Frankfurt University of Applied Science

Java Project: Fractal Tree

Tran Quang Minh - 1403567

Ngo Dinh Anh Khoa - 1403428

Java Documentation
Frankfurt University of Applied Science
Bachelor of Science in *Computer Science*

March 13, 2023

Abstract

This is an undergraduate Java project report. In computer science, a fractal tree index is a tree data structure that keeps data sorted and allows searches and sequential access simultaneously as a B-tree but with insertions and deletions that are asymptotically faster than a B-tree. The purpose of the project is to create an application that contains a GUI and a drawing tool for Fractal Tree. In this project, JavaFX and its libraries are used as the main methods to program the entire application. We find that using JavaFX and iteration methods is an effective way to create a Fractal Tree drawing tool and it also produces very positive results.

Keywords: JavaFX, iteration

Report's total word count: we expect a maximum of 10,000 words (excluding reference and appendices) and about 40 - 50 pages.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The public is more likely to be familiar with fractal art than the mathematical concept when it comes to the word "fractal." It can be difficult to define the mathematical concept formally, even for mathematicians, but key features can be understood with a little mathematical knowledge. A lens or other device that zooms in on digital images to reveal finer, previously invisible, new structure is an easy analogy for explaining "self-similarity". On fractals, however, no new details are revealed; nothing changes and the same pattern repeats over and over, or for some fractals, nearly the same pattern appears repeatedly. People have pondered self-similarity informally, for example, in the infinite regress in parallel mirrors or in the homunculus, the little man inside the head... Fractals differ in that the pattern reproduced must be detailed, Wikipedia(2023).
Here are some examples of fractals:

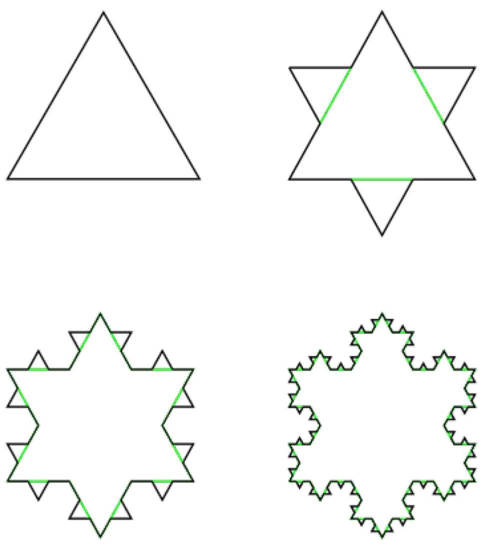

Figure 1.1: Snowflakes made out of a fractal.

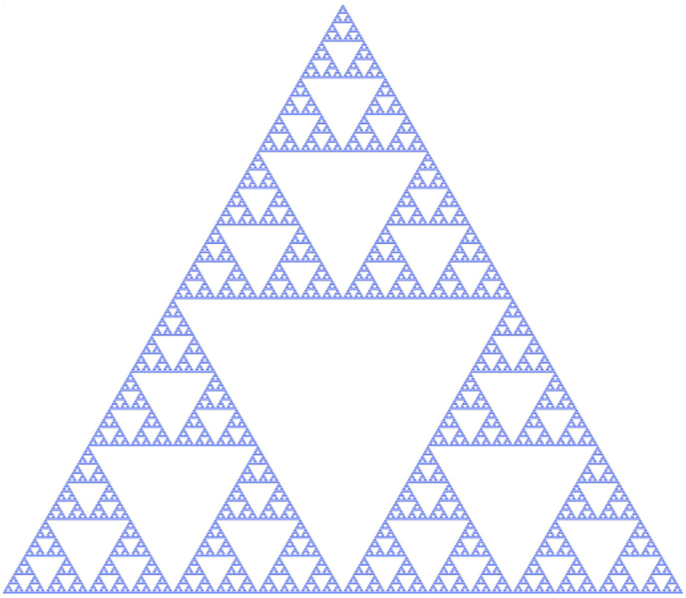

Figure 1.2: Sierpiński triangle.

