****

**A PROJECT REPORT ON**

**DEMONSTRATION OF SIERPINSKI TRIANGLE**

**Project Report submitted by Srichaitanya C (21BCE1508),**

**B.Tech 2nd year, CSE Core,**

**Vellore Institute Of Technology, Chennai**

**ABSTRACT:**

This project report aims to demonstrate the implementation of the Sierpinski Triangle using the Tkinter library in Python. The Sierpinski Triangle is a well-known fractal pattern generated by iteratively dividing an equilateral triangle into smaller triangles. Tkinter is a popular standard Python library that provides a simple way to create graphical user interfaces.

In this project, we explore how Tkinter can be utilized to develop a visually appealing and interactive program that generates the Sierpinski Triangle. The report provides a step-by-step guide to building the program, including a detailed explanation of the code used. We start by creating the main window and drawing the initial equilateral triangle. Then, we use iteration to divide each of the smaller triangles into even smaller triangles. The resulting pattern is the Sierpinski Triangle. The report also includes screenshots of the generated fractal of the Sierpinski Triangle.

Overall, this project serves as an example of how programming and mathematics can be combined to create complex and captivating patterns. It demonstrates the use of Tkinter to create a visually appealing program that generates the Sierpinski Triangle while providing insight into the mathematical principles behind the fractal. This project is an excellent resource for anyone interested in learning more about fractals, iteration, and using Tkinter in Python programming.

**Table of contents:**

|  |  |  |
| --- | --- | --- |
| **Sl.No.** | **Topics** | **Page No.** |
| 1 | Introduction | 4 |
| 2 | Objective | 6 |
| 3 | Methodology | 7 |
| 4 | Code | 8 |
| 5 | Outputs | 10 |
| 6 | Result | 17 |
| 7 | Conclusion | 18 |
| 8 | References | 19 |

**Introduction:**

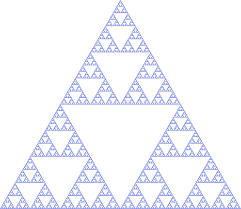
The Sierpinski triangle is a fractal named after Polish mathematician Waclaw Sierpinski. It is created by repeatedly dividing equilateral triangles into smaller triangles and removing the central triangle. The resulting pattern is a fractal that exhibits self-similarity and has interesting geometric properties. In this project, we have created the Sierpinski triangle using Python's tkinter library.

It is a beautiful geometric pattern with a complex structure that has fascinated mathematicians for centuries. As part of my project, I used Python's Tkinter library to create a graphical representation of the Sierpinski Triangle and explore its intricate properties.

In my report, I first provide an overview of the Sierpinski Triangle and its iterative construction using an equilateral triangle. Next, I discuss how I implemented the algorithm for generating the Sierpinski Triangle using Tkinter, including the use of iteration and drawing functions. I provide code snippets and graphical outputs to illustrate the process.

Finally, I conclude my report by reflecting on the challenges and insights gained from the project, as well as the potential for further exploration of the Sierpinski Triangle using Tkinter and other programming languages.

Overall, my project report provides an engaging and hands-on exploration of the Sierpinski Triangle using Python's Tkinter library, highlighting its beauty and complexity in a tangible way.

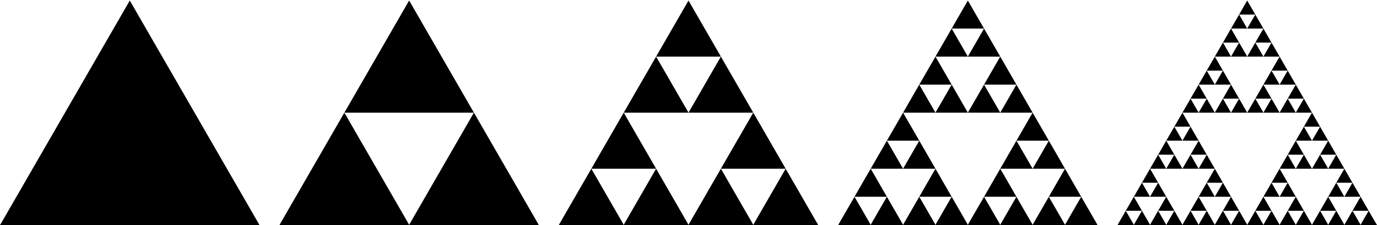


Wacaw Sierpiski introduced the Sierpiski triangle in 1915. However similar patterns can be found as a repeating motif in Cosmatesque inlaid stonework from the 13th century.

The Apollonian gasket, the curved forerunner of the Sierpiski triangle, was first introduced by Apollonius of Perga in the third century BC and thoroughly investigated by Gottfried Leibniz in the seventeenth century.

The Sierpinski triangle shows the basic concepts of fractals, including how a pattern can repeat itself repeatedly at various scales and how this complicated form can be created from straightforward repetition.

