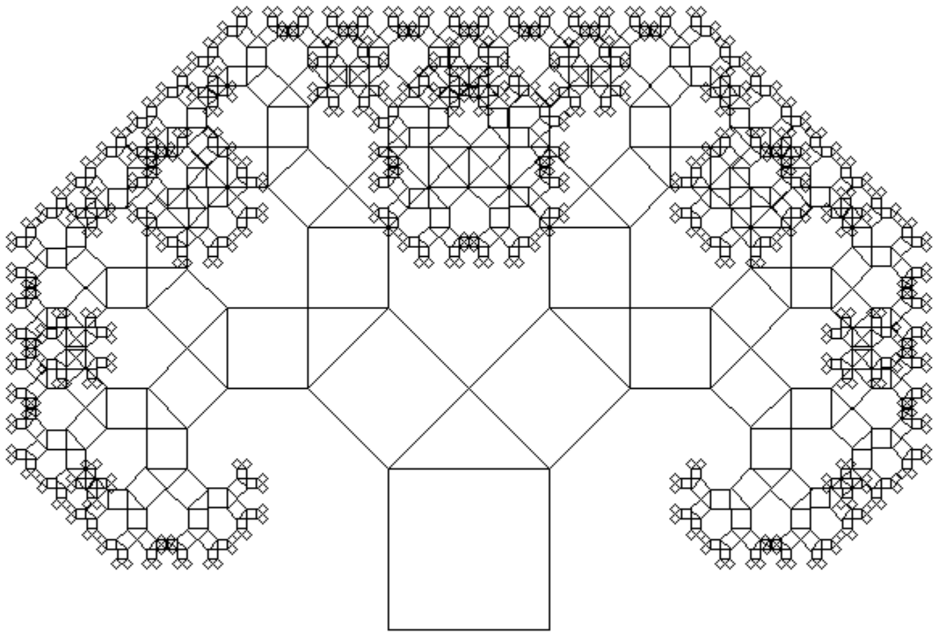
PS1: Recursive Graphics (Pythagoras tree)

In this assignment you will write a program that plots a Pythagoras tree using a square as a base (preferably using recursion), as illustrated below.



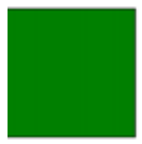
Part 1.

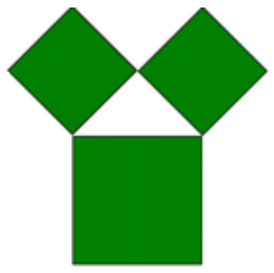
The Pythagoras tree is named after the Greek mathematician Pythagoras because each triple of touching squares encloses a right triangle, in a configuration traditionally used to depict the Pythagorean theorem. It is a plane fractal constructed from squares invented by the Dutch mathematics teacher Albert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

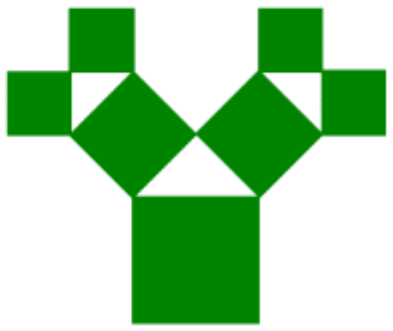
Your task is to write a program `PTree.cpp` with a recursive function `pTree()`, and a `main()` program that calls the recursive function.

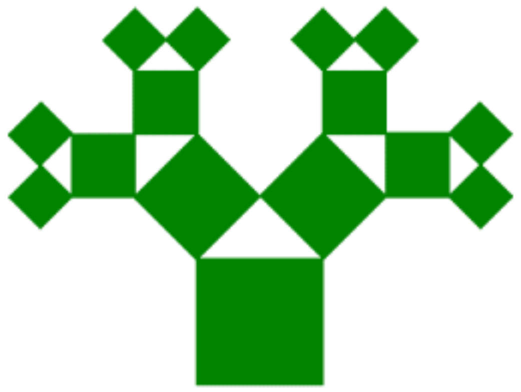
Your program shall take two command-line arguments *L* and *N*:

L	size of the base square (double)
N	the depth of the recursion


Base square


Iteration 1


Iteration 2


Iteration 3

API specification. You should implement class `PTree`

Notes:

- You should create a `PTree` class that derives from `sf::Drawable`. Then, you can have it just `draw` itself to your main window.
- Review Jon's `LineSegment` example to see a simple example of how to do this: LineDemo.tar.gz.
- Using SFML's ConvexShape class is a good way to draw a square.
- Your executable must read two parameters (integers): recursion-depth and window-size. You should create a square SFML window that's exactly as big as the window-size argument, and your triangle should fill it.

What to turn in

It's important that you turn in everything needed to build your projects.

Create a directory that will contain all of your work**.** The directory should be named **ps1** and must contain:

1. Your `Makefile` (see below for further instructions)
2. `.cpp` and `.hpp` files for project
3. Any images and fonts you are using
4. Anything else needed to build and run your code
5. Screenshot of program output
6. A completed version of the `ps1-readme.txt` file (download the template from the PS1 assignment page on Blackboard)

Your `Makefile` should contain two targets: `all` and `clean`. The former should build both executables, and the latter should remove the executables, `.o` files, and all other temporary files created during the build.

Remember, we will have to build and run your code, so make sure to submit all that's needed!

Use `tar` command from the parent directory of your `ps1`:

```
   tar czvf <archive-file-name>.tar.gz ps1
```

to compress your directory structure. Include your name in the archive file name (e.g., `Tom_Wilkes_P1.tar.gz`)

How to turn it in

Submit your compressed archive file via the PS1 assignment page on Blackboard.

Grading rubric

Feature	Value	Comment
PTree implementation	**10**	**full & correct implementation**
		1 pt file name correct
		1 pt reads base square size and depth args
		7 pts draws tree properly (recursive implementation)
		(2 pts for non-recursive implementation)
		1 pt implements draw function as derived class of `sf::Drawable`
Makefile	**6**	**full & correct implementation**
		1 pt builds objects associated with `PTree` project
		1 pt links "tree" executable
		1 pt "make clean" removes temporary files, objects, and executables
tar.gz archive	**2**	**all files packaged in .tar.gz file with correct directory structure**
ps1-readme.txt	**4**	**complete and discusses work**

Total	**22**	
Extra points (see examples below)	**6**	You can implement a variation of Pythagoras tree (with different angles, or an asymmetric tree using a rectangle as a base instead of a square)
	2	Add color to your tree