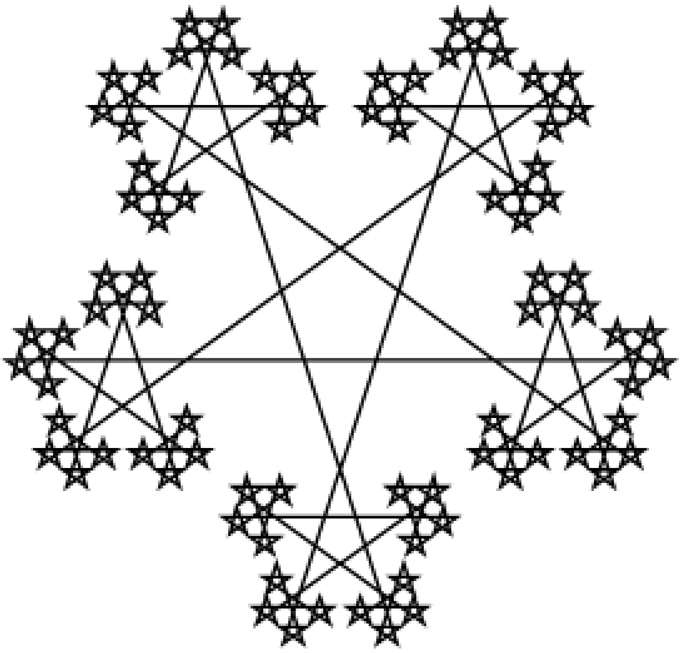

Figure 1.3: Star Fractal.

1.2 Problem statement

In this documentation, we will talk about the Fractal Tree, which is the subset of the fractal. Let's take a look at "Fractal Canopy", which is a type of fractal tree. It is one of the easiest-to-create types of fractals. Each canopy is created by splitting a line segment into two smaller segments at the end (symmetric binary tree) and then splitting the two smaller segments as well, and so on, infinitely (but in our program, we implement the tree by level). Canopies are distinguished by the angle between concurrent adjacent segments and the ratio between lengths of successive segments. The basic fractal tree must follow these rules:

- The angle between any two neighboring line segments is the same throughout the fractal.
- The ratio of lengths of any two consecutive line segments is constant.
- Points all the way to the end of the smallest line segments are interconnected, which is to say the entire figure is a connected graph.

1.3 Aims and objectives

The aim of our group is to create a complete program containing GUI and Fractal Tree drawing tool. The GUI will have buttons connected to the drawing tool. Moreover, the tool will perform drawing fractal trees with many different levels and angles as well as colors.
The objective of the project is to use JavaFX and SceneBuilder in order to create application menus and Fractal Tree drawing tools.

1.4 Solution approach

To create the fractal tree, these rules must be followed:

- Start at some point and move a certain distance in a certain direction.
- At this point makes a branch. Turn some angle to the right and then repeat the previous step with the distance being decreased.
- Go back and do the same thing for the left side.

A fractal tree is known as a tree that can be created by recursively symmetrical branching. In practice, there are a lot of cases that we have to use recursive functions to solve. Therefore, via fractal tree problems, it is very useful to solve similar recursive problems. Moreover, fractal trees are also related to the space-filling curve problem.

1.5 Organization of the report

This report is organized into eight chapters. Chapter 1 is the introduction of the entire project which will give a glance at the content of the project. Chapter 2 summarizes the contributions and achievements of each member of the project. Chapter 3 is about the problem description which will give information about the definitions and examples, etc. Chapter 4 will point out the related work which contains the related algorithm and applications. Chapter 5 details the proposed approaches such as input/output format, benchmarks, and algorithm pseudocode. Chapter 6 will give information on the implementation details namely application structure, GUI details, UML diagram, used libraries, and code snippets. Chapter 7 is the part of the experimental results and statistical Tests which contains simulations, uses benchmarks, tables, charts, and evaluation. The final chapter - chapter 8 is about the conclusions and future works.

Chapter 2

Summary of contributions and achievements

2.1 Team Contributions

For drawing trees, Ngo Dinh Anh Khoa has created the GUI for each tree, those GUI are very basic but the functionality is acceptable. Also, Khoa created and managed the Controller for each GUI of the tree, and the Node.java file to work with the tree branch (leaf).
For creating the starting GUI and its effect, Tran Quang Minh managed TreeApp.java and GameViewManager.java files. Minh's job is to link all GUIs together and create the transition between those GUIs.

2.2 Work Structure

Both of us have a good understanding of others' ideas, and ways to solve problems. That is the reason why our communication is favorable. For the repository, both of us agree to use GitHub for the project, and it turns out to be very effective and convenient for us to make adjustments to our project. Besides Github, we often communicate together and exchange ideas directly, or via Facebook and email. Using an application made for GitHub called GithubDesktop, we could easily push and pull code from each other.

