PS1
Recursive Graphics (Sierpinski's Triangle)

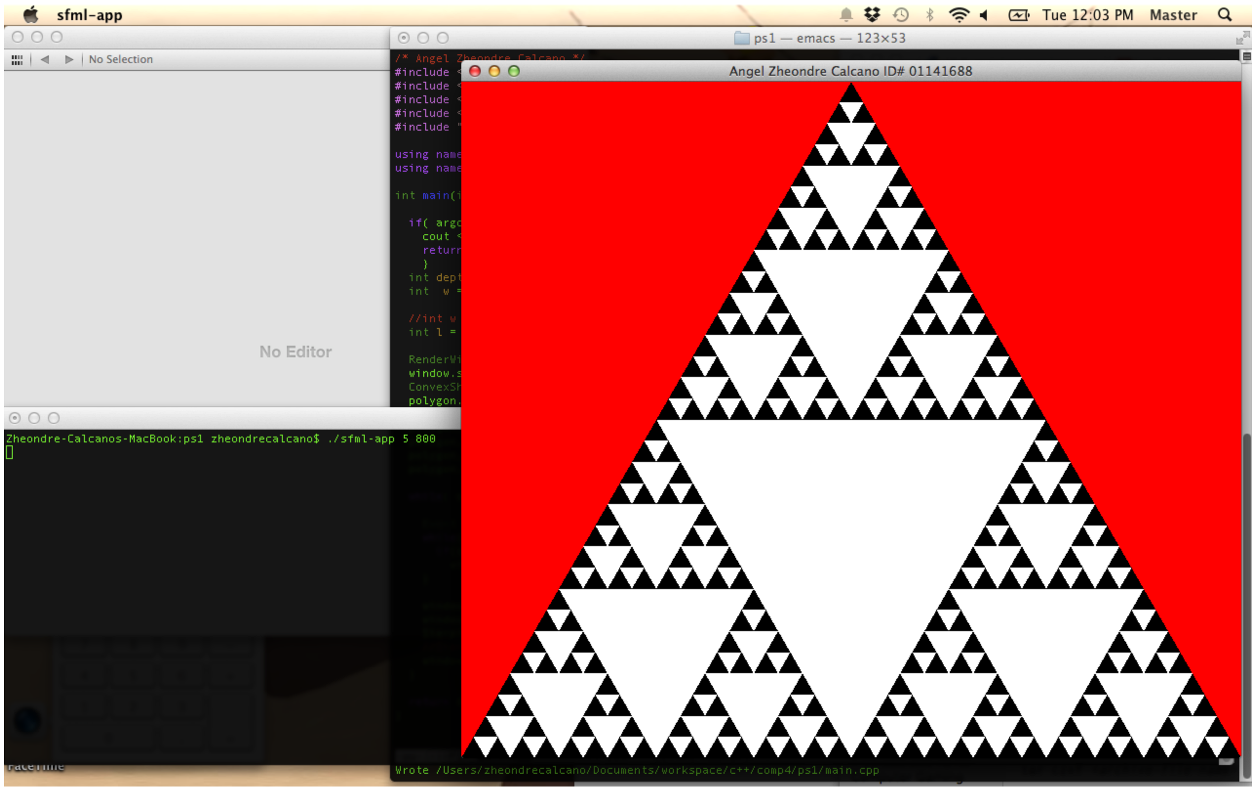
Assignment Summary:
Responsible of writing a program to plot a Sierpinski triangle, Sierpinski triangle was described by a polish mathematician Waclaw Sierpinski in 1915, this complex pattern can be developed with a small simple short recursive program. This program needs to call the Sierpinski function once, draw the first triangle then call the function three more times. After that have every other function that is called to call three more functions until a base case is hit. After the program is running, a separate pattern is required to be made using the same algorithm

What I learned:
How to use the SFML library to recursively draw images to the screen, and small algorithms can create complex design patterns,

Design:
Used the SFML library to make shapes and print them on the screen, followed the algorithm to have the proper design.

What I accomplished : Sierpinski's Triangle
After many hours of debugging I was able to complete both, using a pattern with circles was a bit weird, and I had to change the math in order to use circles, to be honest I didn't like the way my pattern came out but I didn't have enough time to go back and change it. I designed a math algorithm for sierpinski to find the lowest midpoint of the current triangle. Once it is found find the two points that is found from side*sin(60) and side*cos(60). Once that is found pass the location of the midpoints of the three smaller triangles into three other function calls.

