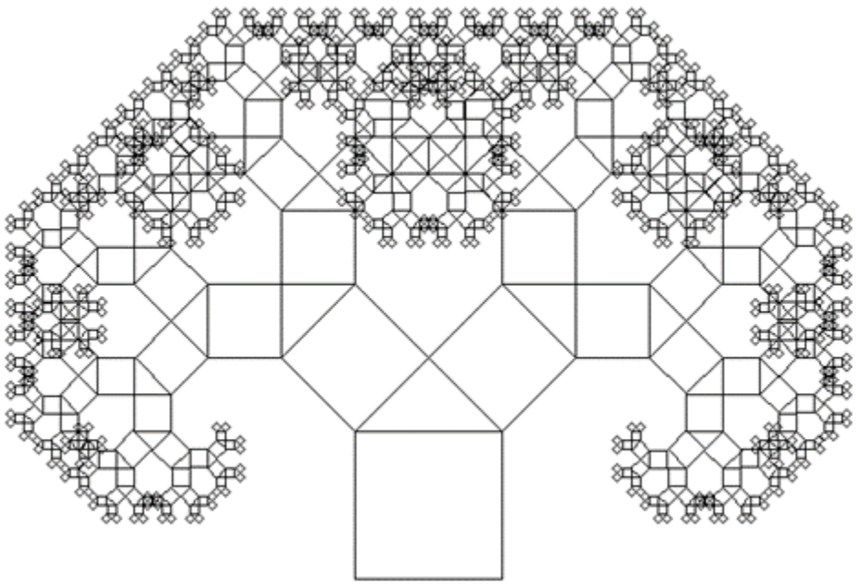
PS2: Recursive Graphics (Pythagoras Tree)

Due: Monday, February 12, 11:59pm

In this assignment, you will write a program that plots a Pythagoras tree as illustrated below.



The Pythagoras tree is named after the Greek mathematician Pythagoras because each triple of the touching squares encloses a right triangle, in a configuration traditionally used to depict the Pythagorean theorem. It is a plane fractal constructed from squares invented by the Dutch mathematics teacher Albert E Bosman in 1942. In 1957, Bosman published a book *on Het wondere onderzoekingsveld der vlakke meetkunde* ("the wonderous exploration field of plane geometry") that contained a description of the Pythagorean tree. If the largest square has a size of $L \times L$, the entire Pythagoras tree fits snugly inside a box of size $6 \cdot L \times 4 \cdot L$.

1 Details

Your task it to write a program `PTree` with a *recursive* function `pTree()` and a `main()` program that calls the recursive function. You may write a recursive helper function that `pTree()` calls rather than making `pTree()` 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 square (`double`)
- N The depth of the recursion (`int`)

Base square

Iteration 1

Iteration 2

Iteration 3

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.

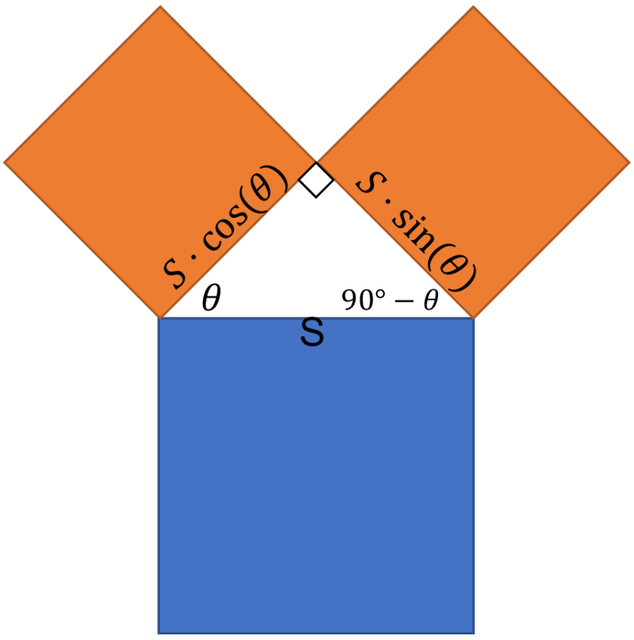
The `sf::RectangleShape` and `sf::ConvexShape` classes are both good ways to draw a square. Note that the SFML libraries expect angles to be in degrees, but the standard math libraries expect radians. You can use the `M_PI` constant in the `<cmath>` library to help convert between radians and degrees. 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.

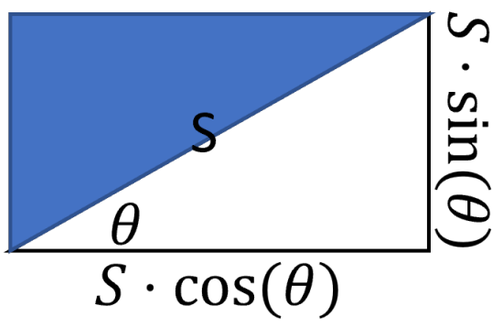
You should create an SFML window that is sized appropriately for your final image. A large value for L should not cause the image to spill over the boundary of your window and a small value should not cause most of the window to be empty space.