Here is our GitHub page: `https://github.com/MinhTran1506/Game.git`

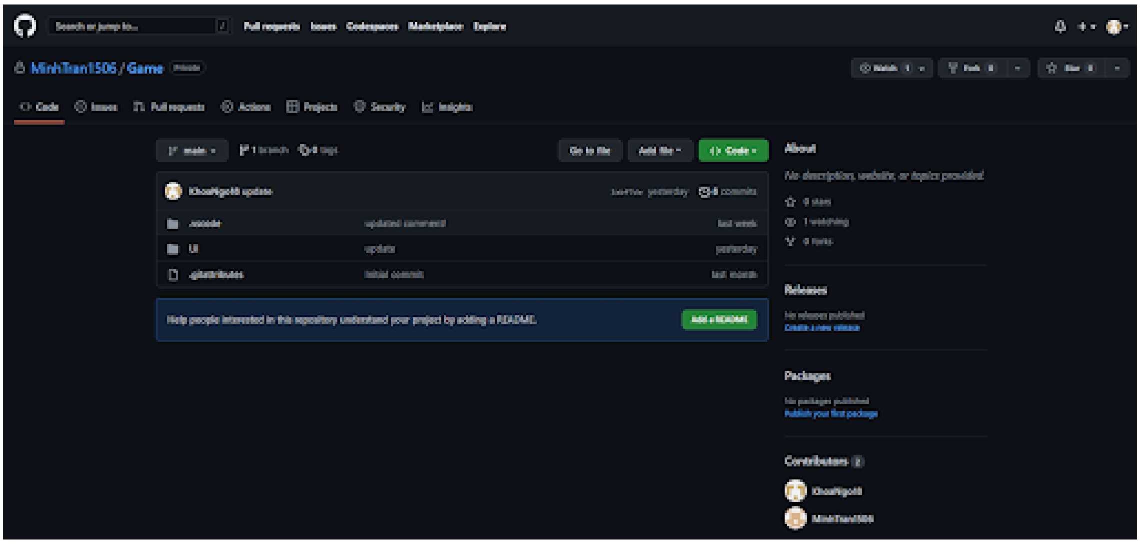

Figure 2.1: GitHub Website.

2.3 Project Idea

For the decision of the project, we consulted ideas from several sources such as Youtube (BroCode, ...), Reddits, Stackoverflow, and Github,... And finally, we agree with each other that Ngo Dinh Anh Khoa will do the implementation of trees, and Tran Quang Minh will do the UI of the application and link the GUIs together. //The first prototype contained a lot of things we wanted to do. After a few weeks, we have to reduce the amount of stuff in the project since there is not enough time to implement everything. Here is our first prototype

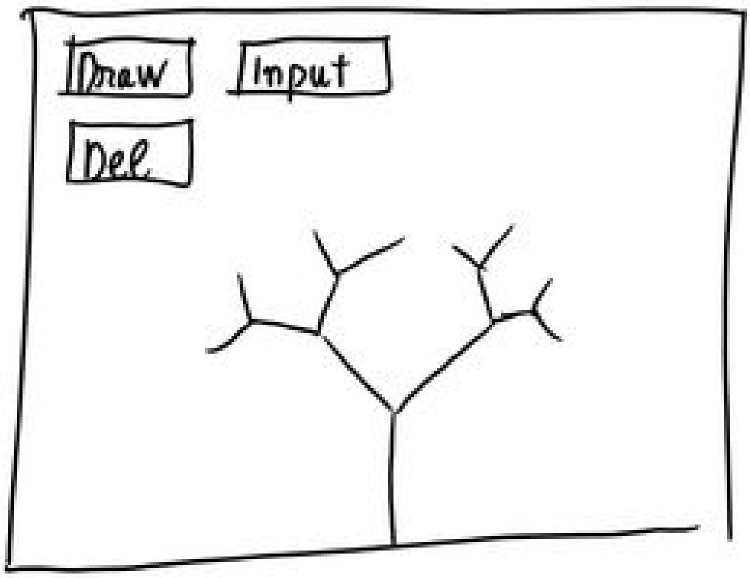

Figure 2.2: Prototype.

Chapter 3

Problem Description

3.1 Definition of Fractal Tree

If we want to understand the “Fractal Tree”. Firstly, we need to understand what Fractal is. Definition of **Fractal**: In mathematics, a fractal is a geometric shape containing detailed structure at arbitrarily small scales, usually having a fractal dimension strictly exceeding the topological dimension. Many fractals appear similar at various scales, as illustrated in successive magnifications of the Mandelbrot set. This exhibition of similar patterns at increasingly smaller scales is called self-similarity, also known as expanding symmetry or unfolding symmetry; if this replication is the same at every scale, as in the Menger sponge, the shape is called affine self-similar. Fractal geometry lies within the mathematical branch of measure theory, Wikipedia (2022).

3.2 Recursion Method

A “Fractal Tree” is a subcategory of Fractal. A basic Fractal Tree is based on self-similarity. This means the next branch of the tree follows the same pattern as the parent branch (root). The angle of the branch always stays the same, and the length and the thickness of the branch are decreased by a factor.
The first approach for drawing the tree is to use recursion. Here is the algorithm for drawing the tree using recursion:

```
Function drawingTreeRes(int level){
If (level != 0){
  draw(root);
  create(left, root);
  create(right,root);
  drawingTreeRes(root.left,level--);
  drawingTreeRes(root.right,level--);
    }
}
```

Listing 3.1: Recursion Method Pseudo-code.