Output: Sierpinski's Triangle & MyCircles



```
/* Angel Zheondre Calcano */
#include <
#include <
#include <
#include <
#include "

using name
using name

int main(i

  if( argc
    cout <
    return
    }
  int dept
  int  w =

  //int w
  int l =

  RenderWi
  window.s
  ConvexSh
  polygon.
```

```
Zheondre-Calcanos-MacBook:ps1 zheondrecalcano$ ./sfml-app 5 800
```

```
Wrote /Users/zheondrecalcano/Documents/workspace/c++/comp4/ps1/main.cpp
```

```
all: original sierpinski

original: main.o func.o
	g++ main.o func.o -o original -lsfml-graphics -lsfml-window -lsfml-system

main.o: main.cpp
	g++ -c main.cpp -Wall -Werror -ansi -pedantic

func.o: func.cpp
	g++ -c func.cpp -Wall -Werror -ansi -pedantic

sierpinski: smain.o sierpinski.o
	g++ smain.o sierpinski.o -o sierpinski -lsfml-graphics -lsfml-window -lsfml-system

smain.o: smain.cpp
	g++ -c smain.cpp -Wall -Werror -ansi -pedantic

sierpinski.o:
	g++ -c sierpinski.cpp -Wall -Werror -ansi -pedantic

clean:
	rm *.o original sierpinski *~
```

```
/* Angel Zheondre Calcano */

#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <cmath>
#include <iostream>
#include "sierpinski.hpp"

using namespace sf ;
using namespace std ;

int main(int argc, char* argv[]) {

  if( argc < 3 ) {
    cout << "sierpinski [recursion-depth][side-length]" << endl ;
    return -1 ;
    }
  int depth = atoi(argv[1]) ;
  int  w = atoi(argv[2]) ;

  int l = (int)(.5*sqrt(3.)*(float)w) ;

  sf::RenderWindow window(VideoMode(w,l), "Angel Zheondre Calcano ID# 01141688" ) ;
  window.setFramerateLimit(1) ;
  ConvexShape polygon;
  polygon.setPointCount(3);
  polygon.setPoint(0, Vector2f(w/2,0));
  polygon.setPoint(1, Vector2f(0,l));
  polygon.setPoint(2, Vector2f(w,l));
  polygon.setFillColor(Color::Black);
  polygon.setPosition(0,0);

  while( window.isOpen()){

    Event event ;
    while( window.pollEvent(event)) {
      if(event.type == Event::Closed )
        window.close() ;
    }

    window.clear(Color::Red) ;
    window.draw(polygon);
    Sierpinski( w/2, depth, w/2, l, window) ;
    window.display() ;
  }

  return 0 ;
}
```