The evolution of the Sierpinski Triangle



**Objective:**

The objective of my project report is to explore the Sierpinski Triangle, a fascinating geometric figure, using Python's Tkinter library to create a graphical representation of its iterative construction. Through this project, I aim to achieve the following objectives:

1. Gain a deeper understanding of the properties and structure of the Sierpinski Triangle, including its self-similarity and fractal nature.
2. Develop proficiency in using Python's Tkinter library to create graphical representations of mathematical objects.
3. Implement the iterative algorithm for generating the Sierpinski Triangle using Python and Tkinter, and explore its functionality and limitations.
4. Reflect on the challenges and insights gained from the project, and the potential for further exploration of the Sierpinski Triangle using Tkinter and other programming languages.

Overall, my project aims to provide a comprehensive and engaging exploration of the Sierpinski Triangle, highlighting its beauty and complexity in a tangible and visual way through the use of Python's Tkinter library.

**Methodology:**

The methodology for my project report on the Sierpinski Triangle using Python's Tkinter library involves the following steps:

1. Background research: Conducting extensive research on the Sierpinski Triangle, including its history, properties, and applications in various fields.
2. Understanding Tkinter: Gaining a solid understanding of Tkinter, including its features, functionality, and syntax, through online tutorials and documentation.
3. Planning and designing the program: Creating a detailed plan for the program's design, including the use of iteration, drawing functions, and user interface elements.
4. Implementation: Implementing the program using Python and Tkinter, testing and debugging as necessary, and ensuring the program runs smoothly and efficiently.
5. Analysis: Analysing the program's output, including visual representation of the Sierpinski Triangle, and comparing it to theoretical expectations and known properties of the triangle.
6. Reflection: Reflecting on the challenges and insights gained from the project, including any limitations of the program or areas for further improvement.
7. Documentation: Creating a detailed project report documenting the methodology, results, and reflections from the project.

Overall, this methodology allows for a structured and comprehensive exploration of the Sierpinski Triangle using Python's Tkinter library, ensuring a thorough understanding of its properties and providing a tangible visual representation of this fascinating geometric figure.

**Code:**

import tkinter as tk

import math

import random

import time

class App:

def \_\_init\_\_(self, master):

self.R = 300/math.sqrt(3)

rad = 4.71239

self.a = (300+self.R\*math.cos(rad),300+self.R\*math.sin(rad))

self.b = (300+self.R\*math.cos(rad+(2\*math.pi)/3),300+self.R\*math.sin(rad+(2\*math.pi)/3))

self.c = (300+self.R\*math.cos(rad+(4\*math.pi)/3),300+self.R\*math.sin(rad+(4\*math.pi)/3))

self.steps = 0

self.current\_coord = (0,0)

self.master = master

self.canvas = tk.Canvas(master, width=600, height=600, borderwidth=2, relief='solid')

self.canvas.create\_oval(self.a[0]-4,self.a[1]-4,self.a[0]+4,self.a[1]+4, fill="red")

self.canvas.create\_oval(self.b[0]-4,self.b[1]-4,self.b[0]+4,self.b[1]+4, fill="red")

self.canvas.create\_oval(self.c[0]-4,self.c[1]-4,self.c[0]+4,self.c[1]+4, fill="red")

self.canvas.pack()

self.x1\_button = tk.Button(master, text="x1", command=self.x1)

self.x1\_button.pack(side='left')

self.x10\_button = tk.Button(master, text="x10", command=self.x10)

self.x10\_button.pack(side='right')

self.x100\_button = tk.Button(master, text="x100", command=self.x100)

self.x100\_button.pack(side='right')

self.x1000\_button = tk.Button(master, text="x1000", command=self.x1000)

self.x1000\_button.pack(side='right')

def draw\_circle(self,coords,color,size):

self.canvas.create\_oval(coords[0]-size,coords[1]-size,coords[0]+size,coords[1]+size,fill=color,outline=color)

def half\_point(self,current\_coord,root):

x = (current\_coord[0]+root[0])/2

y = (current\_coord[1]+root[1])/2

return (x,y)

def x1(self):

if self.steps==0:

self.current\_coord = self.half\_point(self.a,self.b)

self.draw\_circle(self.current\_coord,"green",2)

self.steps+=1

else:

self.draw\_circle(self.current\_coord,"white",2)

self.draw\_circle(self.current\_coord,"red",1)

self.current\_coord=self.half\_point(self.current\_coord,random.choice([self.a,self.b,self.c]))

self.draw\_circle(self.current\_coord,"green",2)

print("x1 button pressed")

def x10(self):

for i in range(10):

self.x1()

#time.sleep(0.001)

#self.canvas.update\_idletasks()

def x100(self):

for i in range(10):

self.x10()

def x1000(self):

for i in range(100):

self.x10()

