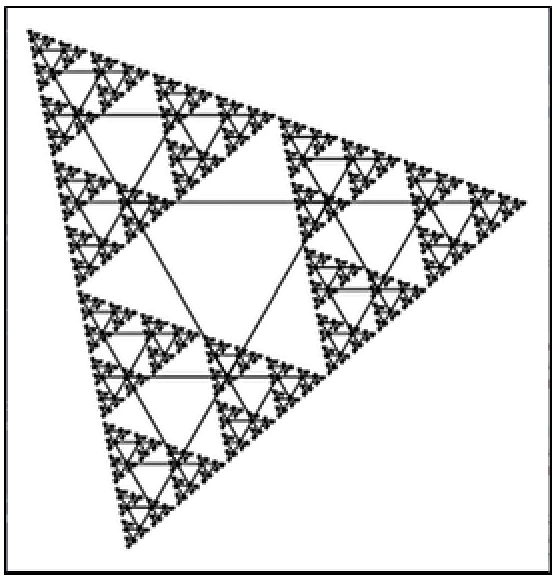
PS2: Recursive Graphics (Triangle Fractal)

In this assignment, you will write a program that plots a Triangle Fractal as illustrated below. It is a variation on the Sierpinski triangle. The Polish mathematician Waclaw Sierpinski described the pattern in 1915, but it has appeared in Italian art since the 13th century.



1 Pair Programming

On this assignment, you are encouraged (but not required) to work with a partner provided you practice **pair programming**. Pair programming is "*a practice in which two programmers work side-by-side at one computer, continuously collaborating on the same design, algorithm, code, or test.*" One partner is driving (designing and typing the code) while the other is navigating (reviewing the work, identifying bugs, and asking questions). The two partners switch roles every 30-40 minutes, and on demand, brainstorm.

If you are working with a partner, you should remember

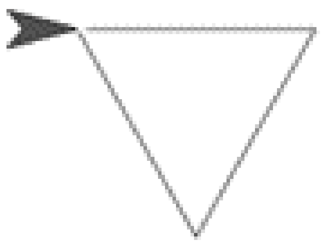
- Make a single submission with both partners names listed in the group when submitting to Gradescope.
- You are responsible for making sure that you were included in the submission.
- You are responsible for making sure that your submission fulfills the project requirements.
- You should clearly identify both partners in your `Readme-ps2.md` file.

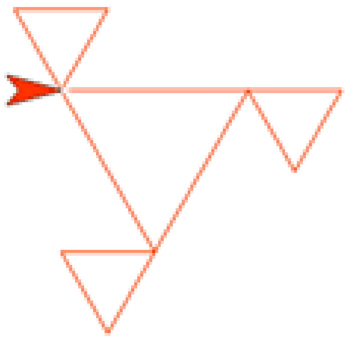
2 Details

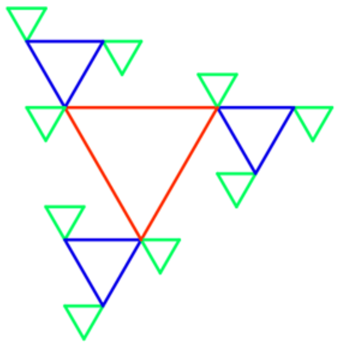
Your task it to write a program `Triangle` with a *recursive* function `fractal()` and a `main()` program that calls the recursive function. You may write a recursive helper function that `fractal()` calls rather than making `fractal()` itself recursive.

