Fractal Generation Application

Ohr Benshlomo

July 2020

1 Installation

To use the software, first install Blender to the machine:

```
sudo apt install blender
```

Also make sure the pi has ImageTk installed:

```
sudo apt-get install python3-pil python3-pil.imagetk
```

Once installed, unpack the project's tar file to the desired directory. Don't remove any files from the project folder, if anything is removed the program will not work properly. Don't move or delete the files in the BlendFile folder called CubeScene or TreeScene, as these are used to render the final product.

2 External Hardware

None needed.

3 Project Description

The goal of this project was to use Blender's python integration to make a program that would procedurally generate a three dimensional structure based on a "fractal" algorithm.

There are three different algorithms the program uses to generate the fractals, falling into two types based on their initial primitive. The first type is the Cube Fractal. Here is a brief step by step description of how the Cube Fractal algorithm generates the fractal:

1. Create a Cube Primitive, centered at the origin.
2. Randomly select several vertices of the Cube. The ratio of vertices selected to total vertices is given by the user.
3. Duplicate and center a cube at each vertex, then rotate and scale them by a user specified amount.
4. Either add the child cube to the parent to create a larger structure, or subtract the intersection between of the child from the parent to create a smaller structure.
5. If set to 2 iterations, repeat steps 2, 3, and 4 one more time with the result.

The second type is the Tree Fractal, which I won't explain here fully as it is quite an involved process. The simplified step by step description is as follows:

1. Create a 2 dimensional circle primitive at the origin.
2. Duplicate "twice" (actually four times) and move the two children to two different positions in reference to the first. Then rotate and scale those duplicates by a user specified amount (slightly more complicated than this).
3. Bridge the moved and scaled children with the parent by a sort of "tube".
4. Go up to the children's "layer" and repeat steps 2 and 3 with each circle in that layer. Continue iterating up the layers a user specified number of times.

The GUI allows a user to set the parameters of this generation and choose which algorithm to use. The program also allows the user to specify parameters for two different trees and combine them into a tree who's parameters gradually transition from one tree on one side to the other on the other side.

After generating the fractals, the program saves a .blend file of the fractal into the /BlendFiles folder and a render of the file into the /Renders folder. The program displays the final render in a popup window and allows the user to generate another fractal.