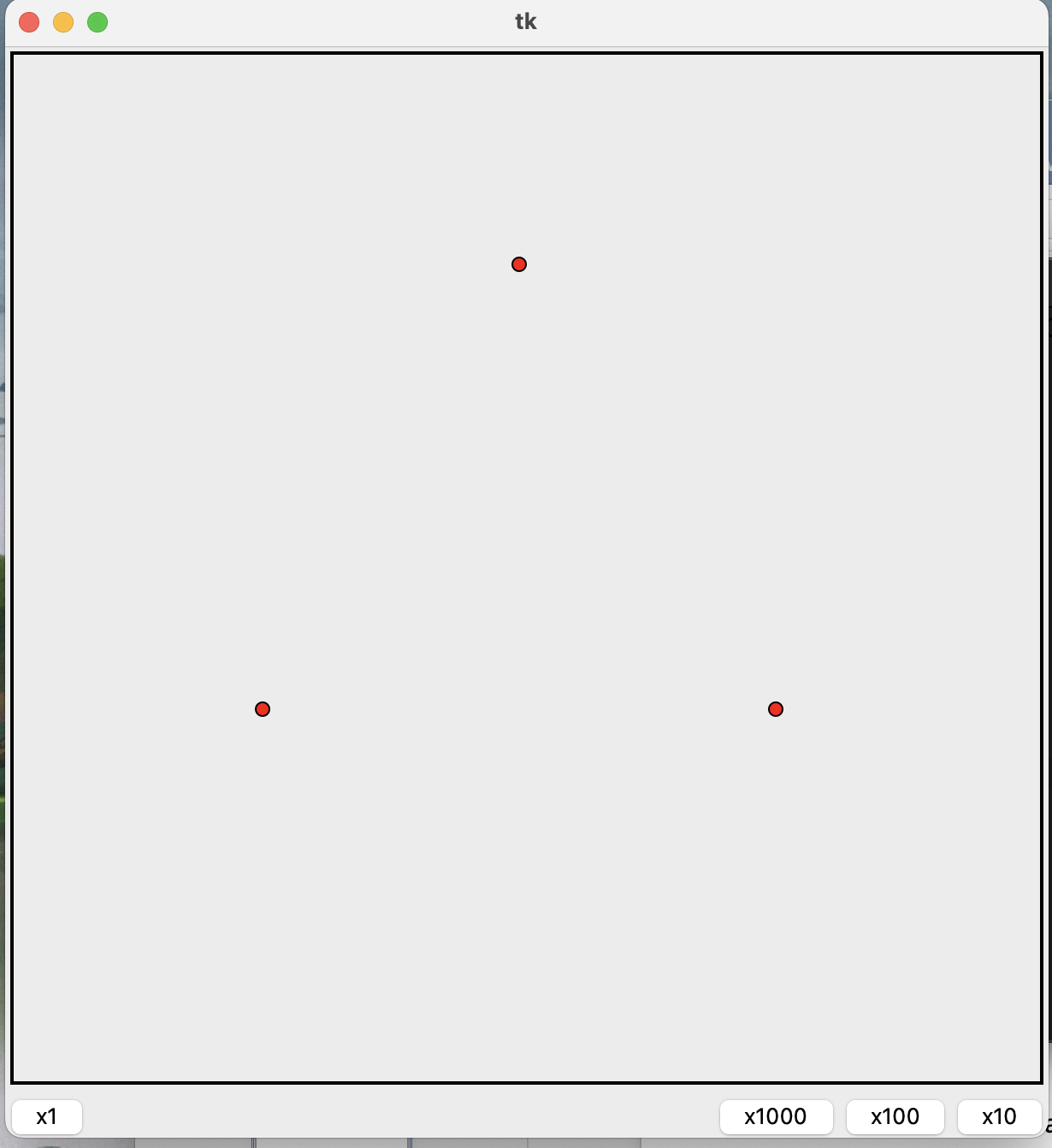
root = tk.Tk()

app = App(root)

