```
    1: // Copyright 2022 Anson Cheang
    2: /**
    3: * TFractal.cpp - essentially the main function,
    4: which calls upon the recursive function to create
    5: each new triangle and draw them out in different color
    6: *
    7: * Date 2/22/22 - 2/28/22
    8: *
    9: * Created by: Anson Cheang
   10: *
   11: */
   12: #include "Triangle.h"
   13: #include <iostream>
   14: #include <cstdlib>
   15: #include <cmath>
   16: #include <SFML/System.hpp>
   17: #include <SFML/Window.hpp>
   18: #include <SFML/Graphics.hpp>
   19:
   20: // using namespace std;
   21:
   22: void fTree(sf::RenderWindow& window, Triangle ET, double size, int depth)
;
   23:
   24: int main(int argc, char* argv[]) {
   25:     /*double windowSize = atoi(argv[1]);
   26:     double currentSize = atoi(argv[1]);
   27:     for(int i = 1; i < atoi(argv[2]); i++)
   28:     {
   29:         currentSize = (sqrt(3)/4) * pow(currentSize, 2);
   30:         currentSize = currentSize/4;
   31:         currentSize = currentSize * 4/sqrt(3);
   32:         currentSize = sqrt(currentSize);
   33:         windowSize += currentSize;
   34:     }*/
   35:
   36:     sf::RenderWindow window(sf::VideoMode(700, 700), "Input");
   37:     double height = sqrt(3)/2*atoi(argv[1]);
   38:     sf::Vector2f position;
   39:     position.x = (700/2) - atoi(argv[1])/2;
   40:     position.y = (700/2) - (height/2);
   41:     Triangle triangle(atoi(argv[1]), position, 'n');
   42:
   43:     while (window.isOpen()) {
   44:         sf::Event event;
   45:         while (window.pollEvent(event)) {
   46:             if (event.type == sf::Event::Closed) {
   47:                 window.close();
   48:             }
   49:         }
   50:
   51:         window.clear(sf::Color::White);
   52:         fTree(window, triangle, atoi(argv[1]), atoi(argv[2]));
   53:         // window.draw(triangle);
   54:         window.display();
   55:     }
   56:
   57:     return 0;
   58: }
   59:
   60: void fTree(sf::RenderWindow& window, Triangle ET, double size, int depth)
 {
   61:     window.draw(ET);
   62:     if (depth > 0) {
   63:         sf::Vector2f position = ET.getP1();
```

```
64:         size = size/2;
65:         position.y = position.y - size * sqrt(3) / 2;
66:         position.x = position.x - size/2;
67:         Triangle T1(size, position, 'g');
68:         fTree(window, T1, size, depth - 1);
69:         // position = ET.getP2();
70:         Triangle T2(size, ET.getP2(), 'r');
71:         fTree(window, T2, size, depth - 1);
72:         position = ET.getP3();
73:         position.x = position.x - size;
74:         Triangle T3(size, position, 'b');
75:         fTree(window, T3, size, depth - 1);
76:     }
77: }
```