	$1/4$	$\sqrt{3}/4$
$f_3$	$1/2$	$0$	$0$	$1/2$	$1/2$	$0$

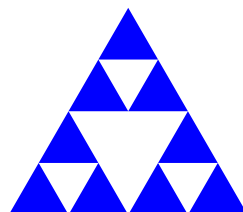
Step 0



Step 1



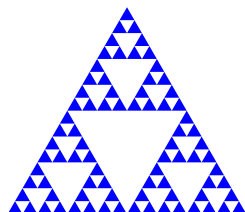
Step 2



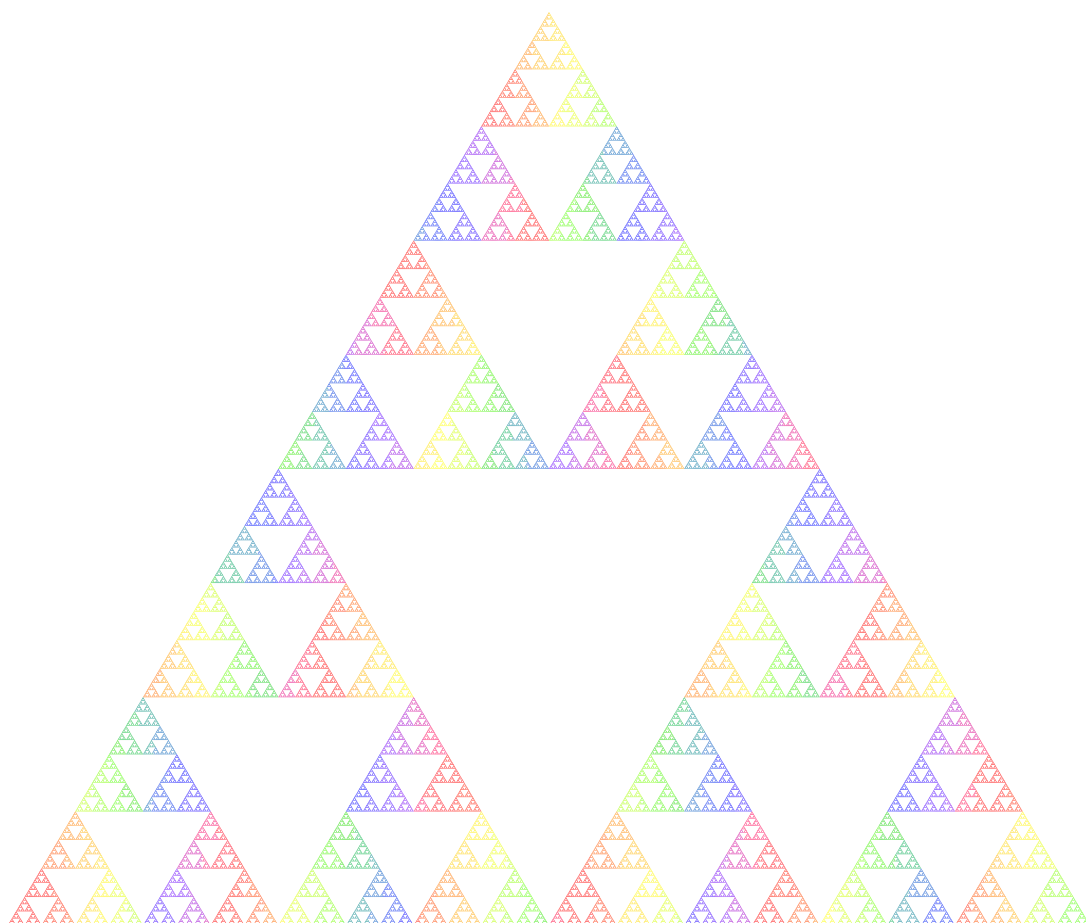
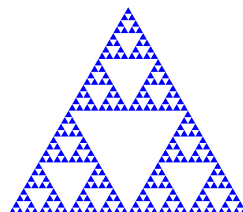
Step 3