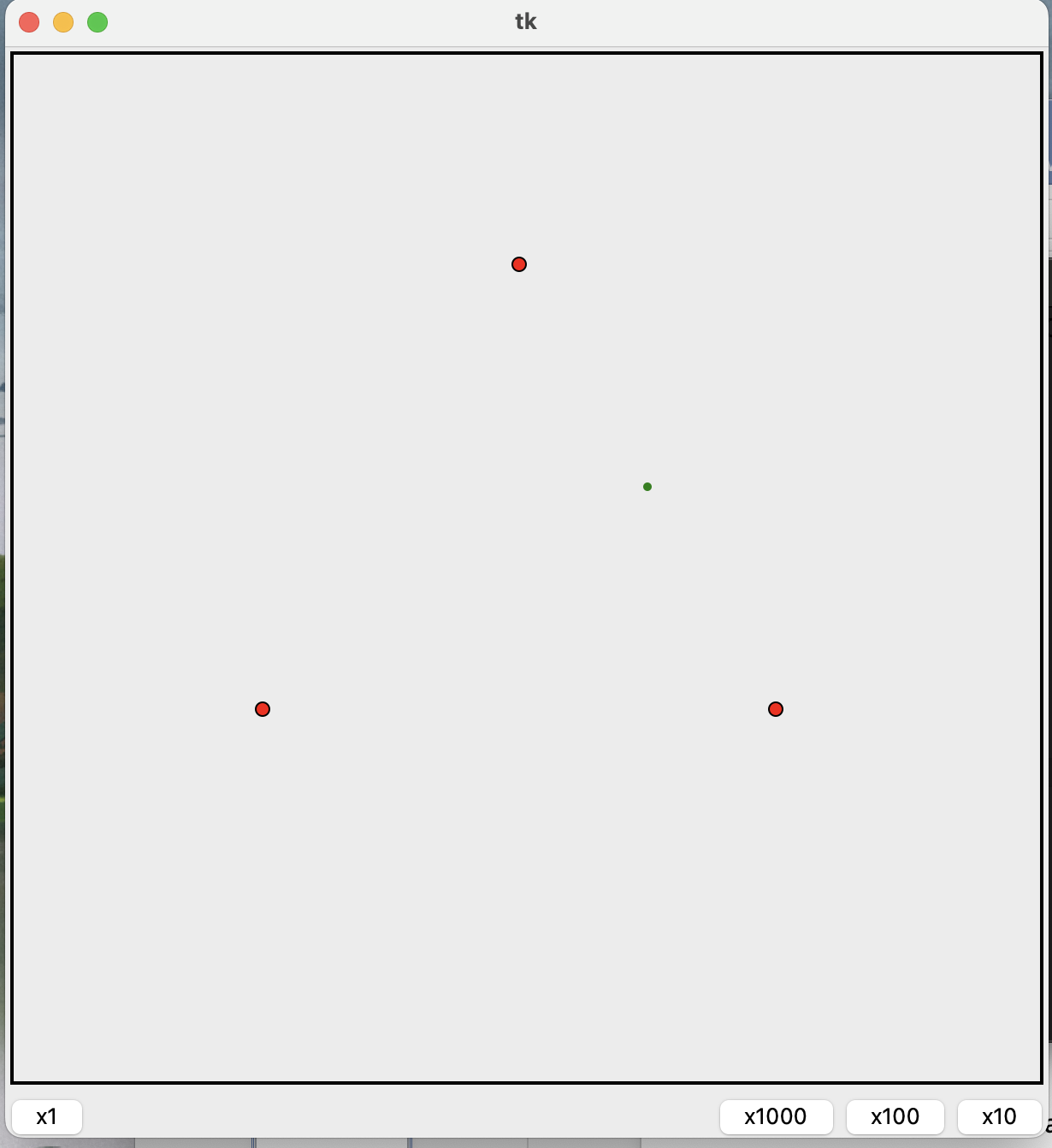
root.mainloop()