There is another way, that is using iteration. The idea is still the same just like recursion, here is the algorithm for iteration:

```
Function drawingTreeIteration(){
    for (int i = branchList.size() - 1; i >=0; i-){
      create(leftBranch, branchList.get(i));
      create(rightBranch, branchList.get(i));
      branchList.add(leftBranch);
      branchList.add(rightBranch);
    }
}
```

Listing 3.2: Recursion Method Pseudo-code 2.

In our project, we choose to use iteration, and for the branch, we create a new data type called Node. The detail for each file will be discussed later in Chapter 6.
Some examples of the fractal tree:

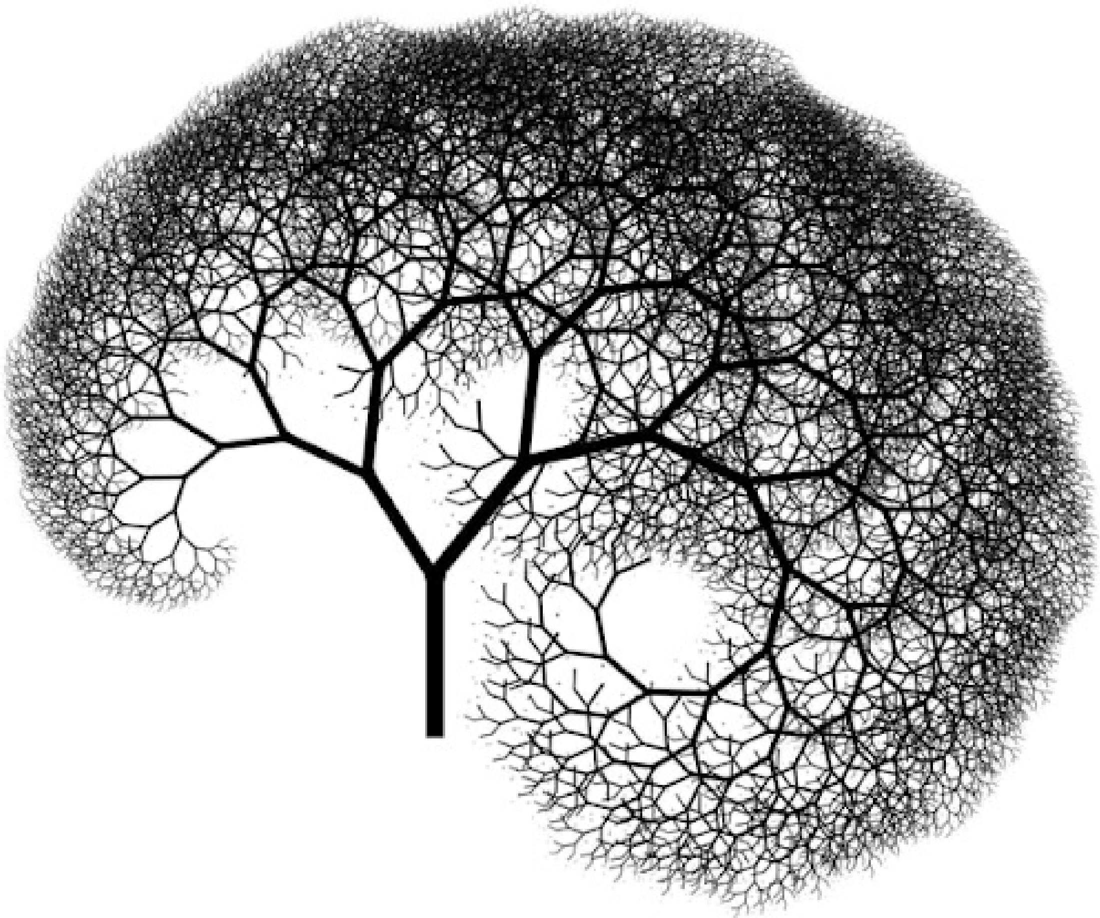

Figure 3.1: Fractal Tree Example 1.