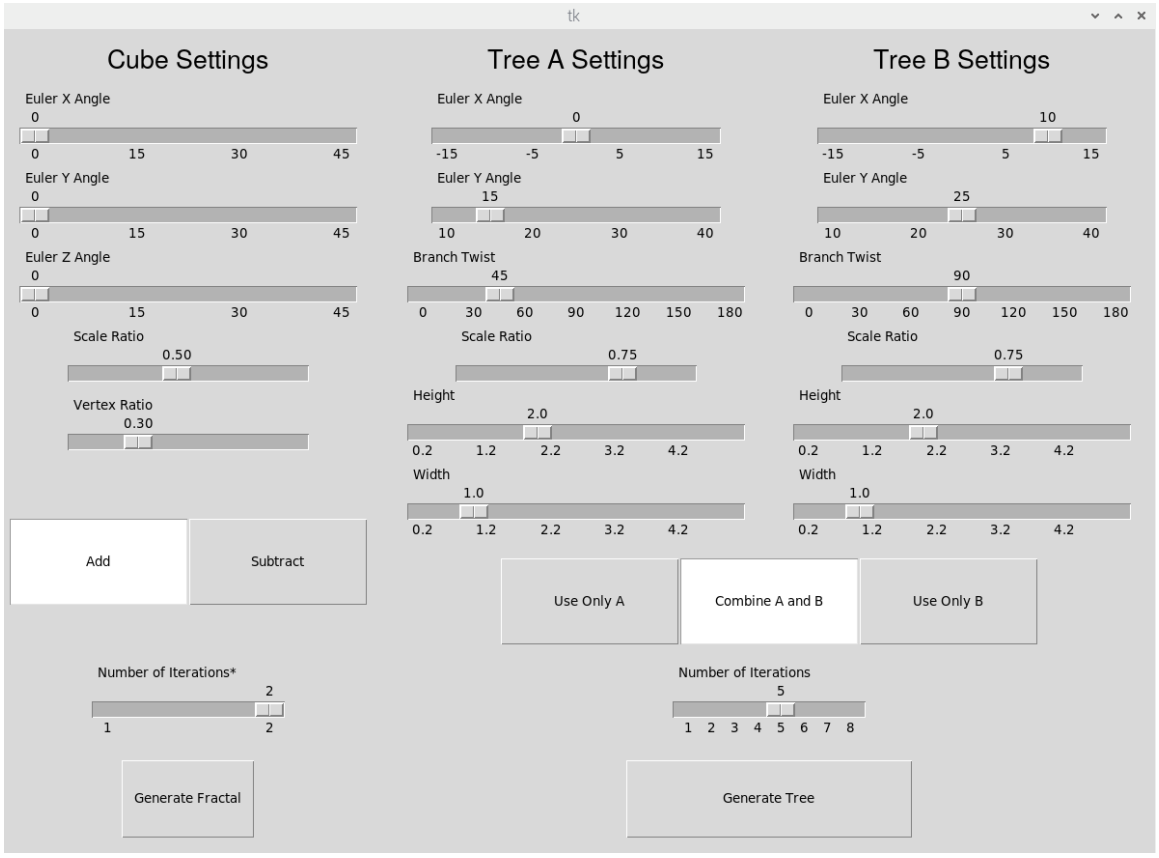


Figure 1: The default state of the GUI on opening

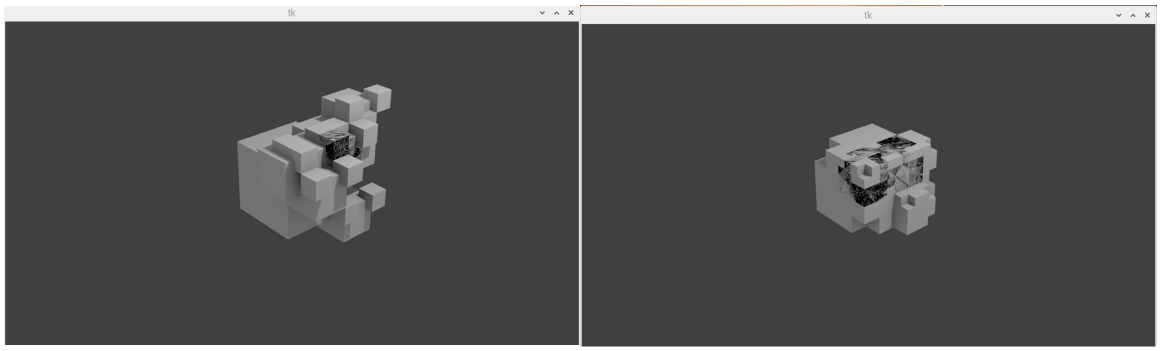
4 Usage Instructions and Examples

To run the program, simply run the FractalGenerator.py script. The GUI on opening should look like Figure 1.

4.1 Generating a Cube Fractal

The leftmost column is specifically for generating a cube fractal. Figure 2 shows two generated examples of what kind of output the cube fractal creates. The addition generated cube used the default settings in the program. The subtraction generated cube used a vertex ratio of 0.55 instead, leaving the rest of the defaults unaltered. Subtraction generated cubes usually look better with higher vertex ratio values. The rotation angles can occasionally generate interesting looking structures, but because of how messy Blender's Union and Difference Boolean modifiers are it tends to just make a mess. Subtraction generated cubes in particular become messy when processed with rotation angles other than zero.

Also do note that, due to the random nature of vertex selection, sometimes the structures will be unremarkable so I recommend generating multiple and see what you get.



(a) A cube generated with addition.

(b) A cube generated with subtraction.

Figure 2: Two generated Cube fractals

Each generation + render should only take about 30 seconds or less.