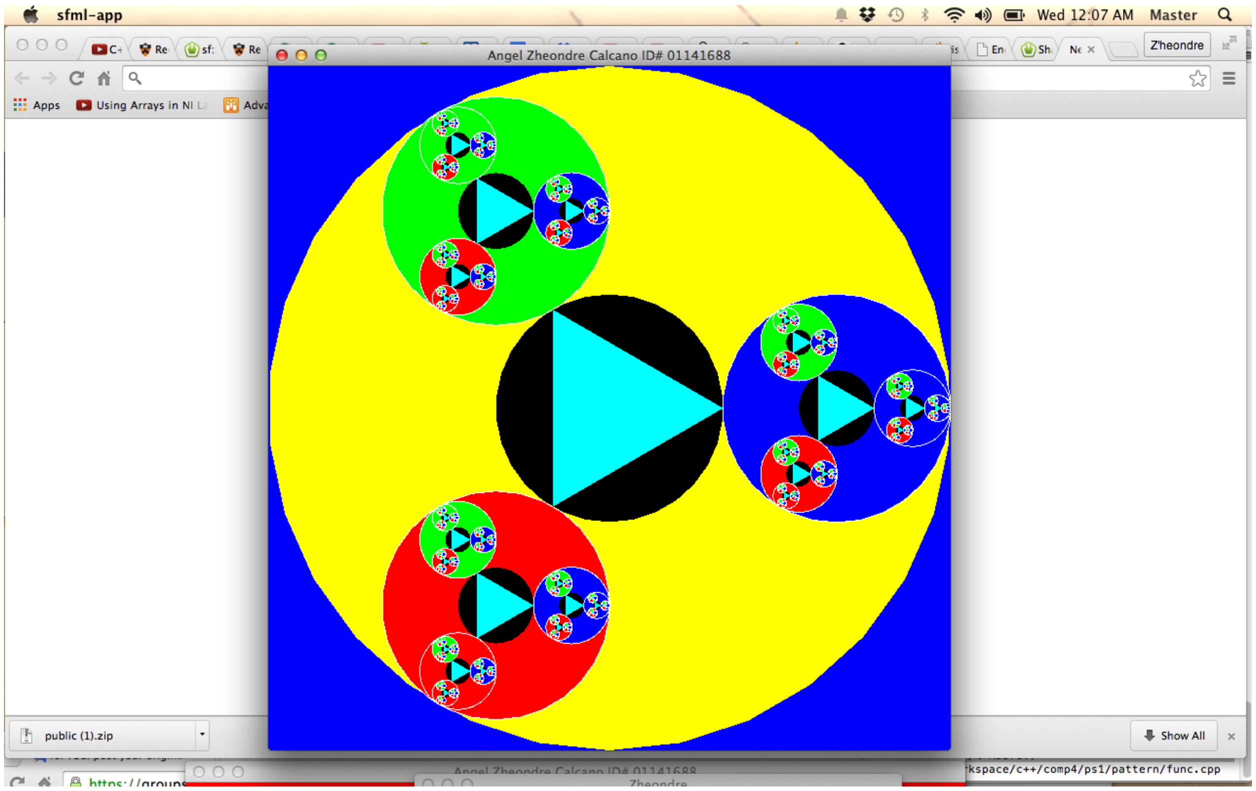
```
#ifndef _sierpinski
#define _sierpinski

#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>

using namespace sf ;
using namespace std ;

int Sierpinski( double side, double depth, double p2x, double p2y, sf::RenderWindow& a) ;

#endif
```

```cpp
// Angel Zheondre Calcano
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <cmath>
#include <math.h>
#include <iostream>
#include "sierpinski.hpp"

using namespace sf ;
using namespace std ;

int Sierpinski( double side, double depth, double p2x, double p2y, sf::RenderWindow& a ) {

  if(depth == 0) return 0 ;

  double p0y, p0x, p1y, p1x, mp1x, mp1y, mp2x, mp2y, mp3x, mp3y ;

  double ylnd = (.866)*side ;

  //p0’s values, left point inner triangle
  p0y = p2y - ylnd ;
  p0x = p2x - side*(.5) ;

  //p1’s values, right point inner triangle
  p1y = p2y - ylnd ;
  p1x = p2x + side*(.5) ;

  //Draw picture of triangle
  ConvexShape T ;
  T.setPointCount(3) ;
  T.setPoint(0, Vector2f(p2x,p2y));
  T.setPoint(1, Vector2f(p1x,p1y));
  T.setPoint(2, Vector2f(p0x,p0y));
  T.setPosition(0,0);
  a.draw(T) ;

  // Find the midpoints of the other triangles
  mp1x = p2x + side/2 ;
  mp1y = p2y ;
  mp2x = p2x ;
  mp2y = p0y ;
  mp3x = p2x - side/2 ;
  mp3y = p2y  ;

  Sierpinski( side/2 , depth - 1, mp1x, mp1y, a ) ;
  Sierpinski( side/2 , depth - 1, mp2x, mp2y, a ) ;
  Sierpinski( side/2 , depth - 1, mp3x, mp3y, a ) ;

  return 0 ;
}
```

What I accomplished : My Circles

For the second program I used a similar method. But I did the design based on 120 degrees, instead of 60 degrees. Only thing hard about it was trying to figure out why I kept getting zero when I casted a value as a double or an int. Too much time was spent on debugging. What was annoying was that I had to specify the origin of each circle when I set the position for that shape. If not specified it would be set to 0,0. I didn't use any data structures and I specified the frames to be set to one.