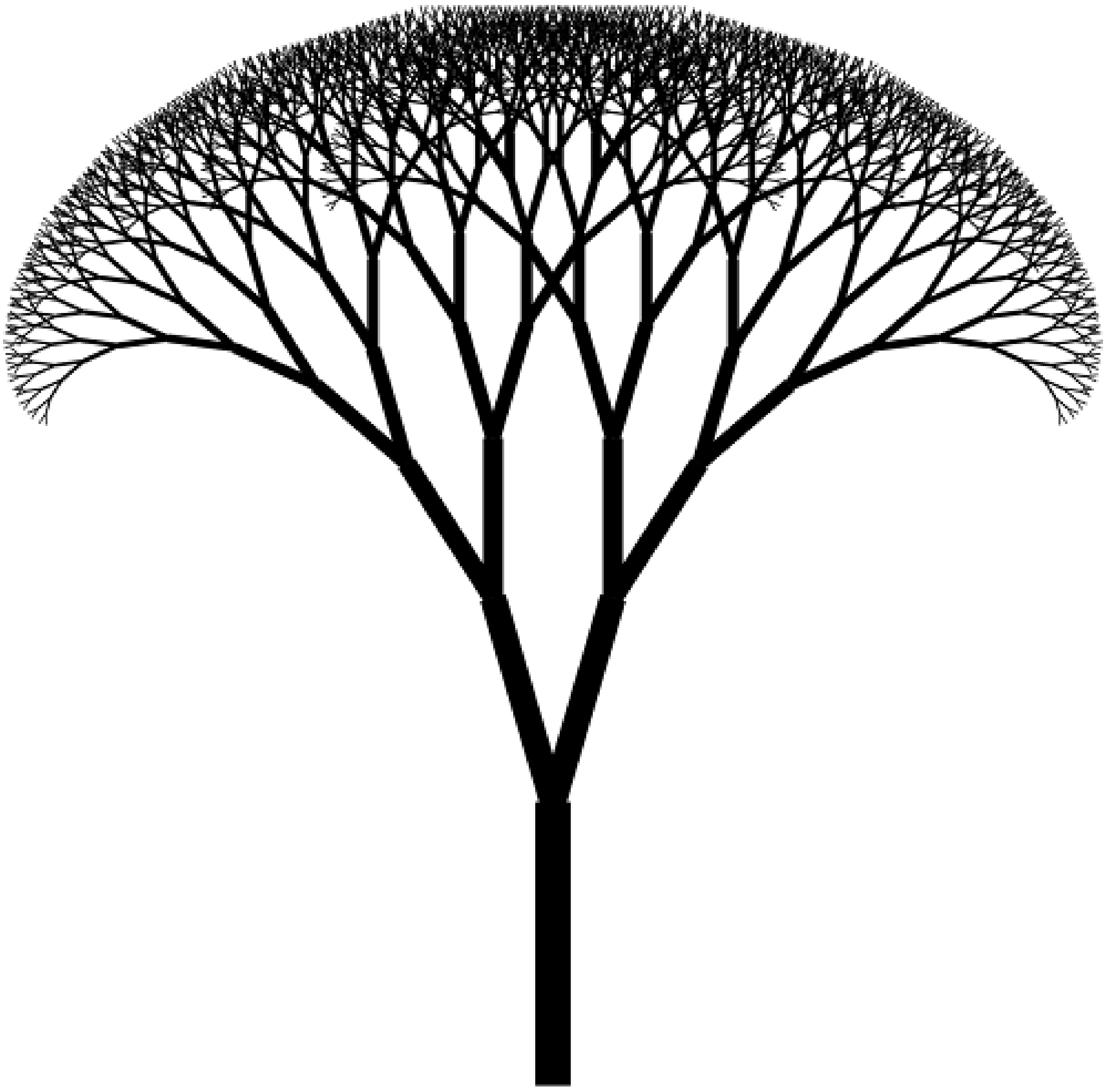

Figure 3.2: Fractal Tree Example 2.

3.3 Rotation Problem

To rotate the branches of the tree, we also need to know about the rotation in coordination. Here is the formal definition of rotation: “In mathematics, a rotation of axes in two dimensions is a mapping from a $xy-Cartesian$ coordinate system to an $x'y'-Cartesian$ coordinate system in which the origin is kept fixed and the x' and y' axes are obtained by rotating the x and y axes counterclockwise through an angle θ. A point P has coordinates (x,y) with respect to the original system and coordinates (x',y') with respect to the new system. In the new coordinate system, point P will appear to have been rotated in the opposite direction, that is, clockwise through the angle. A rotation of axes in more than two dimensions is defined similarly. A rotation of axes is a linear map and a rigid transformation.

The equations defining the transformation in two dimensions, which rotate the xy axes

counterclockwise through an angle "theta" into the $x'y'$ axes, are derived as follows.
In the xy system, let the point P have polar coordinates (r, α). Then, in the $x'y'$ system, P will have polar coordinates $(r, \alpha - \theta)$.
To rotate in the trigonometric formula:

$$x = r\cos\alpha \ (1)$$
$$y = r\sin\alpha \ (2)$$

And using the standard trigonometric formulae for differences, we have:

$$x' = r\cos(\alpha - \theta) = r\cos\alpha\cos\theta + r\sin\alpha\sin\theta \ (3)$$
$$y' = r\sin(\alpha - \theta) = r\sin\alpha\cos\theta - r\cos\alpha\sin\theta \ (4)$$