**Output:**

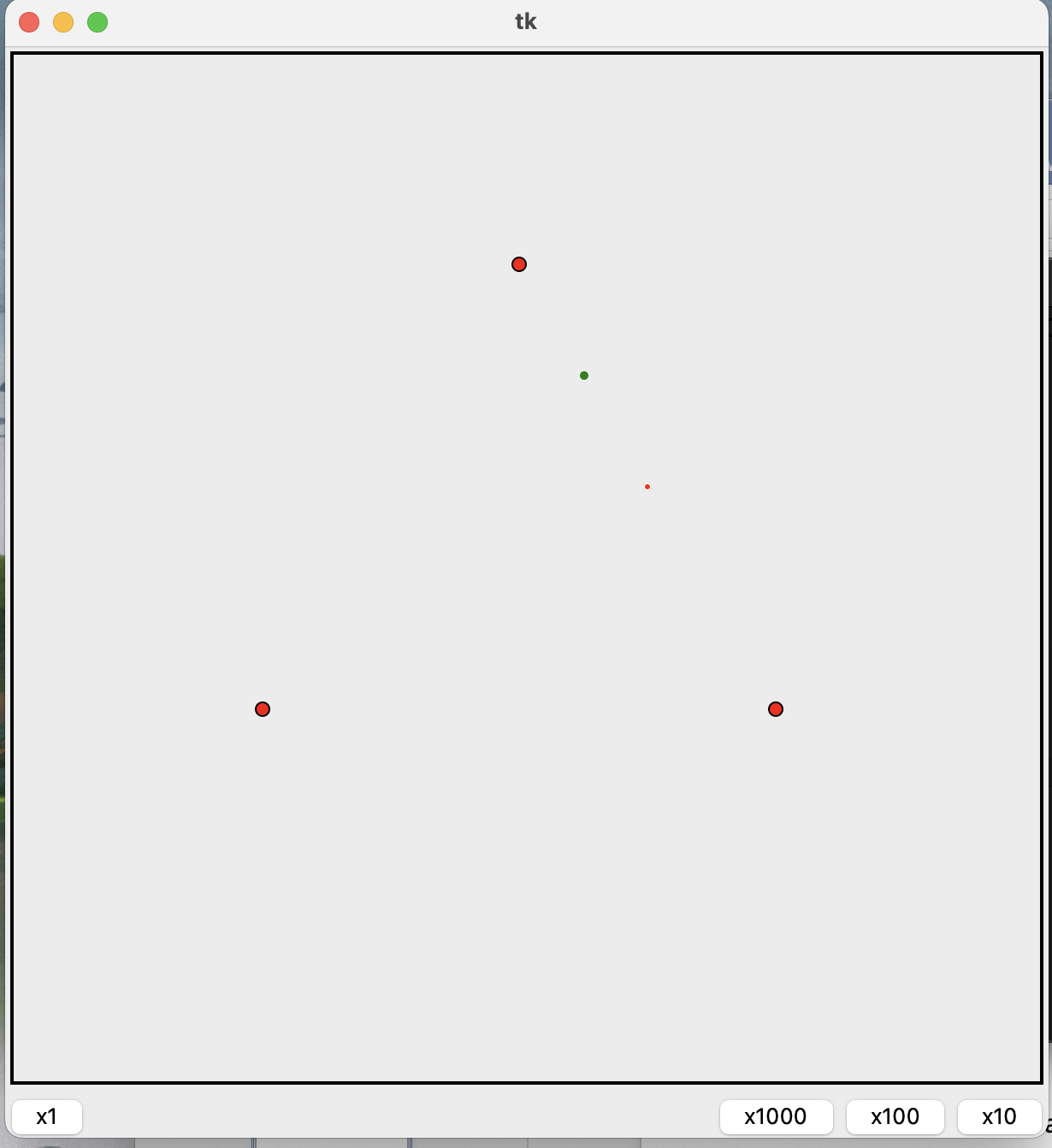
**Initial pattern:**



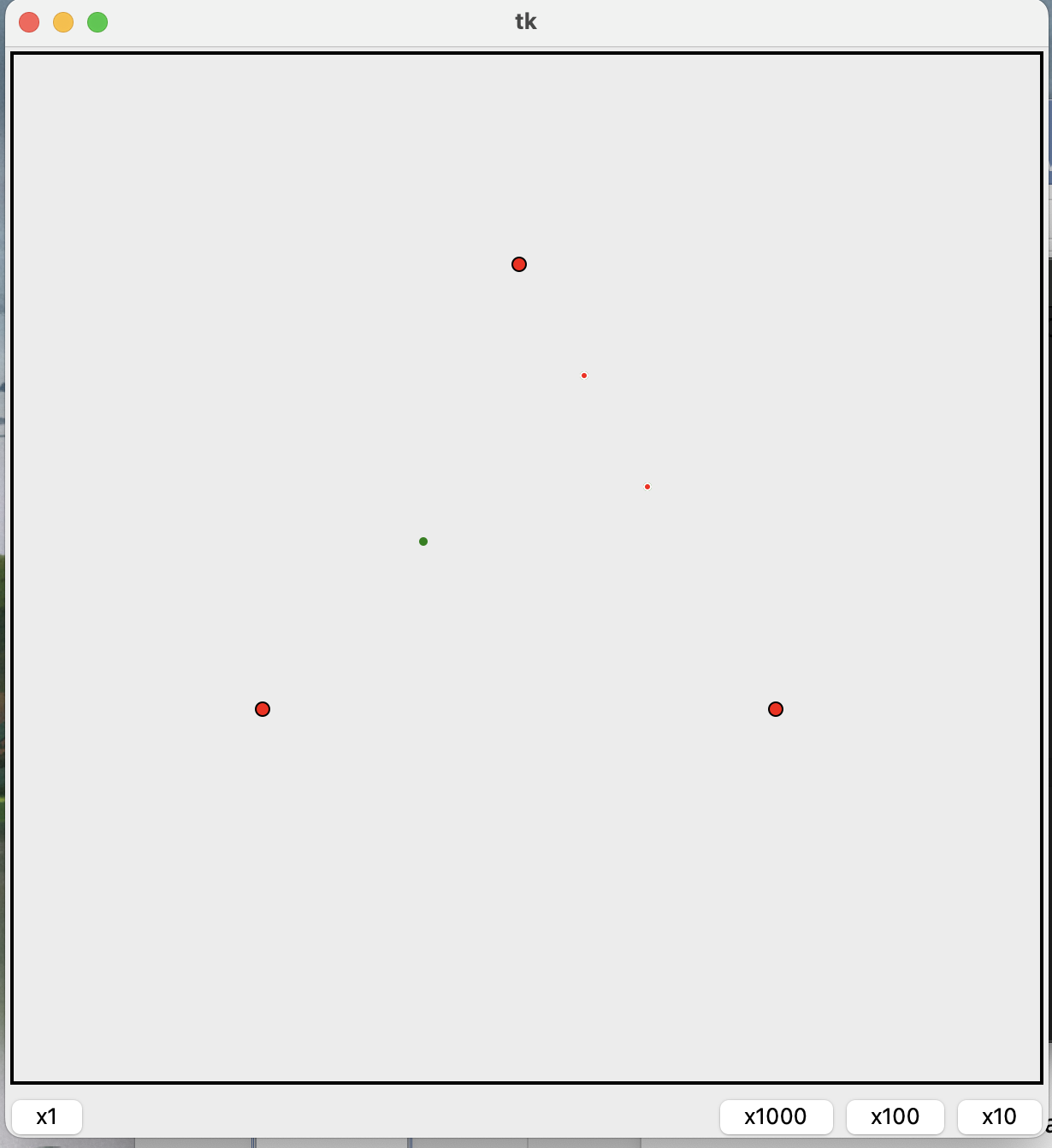
**Pattern after one iteration:**



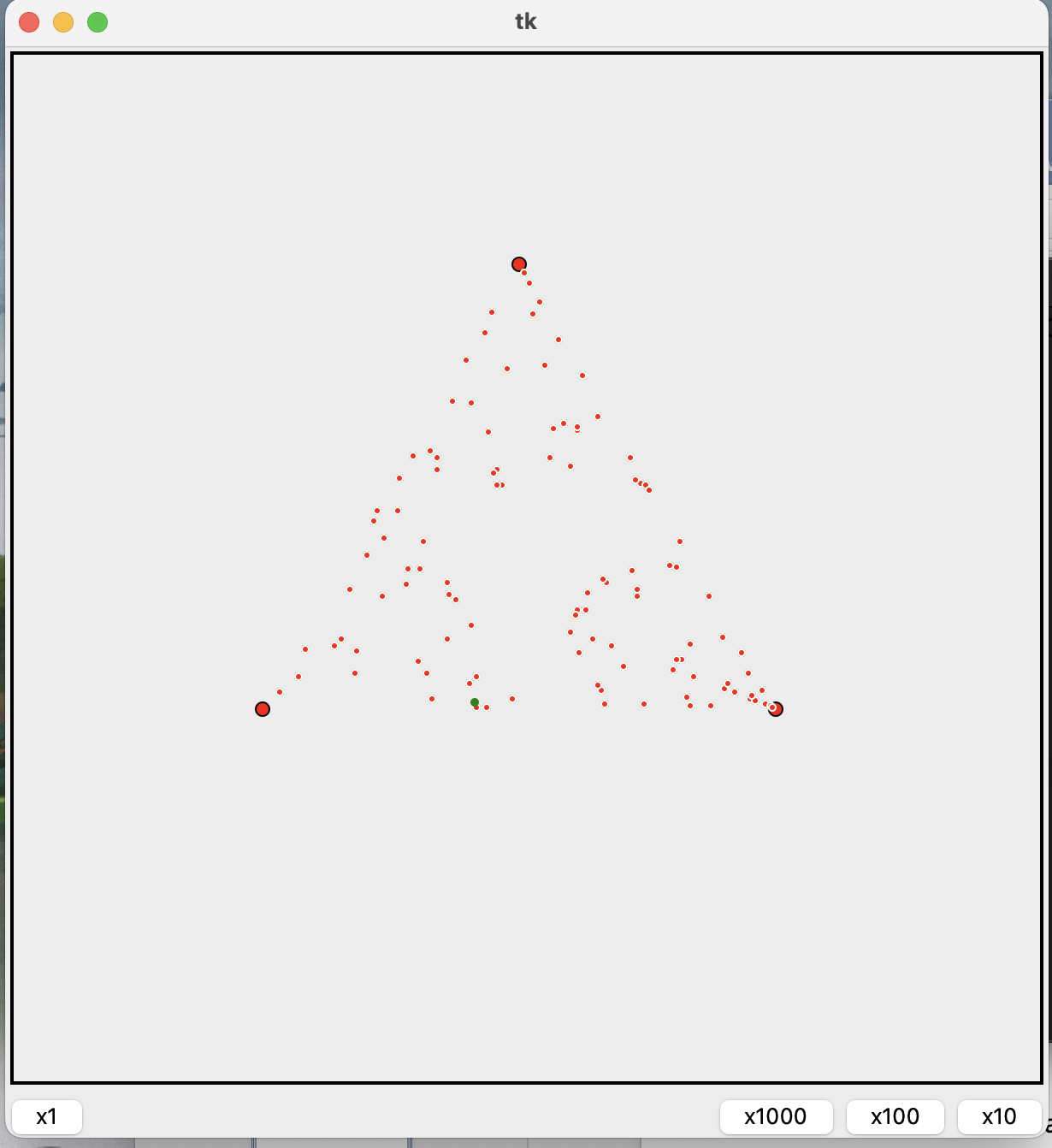
**Pattern after two iterations:**

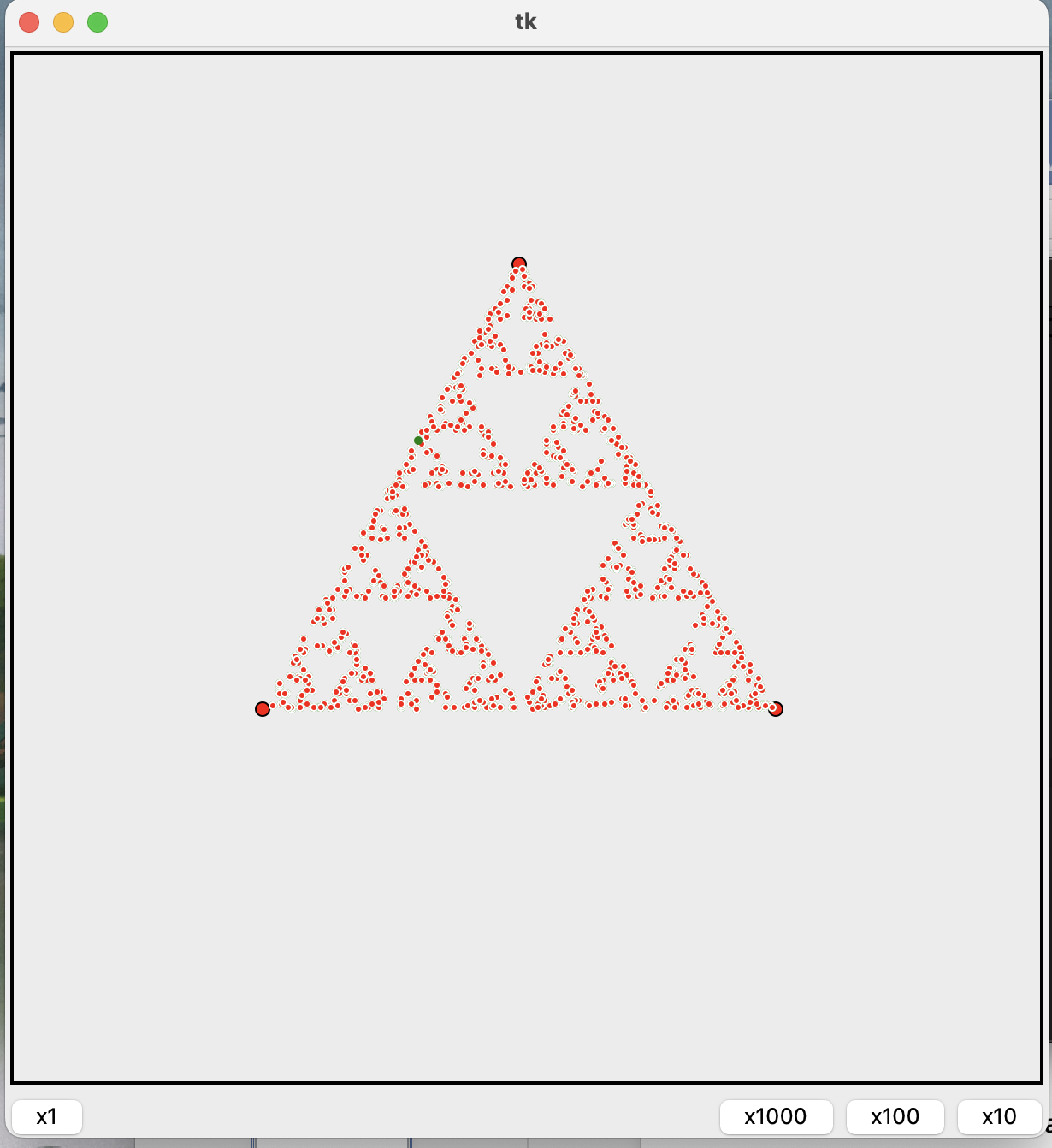


**Pattern after three iterations:**

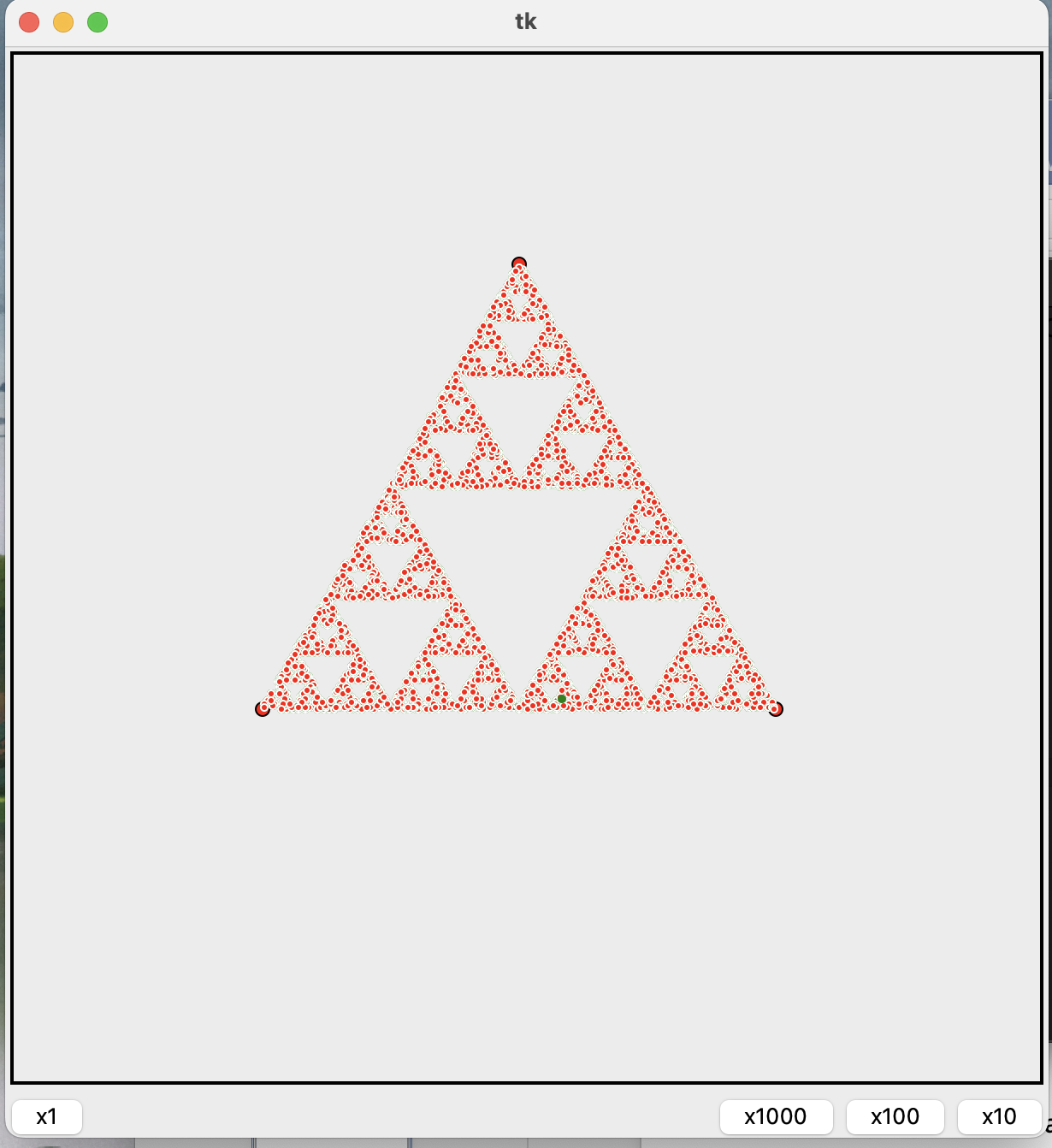


**Pattern after a hundred iterations:**



**Pattern after a thousand iterations:**

**Pattern after a few thousand iterations:**



**Results:**

The result of my project report on the Sierpinski Triangle using Python's Tkinter library is a graphical representation of the Sierpinski Triangle using an equilateral triangle as a starting point. The program uses iteration to generate the triangle, breaking it down into smaller triangles at each iteration until the desired level of detail is reached.