4.2 Generating a Tree Fractal

Now for the more spectacular looking structure, the tree fractal. The y angle determines the angle between the branches and their common vertical axis. The x angle rotates them perpendicular to the y angle. The twist determines the angle the axis is rotated each iteration. Scale ratio is the area ratio of the narrow end to the fat end, height is how long each branch is, and width is how fat each branch is at the base. For iterations, setting to 8 makes the most impressive looking results, but this could take a few minutes to generate. If the user selects only A or only B, the generator will only use the parameters of the selected column to generate the tree. Combine A and B tells the generator to take a weighted average of the two values, with the weights shifting from all A to all B from one side of the tree to the other.

To generate the tree shown in Figure 3, leave all settings default, except change Euler Y Angle in the B column to 40 degrees and change the Number of Iterations to 8. This will take a few minutes to generate.

4.3 Viewing Results in Blender

All render images shown in the GUI are also saved in ./Renders. The name is either CubeUnion*date-time*.png if an added cube, CubeDiff*date-time*.png if a subtracted cube, and Tree*date-time*.png if a tree.

The .blend files in the ./BlendFiles directory use the same naming scheme. These .blend files can be opened in Blender, where the user can view, alter, and render the fractal manually, or export it to another program/filetype.

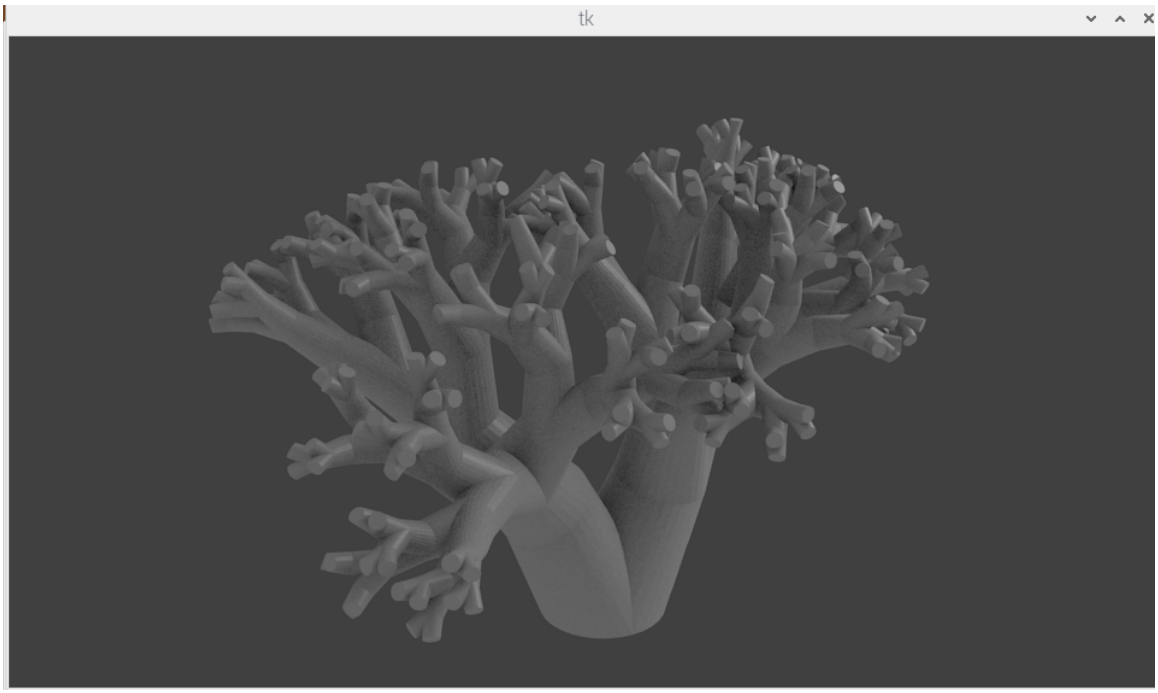


Figure 3: A tree generated with 8 iterations.