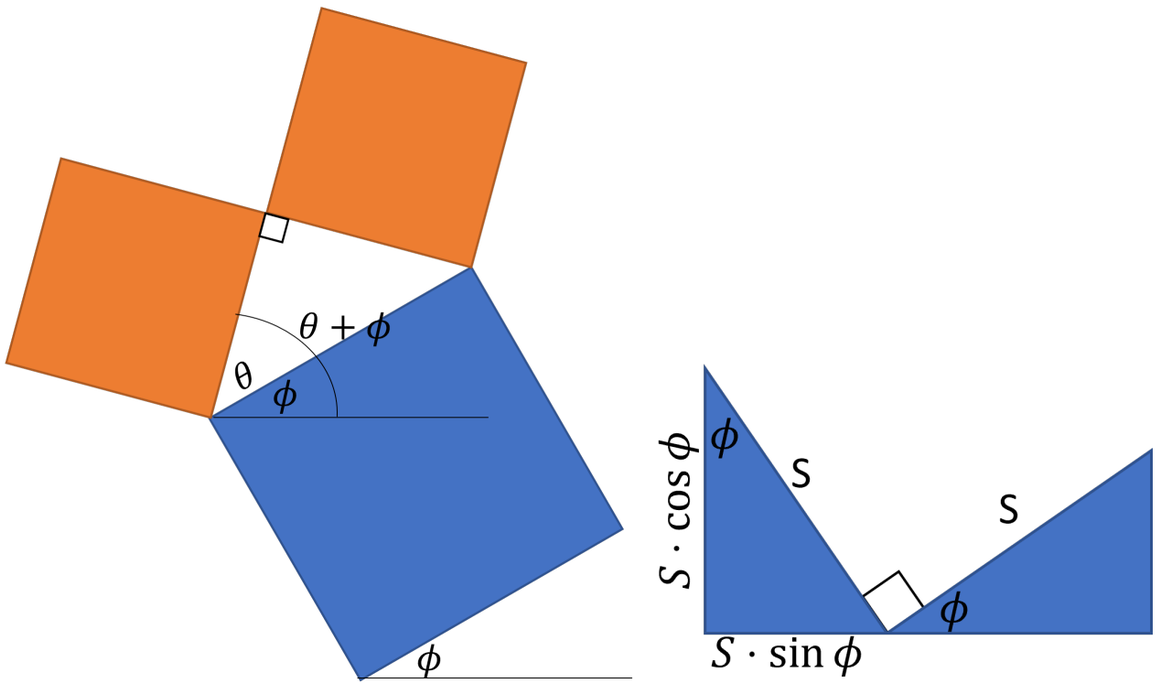
2 Math

Here is some help with the mathematics of the Pythagorean tree.

- Each step involves drawing two smaller Pythagorean trees.
- If the parent square has side S than the two child squares have side $S \cdot \cos(\theta)$ and $S \cdot \sin(\theta)$.
- If $\theta = 45°$ then $\cos(\theta) = \sin(\theta) = \frac{\sqrt{2}}{2}$ and the two squares will be the same size. If you are doing the extra credit, the two squares will be of different sizes.

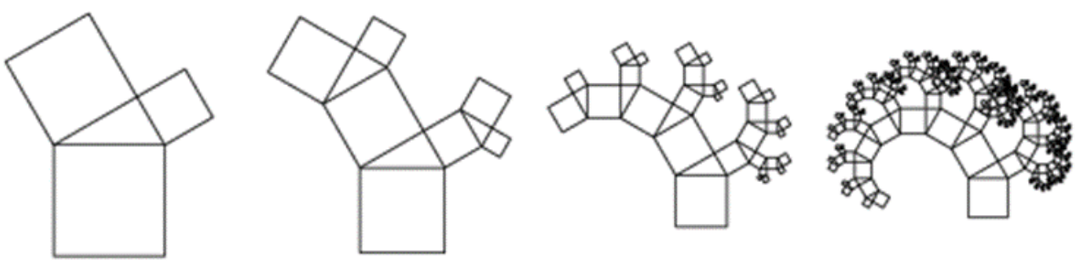


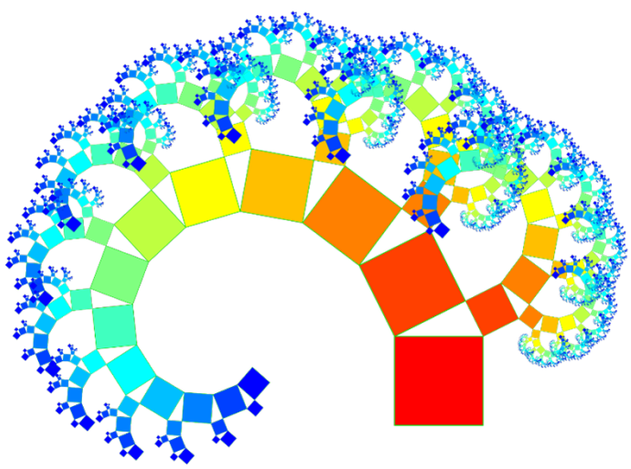




3 Extra Credit

You can earn extra credit by making use of multiple colors in your tree. Additionally, you can adapt the tree to take an additional parameter denoting the angle of the inner triangle (make sure that it has a default of 45°). 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`.





4 What to turn in

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

Submit a zip archive to Blackboard containing:

- Your `main.cpp`
- Your `PTree.cpp` and `PTree.hpp`.
- The makefile for your project. The makefile should have targets `all`, `PTree`, `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.

5 Grading rubric

Feature	Points	Comment
Core Implementation	31	Full & Correct Implementation
	5	Draws the base square
	4	Draws child squares
	4	Recurses to the correct depth.
	6	Squares placed to form a right triangle.
	6	Size of triangles matches expected parameters
	3	Has proper orientation
	3	Sizes window to fit image
	-5	Non-recursive implementation
Makefile	4	
Screenshot	1	
Readme	4	Complete
Extra Credit	7	
	+2	Uses multiple colors at different levels of the tree
	+3	Implements variations of the Pythagoras tree with different angles
	+2	Other reasonable extension.
Penalties		
	-5	Linting problems
	-3	Submission includes `.o` files
	-10%	Each day late
Total	40	