# Ps2: Pythagoras Tree

## Assignment Description:

For this assignment we were tasked to create a recursive program that generates a fractal called a Pythagoras tree. This fractal starts with a square then off the top of that square 2 smaller squares angled 45 degrees and -45 degrees from the base square's rotation. These smaller squares then serve as the base for more squares and so on. The program we were tasked to write had to have a recursive function that performed the following algorithm. We also received extra points for making the program generate trees with different angles, for animation, and for color.

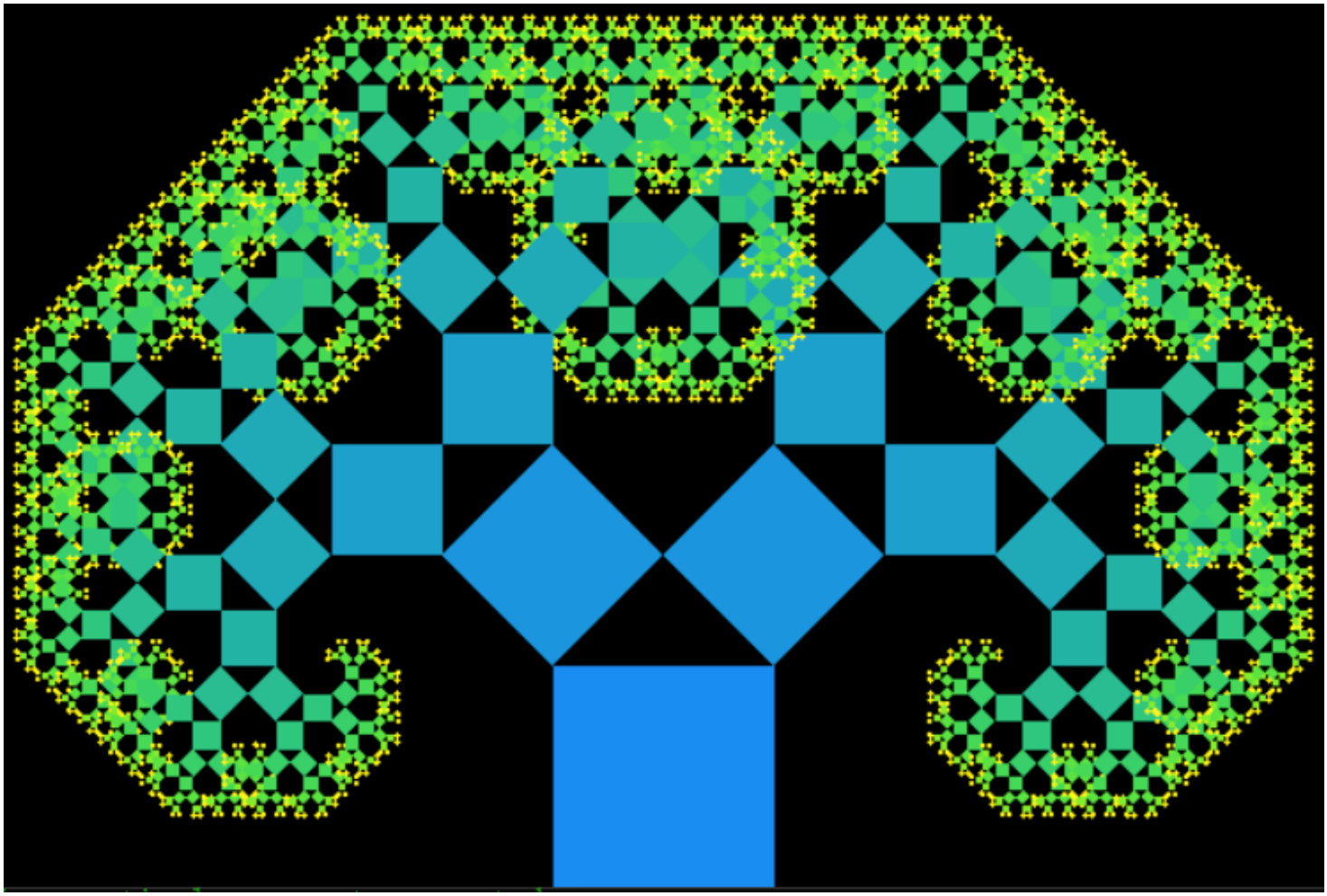
## Key Concepts and Algorithms:

The basis for this assignment is a recursive function. The main algorithm that generates the next square's size is the original square's scale multiplied by 1 over the square root of 2. Unlike the other assignments in this semester this assignment didn't use a class or memory allocation to generate the end result.