Substituting equations (1) and (2) into equations (3) and (4), we obtain:

$$x' = x\cos\theta + y\sin\theta \ (5)$$
$$y' = -x\sin\theta + y\cos\theta \ (6)$$

Equations (5) and (6) can be represented in matrix form as: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$
Given a point, which has a base (x, y). Here are some special rotations:

- 90^0 clockwise rotation: (x, y) becomes $(y, -x)$
- 90^0 counterclockwise rotation: (x, y) becomes $(-y, x)$
- 180^0 clockwise and counterclockwise rotation: (x, y) becomes $(-x, -y)$
- 270^0 clockwise rotation: (x, y) becomes $(-y, x)$
- 270^0 counterclockwise rotation: (x, y) becomes $(y, -x)$

But since the JavaFX library already includes the rotate class, we do not need to use the above formula to find the corresponding coordinate. Here is the website for all of the things you need to know about the Rotate in JavaFX library:
`https://docs.oracle.com/javase/8/javafx/api/javafx/scene/transform/Rotate.html`

Chapter 4

Related Work

4.1 Algorithm

According to *Generating a fractal tree, Jingyi Gao (2020)*, there is a way to draw a fractal tree using mathematics. In the report, he explains the definition of “Similarity dimension”, “Fractal dimension and scaling relationships”, “Da Vinci’s rule and its analog for two-dimensional tree”, “Murray’s law”, “Scaling exponent in the conservation law and exponent in the scaling relationship”, and provides the reader about the principles and algorithm to draw a fractal tree. Here we mostly focus on the algorithm of “JINGYI GAO”.

The algorithm of *Jingyi Gao* in *Generating a fractal tree*:

Step 1: Create a recursive function with parameter x coordinate, y coordinate, and branch length l.

Step 2: Check whether the length of the branch is less than 1/9. If it’s true, break.

Step 3: Create a path object that draws a ”v” with each branch of length 1/2 of the input length.

Step 4: Call the function on the corresponding x coordinate and y coordinate of the left endpoint of the ”v” just drawn.

Step 5: Call the function on the corresponding x coordinate and y coordinate of the right endpoint of the ”v” just drawn.

Jingyi Gao algorithm used recursion to draw the tree, but we used the iterative method. Both methods work in the same way, but we find that iteration is easier to manage in our project. At first, we used recursion and did not store the branches, so we can not do the animation part of the tree. Next, we changed to use iteration to store the branches so that the animation part work perfectly. However, looking back to the recursion, it still can store the branches.

At first, we used recursion and did not store the branches, so we can not do the animation part of the tree. Next, we changed to use iteration to store the branches so that the animation part work perfectly. However, looking back to the recursion, it still can store the branches.

4.2 Application

It is a tree data structure that makes data sorted and allows searches and sequential access at the same time as B-Tree, but with insertions and deletions that are asymmetrically faster than B-Tree. In fractal trees, a node can have more than two children, like a B-tree. Fractal tree indexes also feature intermediate buffers at each node, unlike a B-tree, which allows insertions, deletions, and other changes to be stored. The goal of the buffers is to schedule disk writes so that each write performs a large amount of useful work, thereby avoiding the worst-case performance of B-trees, in which each disk write may change a small amount of data on the disk, Wikipedia (2014).
According to Tutorials Point(n.d.), fractals can be used for:

- Astronomy: For analyzing galaxies, rings of Saturn, etc.
- Biology/Chemistry: For depicting bacteria cultures, Chemical reactions, human anatomy, molecules, plants, etc.
- Others: For depicting clouds, coastline, and borderlines, data compression, diffusion, economy, fractal art, fractal music, landscapes, special effect, etc.