Output: MyCircles



```cpp
/* Angel Zheondre Calcano */
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <cmath>
#include <iostream>
#include "func.hpp"

using namespace sf ;
using namespace std ;

int main(int argc, char* argv[]) {

  if( argc < 3 ) {
    cout << "sierpinski [recursion-depth][side-length]" << endl ;
    return -1 ;
  }
  int depth = atoi(argv[1]) ;
  int  w = atoi(argv[2]) ;

  RenderWindow window(VideoMode(w,w), "Angel Zheondre Calcano ID# 01141688"
) ;
  window.setFramerateLimit(1) ;

  CircleShape circle ;
  circle.setRadius(w/2) ;
  circle.setPosition(0, 0) ;
  circle.setFillColor(Color::Yellow);


  while( window.isOpen()){

    Event event ;
    while( window.pollEvent(event)) {
      if(event.type == Event::Closed )
        window.close() ;
    }

    window.clear(Color::Blue) ;
    window.draw(circle);
    Sierpinski((.5)*(w/3), depth, w/2, w/2, window) ;
    //Sierpinski( 250, 5, 250, 433, window) ;

    window.display() ;
  }

  return 0 ;
}
```

```
#ifndef _func
#define _func

#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>

using namespace sf ;
using namespace std ;

int Sierpinski( double side, double depth, double p2x, double p2y, sf::RenderWindow& a);

#endif
```

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <cmath>
#include <math.h>
#include <iostream>
#include "func.hpp"

using namespace sf ;
using namespace std ;

int Sierpinski( double side, double depth, double p2x, double p2y, sf::RenderWindow& a ) {
  //reminder side = side/3
  if(depth == 0) return 0 ;

  double p0y, p0x, p1y, p1x, p3y, p3x;

  double s = .866 ;
  double c = .5 ;

  //p0's values, left circ
  p0y = p2y - side*2*s ;
  p0x = p2x - side*2*c ;

  //p1's values, right circ pt
  p1y = p2y  ;
  p1x = p2x + side*2 ;

  //buttom circle
  p3y = p2y + side*2*s ;  //side*2 - side*(1/3)  ;
  p3x = p2x - side*2*c ;

  CircleShape circle ;
  circle.setRadius(side) ;
  circle.setOrigin(side, side) ;
  circle.setPosition( (float)p2x, (float)p2y) ;
  circle.setFillColor(Color::Black);
  a.draw(circle);

  CircleShape cb ;
  cb.setRadius(side) ;
  cb.setOrigin(side,side) ;
  cb.setPosition((float)p0x,(float)p0y) ;
  cb.setFillColor(Color::Green);
  cb.setOutlineThickness(1);
  cb.setOutlineColor(sf::Color(250, 255, 255));
  a.draw(cb);

  CircleShape cc ;
  cc.setRadius(side) ;
  cc.setOrigin(side,side) ;
  cc.setPosition(p1x, p1y) ;
  cc.setFillColor(Color::Blue);
  cc.setOutlineThickness(1);
  cc.setOutlineColor(sf::Color(250, 255, 255));
  a.draw(cc);

  CircleShape cd ;
  cd.setRadius(side) ;
  cd.setOrigin(side,side) ;
  cd.setPosition(p3x, p3y) ;
  cd.setFillColor(Color::Red);
  cd.setOutlineThickness(1);
  cd.setOutlineColor(sf::Color(250, 255, 255));
  a.draw(cd);
  //p2's val, bottum circ pt

  /*    if( depth == 1 ) {
        cout << "trval test " << endl ;
        cout << p0x << " "<< p0y << endl ;
        cout << p1x << " "<< p1y << endl ;
        cout << p2x << " "<< p2y << endl ;
        cout << side  << endl ;
    }
  */

  //Draw picture of middle tri

  ConvexShape T ;
  T.setPointCount(3) ;
  T.setPoint(0, Vector2f(p2x + side,p2y));
  T.setPoint(1, Vector2f(p2x - side*c,p2y - side*s));
  T.setPoint(2, Vector2f(p2x - side*c,p2y + side*s));
  T.setPosition(0,0);
  T.setFillColor(Color::Cyan);
  a.draw(T) ;

  Sierpinski( side/3 , depth - 1, p0x, p0y, a ) ;
  Sierpinski( side/3 , depth - 1, p1x, p1y, a ) ;
  Sierpinski( side/3 , depth - 1, p3x, p3y, a ) ;

  return 0 ;

}
```

What I learned:
How to use the SFML library to recursively draw images to the screen, and small algorithms can create complex design patterns,

Design:
Used the SFML library to make shapes and print them on the screen, followed the algorithm to have the proper design.