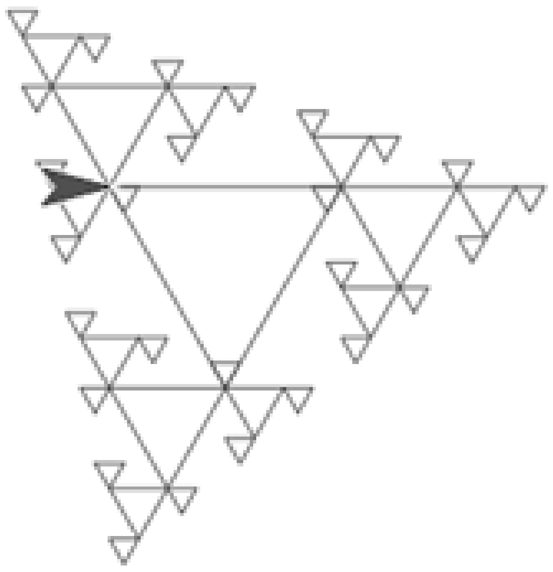
Your program shall take two command-line arguments L and N (in that order):

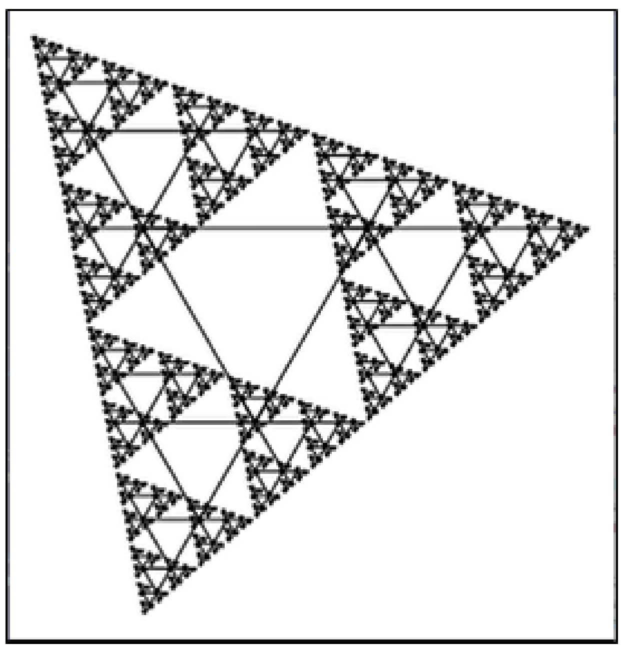
- L The length of one side of the base equilateral triangle (`double`)
- N The depth of the recursion (`int`)


Base pentagon


Iteration 1


Iteration 2


Iteration 3


Final

You may want to implement a class that derives from `sf::Drawable` (or one of its subclasses). You can then have it draw itself to your main window.

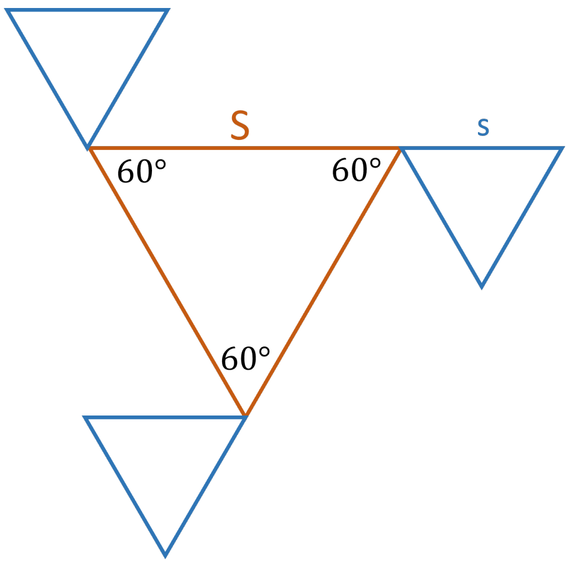
The `sf::CircleShape` and `sf::ConvexShape` classes are both good ways to draw a triangle. Note that the SFML libraries expect angles to be in degrees, but the standard math libraries expect radians. You can use the `M_PI` constant in the `<cmath>` library to help convert between radians and degrees. Also, the `sf::Shape` subclasses use the upper-left corners of their bounding boxes as their origin rather than their centers, which affects rotation transforms. You can use `sf::Shape::setOrigin()` to change this behavior.