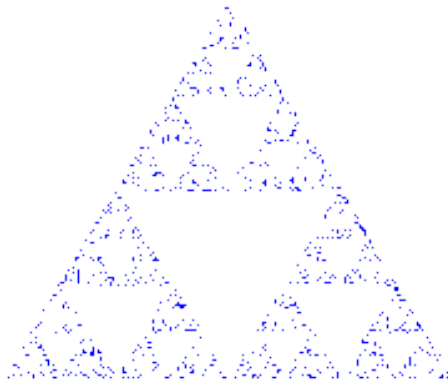
Step 4



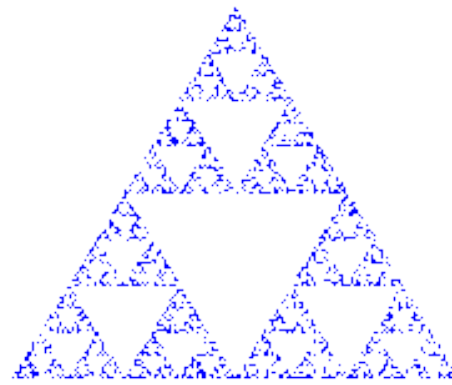
Step 5



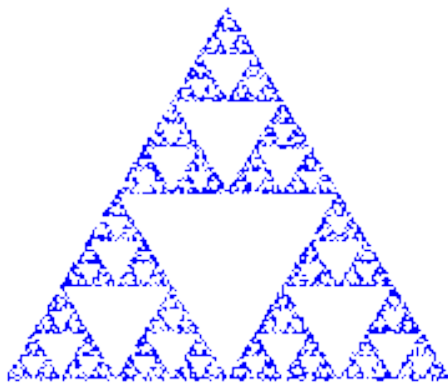
1000 Samples



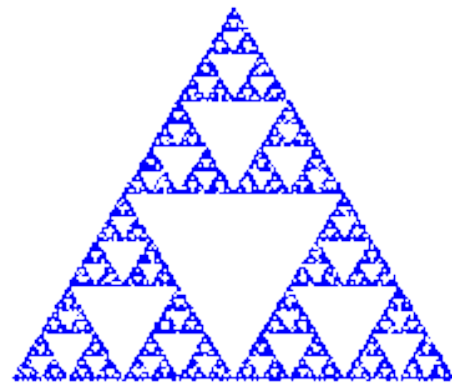
2500 Samples



5000 Samples



10000 Samples



50000 Samples

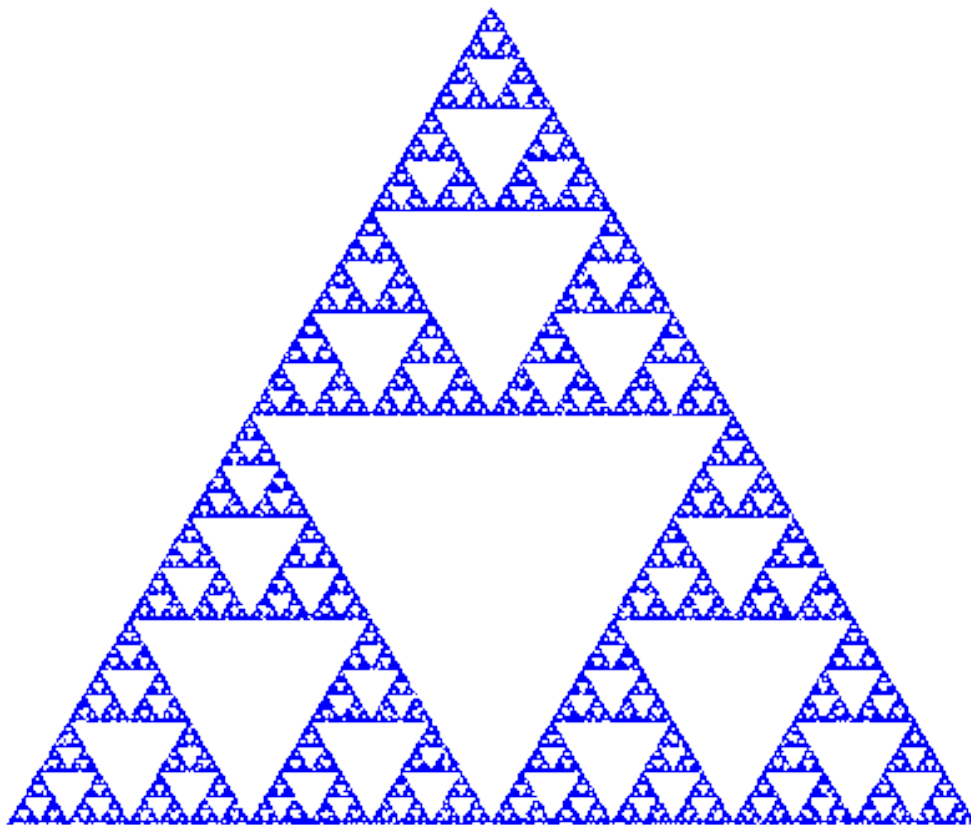
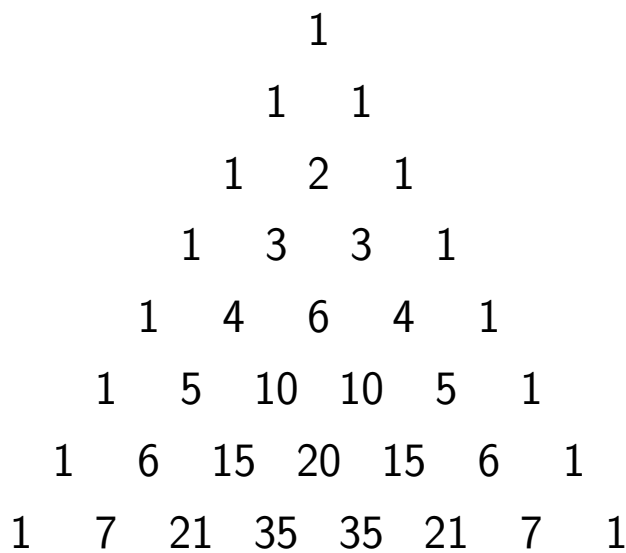
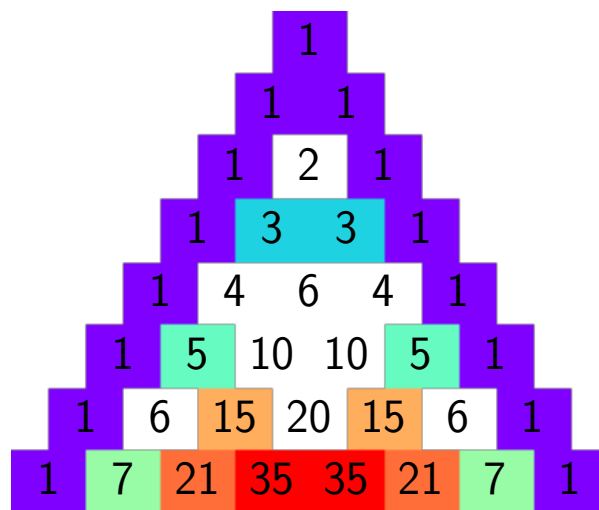