The program also allows the user to control the level of detail through a graphical user interface, adjusting the number of iterations to increase or decrease the complexity of the triangle. The resulting graphical output provides a beautiful and intricate representation of the Sierpinski Triangle, highlighting its self-similarity and fractal nature.

Overall, the project successfully achieved its objective of exploring the Sierpinski Triangle using Python's Tkinter library, providing a comprehensive and engaging visual representation of this fascinating geometric figure. The project also allowed for the development of skills in programming and graphical user interface design, as well as a deeper understanding of the properties and structure of the Sierpinski Triangle.

**Conclusion:**

My project on the Sierpinski Triangle using Python's Tkinter library was an exciting and challenging endeavour that allowed me to explore the beauty and complexity of this geometric figure in a tangible and visual way. Through the project, I was able to develop my skills in programming and graphical user interface design, as well as gain a deeper understanding of the properties and structure of the Sierpinski Triangle.

One of the most interesting aspects of the project was the use of iteration to generate the Sierpinski Triangle. By breaking down the triangle into smaller triangles at each iteration, the program was able to create a detailed and intricate representation of the figure, highlighting its self-similarity and fractal nature. This was a powerful demonstration of the potential of iterative functions in generating complex patterns and structures, and provided a valuable lesson in algorithmic design and efficiency.

In addition to the use of iteration, the project also required a strong understanding of the Tkinter library and its features, including the use of drawing functions and graphical user interface elements. This allowed for the creation of a user-friendly interface that allowed the user to control the level of detail of the Sierpinski Triangle, adjusting the number of iterations to increase or decrease its complexity.

Overall, my project on the Sierpinski Triangle using Python's Tkinter library was a rewarding and engaging experience that allowed me to explore the beauty and complexity of this fascinating geometric figure in a tangible and visual way. It also provided valuable insights into programming and graphical user interface design, and highlighted the potential for these tools to provide engaging and interactive visualizations of mathematical concepts.

**References**

1) Wikipedia

2) Stack Overflow

3) GeeksforGeeks