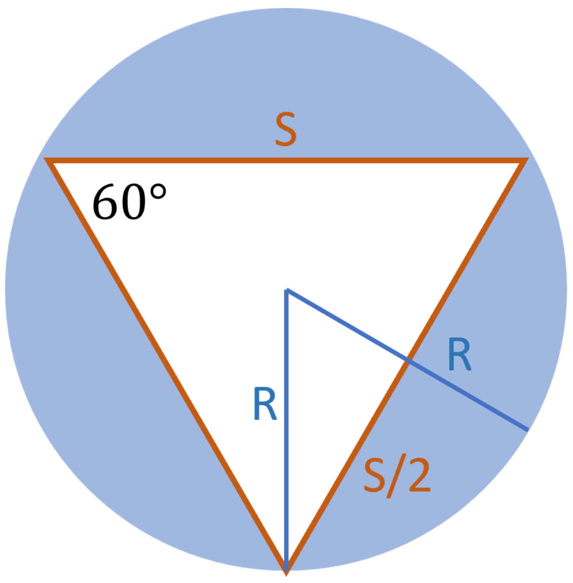
You should create an SFML window that is sized appropriately for your final image. A large value for L should not cause the image to spill over the boundary of your window and a small value should not cause most of the window to be empty space.

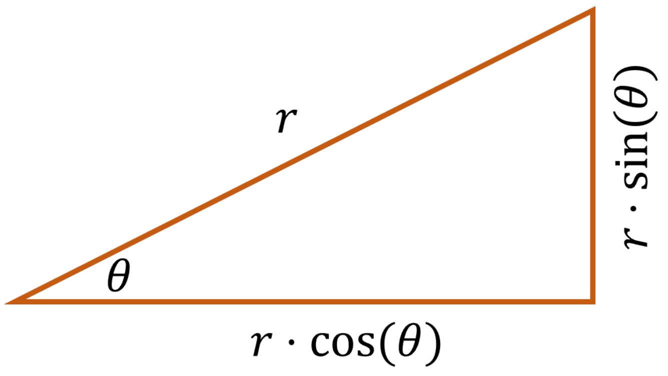
3 Math

Here is some help with the mathematics of the fractal.

- Each step involves drawing three smaller triangles.
- Each child triangle has a side whose length is half that of it's parent.
- Each child triangle shares a corner with its parent.
- One edge of the child triangle is co-linear with its parent.
- The radius of a circumcircle of an equilateral triangle is smaller than the side of the inscribed triangle with $r = \frac{s}{\sqrt{3}}$.