## What I learned in this assignment:

In the course of making this assignment I gained a greater understanding of recursive functions. I also learned about SFML rectangle shapes and other shapes in the SFML documentation that I read when completing this assignment.

## Ps2: Screen shot



## Ps5 Source Code: Makefile

```
CC= g++
CFLAGS= -g -O3 -Wall -Werror -std=c++0x
DEPS= -lsfml-system

all:    ED

ED: main.o ED.o
    $(CC) main.o ED.o -o ED $(DEPS)

main.o: main.cpp ED.hpp
    $(CC) -c main.cpp ED.hpp $(CFLAGS)

ED.o:   ED.cpp ED.hpp
    $(CC) -c ED.cpp ED.hpp $(CFLAGS)

clean:
    rm *.o
    rm *.gch
    rm ED
```

## Ps5 Source Code: PTree.cpp

```
// Name: John Simonson
// Date: 1/27/2020
// Assigment: ps2
#include<SFML/Graphics.hpp>
#include<iostream>
#include<string>
#include<cmath>
#include<vector>
using namespace std;
using sf::Color;
//argv[1] = L = ("Length of window")
//argv[2] = N = ("Depth of tree")
//Scale next = size * 1/sqrt2

void draw(int newDepth, sf::RectangleShape root, sf::RenderWindow &window);

int length;
int width;
int size;
int depth;
```

```
21  int main(int argc, char* argv[]){
22      string temp = argv[1];
23      string tempDepth = argv[2];
24      depth = stoi(tempDepth);
25      size = stoi(temp);
26      length = size * 6;
27      width = size * 4;
28
29      sf::RenderWindow window(sf::VideoMode(length, width), "PS2");
30              sf::RectangleShape root(sf::Vector2f(size, size));
31              root.setPosition(length/2 - size/2, width-size);
32              root.setFillColor(Color((255/depth),255-(depth*10), + (depth*20)));
33              window.draw(root);
34              draw((depth - 1), root, window);
35
36
37          while (window.isOpen())
38      {
39          sf::Event event;
40          while (window.pollEvent(event))
41          {
42              if (event.type == sf::Event::Closed)
43                  window.close();
44
45
46
47          }
48      }
49      return 0;
50  }
51
52  void draw(int newDepth, sf::RectangleShape root, sf::RenderWindow &window){
53          sf::Event event;
54          while (window.pollEvent(event))
55          {
56              if (event.type == sf::Event::Closed)
57                  window.close();
58
59          }
60
61      if(newDepth <= 0){
62          return;
63      }
64      sf::RectangleShape left, right;
65      left.setOrigin(0, root.getSize().y);
66      right.setOrigin(root.getSize().x, root.getSize().y);
```

```
67      left.setSize(root.getSize());
68      right.setSize(root.getSize());
69      left.setScale(root.getScale().x*(1/sqrt(2)),root.getScale().y*(1/sqrt(2)));
70      right.setScale(root.getScale().x*(1/sqrt(2)),root.getScale().y*(1/sqrt(2)));
71      left.setPosition(root.getTransform().transformPoint(root.getPoint(0)));
72      left.setRotation(root.getRotation() - 45);
73      right.setPosition(root.getTransform().transformPoint(root.getPoint(1)));
74      right.setRotation(root.getRotation() + 45);
75
76      left.setFillColor(Color((255/newDepth),255-(newDepth*10), (newDepth*20)));
77      right.setFillColor(Color((255/newDepth),255-(newDepth*10), (newDepth*20)));
78      window.draw(right);
79      window.display();
80      draw((newDepth - 1), right, window);
81      window.draw(left);
82      window.display();
83      draw((newDepth - 1), left, window);
84      return;
85  }
```