Figure 7: Randomly Generating the Sierpiński Triangle

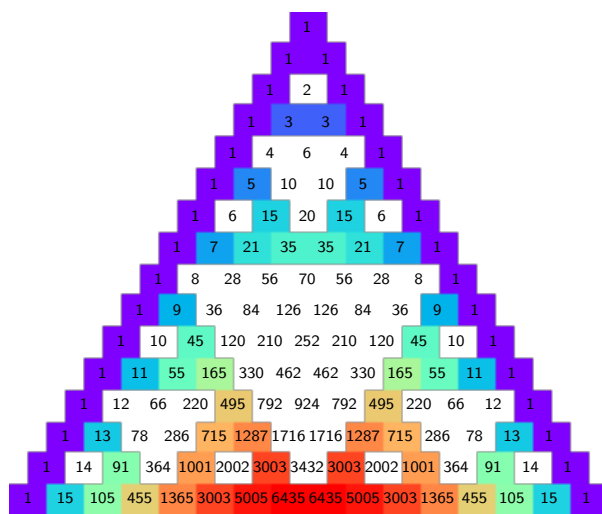
## 4.1 The Sierpiński Triangle Inside Pascal's Triangle



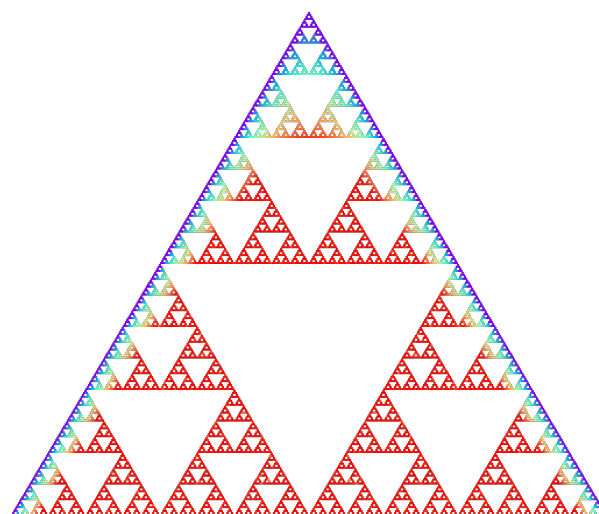
(a) Pascal's Triangle



(b) Connecting the odd numbers generates the Sierpiński triangle



(c) The Sierpiński triangle inside a larger Pascal's triangle



(d) Painted on Pascal's triangle of size 256

## 5 Barnsley Fern

To create this fractal, we need to apply random functions with different probabilities.

transformation	$a$	$b$	$c$	$d$	$e$	$f$	probability
$f_1$	0	0	0	0.16	0	0	1%
$f_2$	0.85	0.04	-0.04	0.85	0	1.6	85%
$f_3$	0.20	-0.26	0.23	0.22	0	1.6	7%
$f_4$	-0.15	0.28	-0.26	0.24	0	0.44	7%



Figure 9: Barnsley fern with 200000 random samples and 25 transformation steps