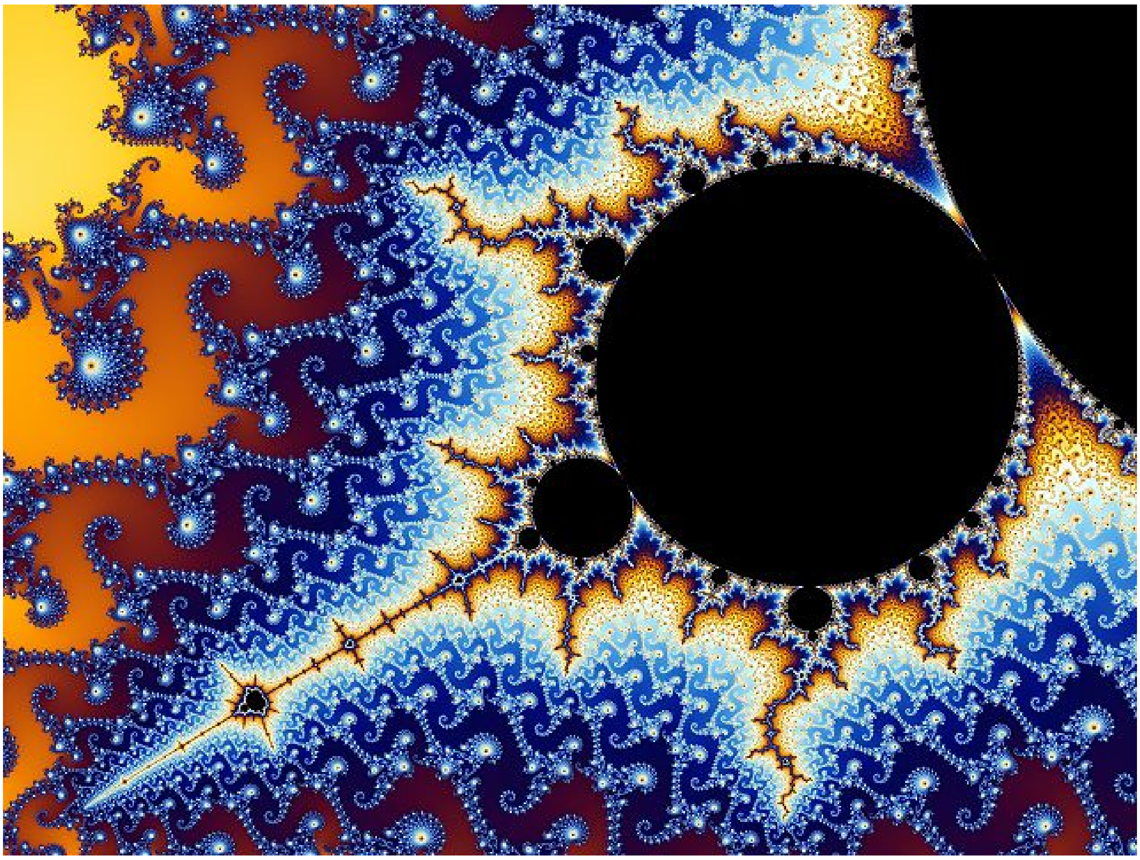

Figure 4.1: Fractal used in computer graphics.

Chapter 5

Proposed Approaches

5.1 Menu user interface

Our first approach is just to create a GUI for a single tree only. After creating the first tree, which is the symmetric tree, there is still a lot of time for us to work on this project, so we agree that we will create other GUIs for other trees and a Starting GUI (Menu GUI) to link all the GUIs of trees together. Here is the Starting GUI (Menu GUI):

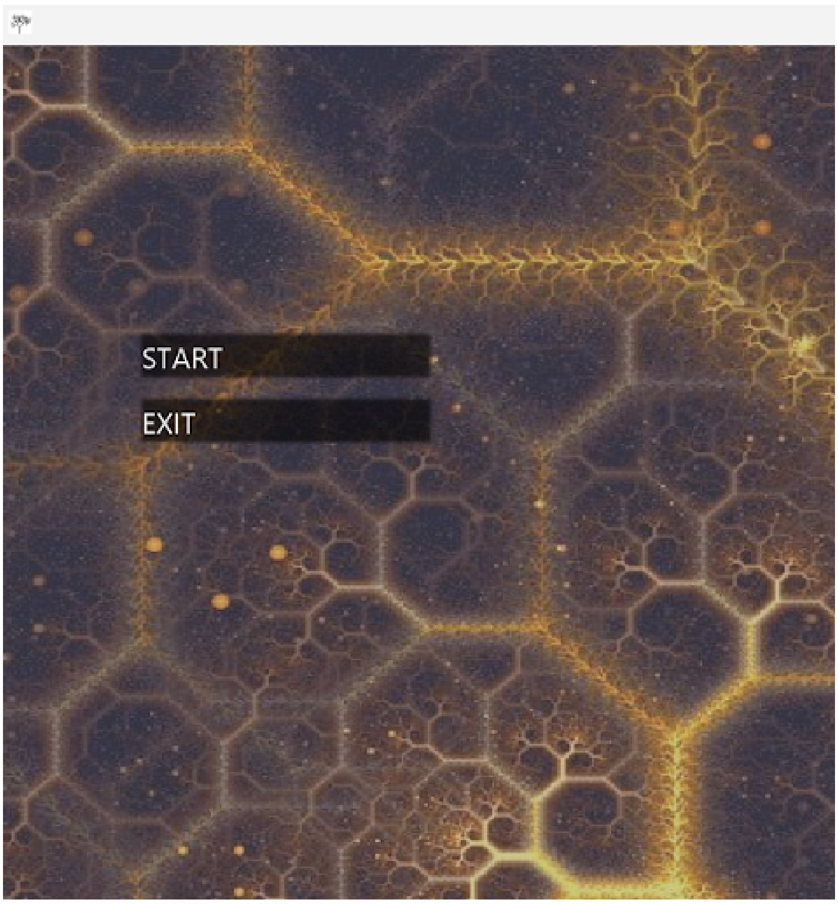

Figure 5.1: GUI of The Application.