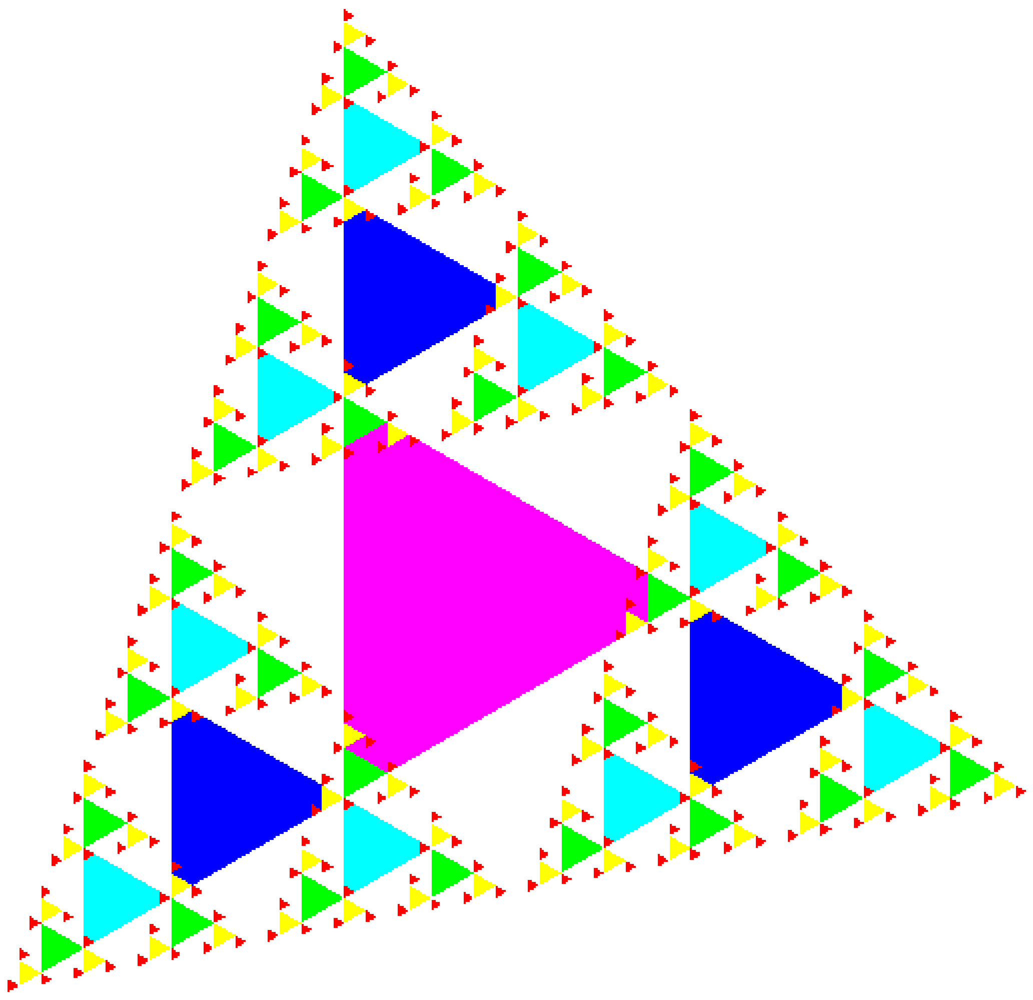




4 Extra Credit

You can earn extra credit by making use of multiple colors in your fractal. These colors must be chosen in a consistent manner to make for an interesting pattern (such as by choosing the color by position).

Additionally, you can adapt the fractal to take an additional optional parameter denoting the rotation of the overall triangle (make sure that it defaults to point down with the emergent triangle pointing towards the upper-left corner). You can also add other reasonable features, such as animation. If you do any of the extra credit work, make sure to describe exactly what you did in `Readme-ps2.md`.



5 What to turn in

Your makefile should build a program named `Triangle` which takes two command line arguments L and N (in that order).

Submit a zip archive to Gradescope containing:

- Your `main.cpp`
- Your `triangle.cpp` and `triangle.hpp`.
- The makefile for your project. The makefile should have targets `all`, `Triangle`, `lint`, and `clean`. Make sure that all prerequisites are correct.
- Your `Readme-ps2.md` that includes
  1. Your name
  2. Statement of functionality of your program (e.g. fully works, partial functionality, extra credit)
  3. Any other notes
- Any other source files that you created.
- Any images, fonts, or other resources used
- A screenshot of program output

Make sure that all of your files are in a directory named `ps2` before archiving it and that there are no `.o` or other compiled files in it.

6 Grading rubric

Feature	Points	Comment
Autograder	4	
	2	Submits required files
	2	Has targets `all`, `pname`, `lint`, and `clean` and builds.
Drawing	14	
	2	Draws a window with at least one triangle
	4	Draws the central triangle
	4	Draws the three child triangles
	4	Recurses to the correct depth (four triangles when n = 1).
Placement	14	
	4	Triangles are placed point-to-point.
	4	Size of pentagons matches expected parameters
	3	Has proper orientation
	3	Draws the child clockwise of the point
Window	3	
	3	Sizes window to fit image
Documentation	5	Complete
	1	The readme describes how the fractal is represented
	2	The readme describes how child positions are calculated
	1	The readme describes how the team adapted to pair programming
	1	Has a screenshot representative of the program's behavior
Extra Credit	6	
	+2	Uses multiple colors in a reasonable manner
	+2	Rotates fractal based on extra parameter
	+2	Other reasonable extension.
Penalties		
	-2	Non-recursive implementation
	-2	Input comes from `stdin` rather than `argv`
	-5	Linting problems
	-3	Non-const global variables or public fields
	-3	Submission includes `.o` files
	-10%	Each day late
Total	40	