```
 1: CC = g++
 2: CFLAGS = -Wall -Werror -pedantic --std=c++14
 3: LIBS = -lboost_unit_test_framework
 4: DEPS = Triangle.h
 5: SFMLFLAGS = -lsfml-graphics -lsfml-window -lsfml-system
 6:
 7: %.o: %.cpp $(DEPS)
 8:         $(CC) $(CFLAGS) -c $<
 9:
10: all: TFractal
11:
12: TFractal: TFractal.o Triangle.o
13:         $(CC) $(CFLAGS) -o TFractal $^  $(LIBS) $(SFMLFLAGS)
14:         cpplint --filter=-runtime/references *.cpp *.h
15:
16: clean:
17:         rm *.o TFractal
```

```
    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: }
```

```
 1: // Copyright 2022 Anson Cheang
 2: #ifndef _HOME_IIFORCE_BADNAME_COMP4_PS3_TRIANGLE_H_  // Triangle_H_
 3: #define _HOME_IIFORCE_BADNAME_COMP4_PS3_TRIANGLE_H_  // Triangle_H_
 4:
 5: #include <string>
 6: #include <cstdlib>
 7: #include <iostream>
 8: #include <SFML/System.hpp>
 9: #include <SFML/Window.hpp>
10: #include <SFML/Graphics.hpp>
11:
12: class Triangle : public sf::Drawable{
13:  public:
14: Triangle(double val, sf::Vector2f position, char color);
15: sf::Vector2f getP1();
16: sf::Vector2f getP2();
17: sf::Vector2f getP3();
18:
19:  private:
20: void draw(sf::RenderTarget& target, sf::RenderStates states) const;
21: double size;
22: sf::Vector2f P1;
23: sf::Vector2f P2;
24: sf::Vector2f P3;
25: sf::ConvexShape shape;
26: };
27:
28:
29: #endif  // _HOME_IIFORCE_BADNAME_COMP4_PS3_TRIANGLE_H_
```

```
    1: // Copyright 2022 Anson Cheang
    2: /**
    3: * Triangle.cpp - as an implementation to create a new triangle object to
store every point
    4: and draw out the triangle at a moments notice
    5: *
    6: * Date 2/22/22 - 2/28/22
    7: *
    8: * Created by: Anson Cheang
    9: *
   10: */
   11: #include "Triangle.h"
   12: #include <string>
   13: #include <cstdlib>
   14: #include <iostream>
   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:
   23: Triangle::Triangle(double val, sf::Vector2f position, char color) {
   24:     size = val;
   25:     sf::Vector2f point1 = position, point2 = position;
   26:     point1.x = point1.x + val;
   27:     point2.x = (position.x + point1.x)/2;
   28:     point2.y = point2.y + sqrt(3)/2 * val;
   29:     P1 = position;
   30:     P2 = point1;
   31:     P3 = point2;
   32:     shape.setPointCount(3);
   33:     shape.setPoint(0, position);
   34:     shape.setPoint(1, point1);
   35:     shape.setPoint(2, point2);
   36:     shape.setOutlineThickness(1);
   37:     if (color == 'g') {
   38:         shape.setOutlineColor(sf::Color::Green);
   39:     } else if (color == 'r') {
   40:         shape.setOutlineColor(sf::Color::Red);
   41:     } else if (color == 'b') {
   42:         shape.setOutlineColor(sf::Color::Blue);
   43:     } else {
   44:         shape.setOutlineColor(sf::Color::Black);
   45:     }
   46: }
   47:
   48: sf::Vector2f Triangle::getP1() {
   49:     return P1;
   50: }
   51:
   52: sf::Vector2f Triangle::getP2() {
   53:     return P2;
   54: }
   55:
   56: sf::Vector2f Triangle::getP3() {
   57:     return P3;
   58: }
   59:
   60: void Triangle::draw(sf::RenderTarget& target, sf::RenderStates states) co
nst {
   61:     target.draw(shape, states);
   62: }
```