In the Starting GUI (Menu GUI), we have only 2 buttons: Start and Exit. If the Start button is selected, we will be able to choose one of the three trees to start the application. If the Exit button is selected, the application will close (end).

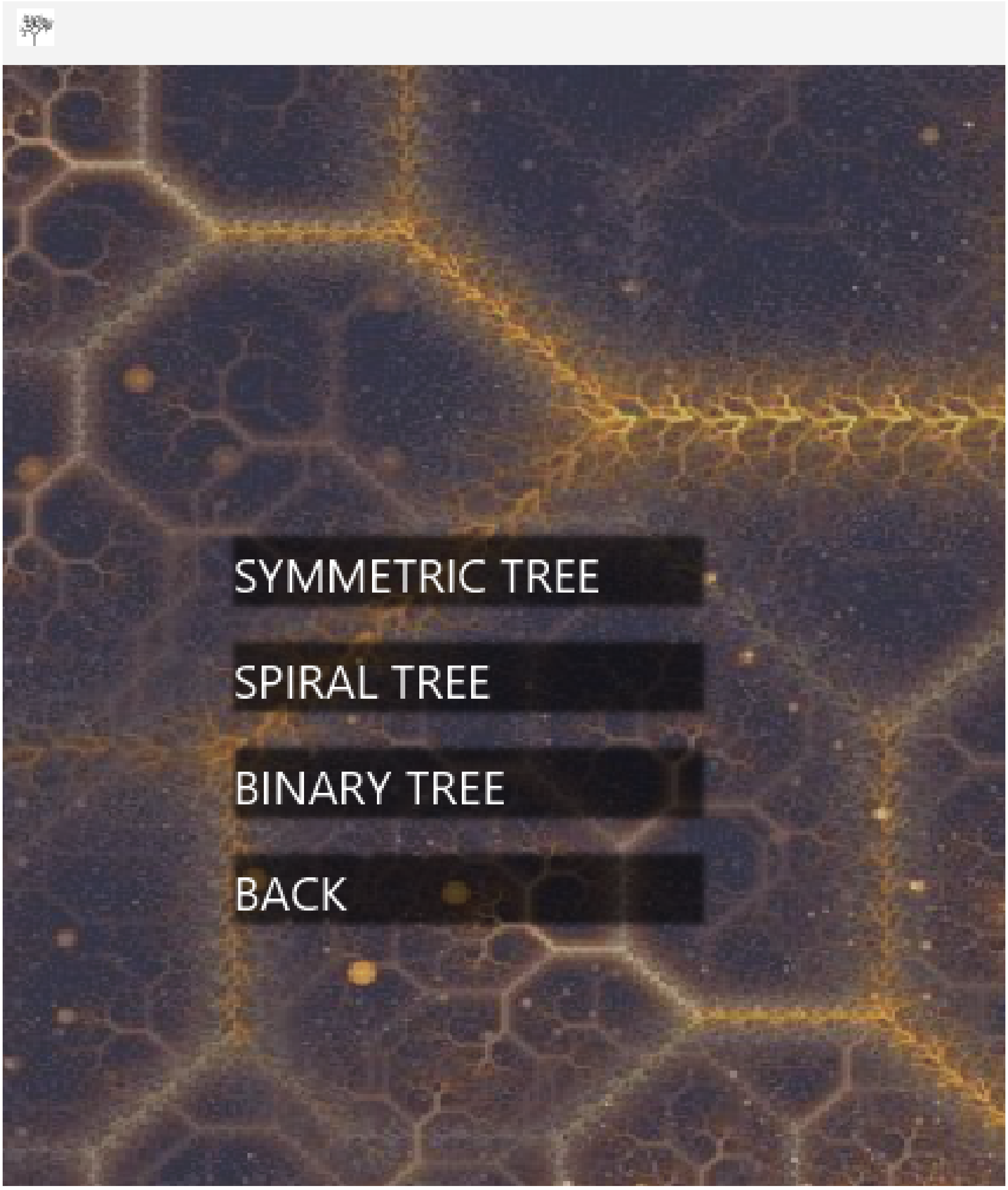

Figure 5.2: Starting GUI (Menu GUI) after the Start button is selected.

After selecting the Start button, the user will be able to choose to draw a “Symmetric Tree”, “Spiral Tree” or “Binary Tree”. The name of the button said it all. With the “Symmetric Tree” button, the user will go to the GUI which draws the symmetric fractal tree. With the “Spiral Tree” button, the user will go to the GUI which draws the spiral fractal tree. With the “Binary Tree” button, the user will go to the GUI which draws the