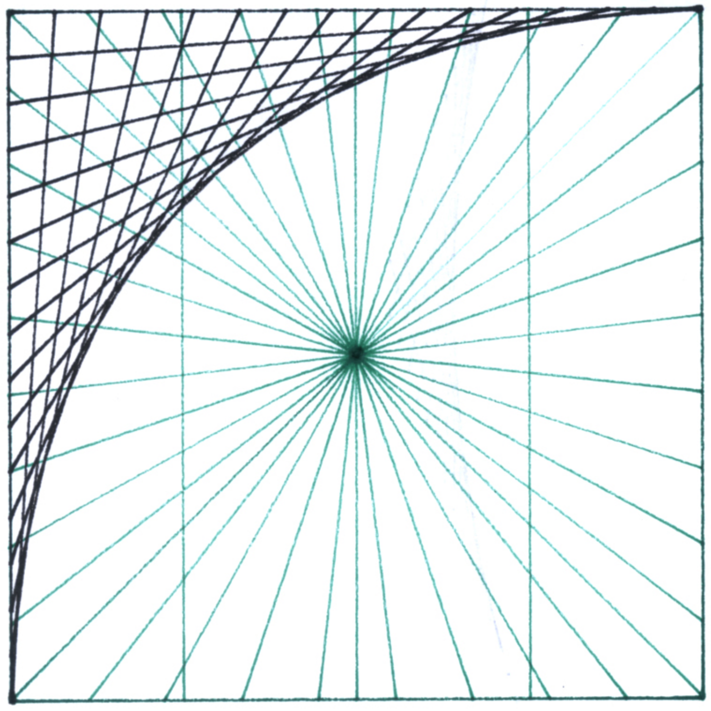
PlotterWriter API

PlotterWriter API is a library created to drive an HP 7475 pen plotter or another plotter that supports is command set. In this case, it was developed using a Houston Instrument ImageMaker plotter, which supports almost the entire command set, which are defined in HPGL, Hewlett-Packard Graphics Language. HPGL is defined in great detail in "HP 7475A Graphics Plotter - Interfacing and Programming Manual" (part no. 07475-90001), which can be found on eBay.

The library is composed of several components:

- The **PlotterDriver** component is written in native C++ to handle all communications over the serial and parallel ports. The development machine, which has no physical serial or parallel ports, uses a pair of USB cables, one for serial port communications to a DB9 connector, and the other for parallel port communications to a 50-pin Centronics connector. This project also contains the C++/CLI wrapper classes that connect the native C++ to the .NET code.
- The **PlotterBuffer** component is written entirely in C# and contains HPGL.cs, which does all the work of formatting the HPGL commands for being sent to the plotter. It also contains PlotterBuffer.cs, which is a not-yet-implemented class intended to sort all the HPGL commands by pen color to minimize the time-consuming actions of swapping pens. The last class in the project is PlotterTester.cs, which contains test code methods for exercising the HPGL.cs code.
- The last project is **PlotterTestApp**, also written entirely in C#, is the entry point for the solution, and drives the rest of the code.

PlotterTestApp calls public methods in the PlotterBuffer project to either exercise test cases, or makes calls directly to the public methods that draw some of the more complex shapes. Some examples follow:



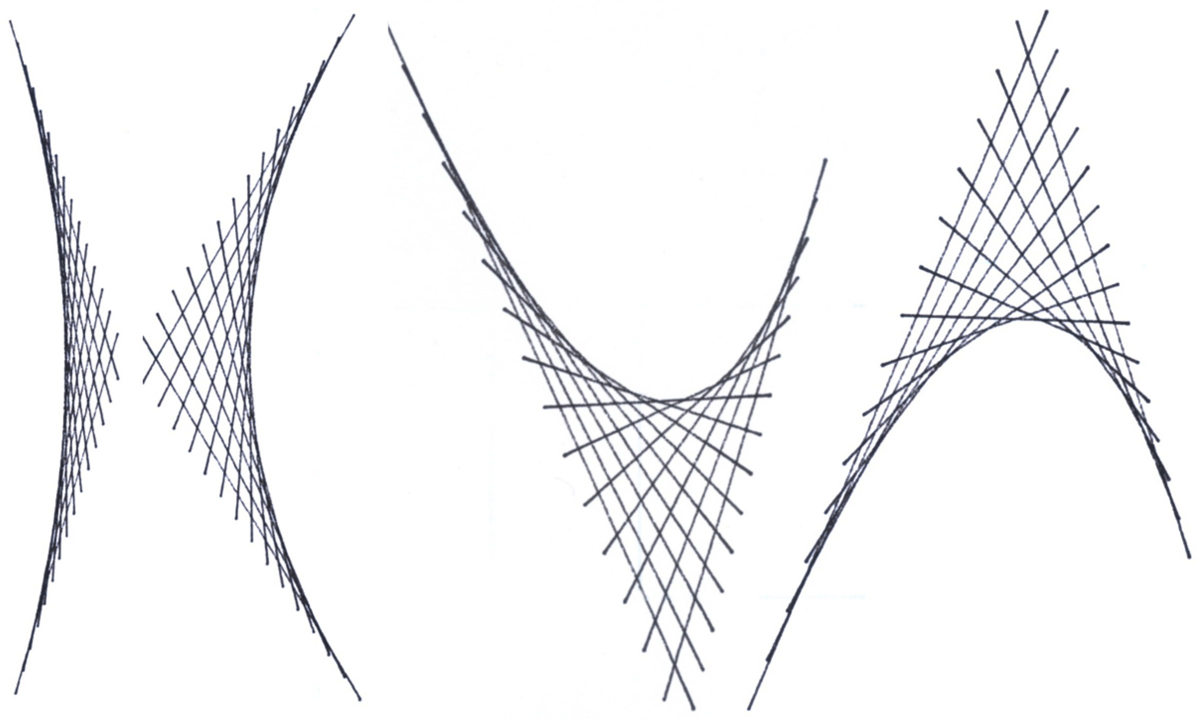
This first example was created from running 3 separate test cases in PlotterTester.cs on the same sheet.

DRAW_LINE_ART_1 drew the 5 vertical lines in green ink.

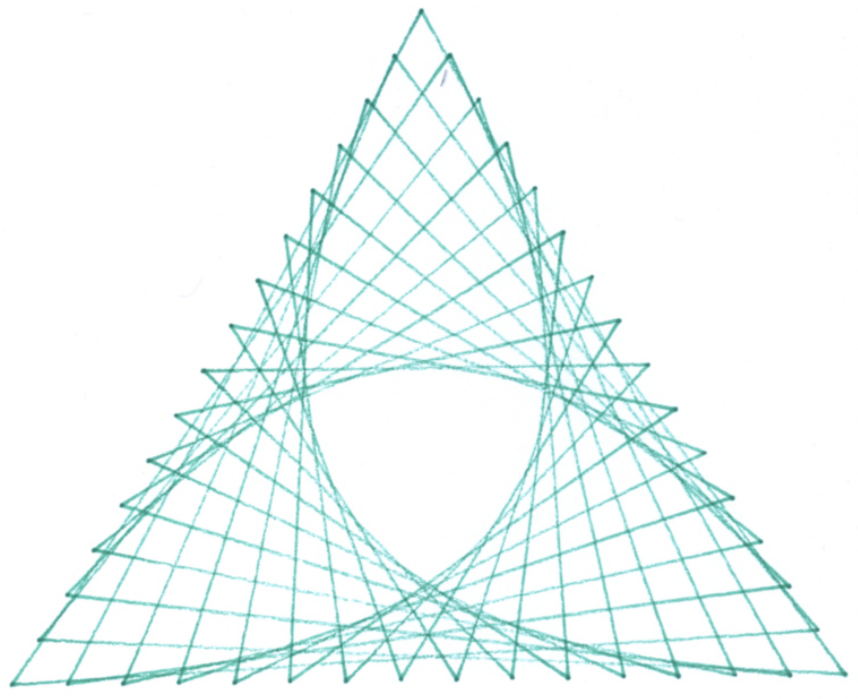
DRAW_LINE_ART_6 drew the curved-grid in black ink. This image was inspired by images of string art seen in "Star Trek" (original series) and found on the Internet.

CPlotterShapes.DrawRadialLines was used to draw the radial lines from the center to the edges in green ink.

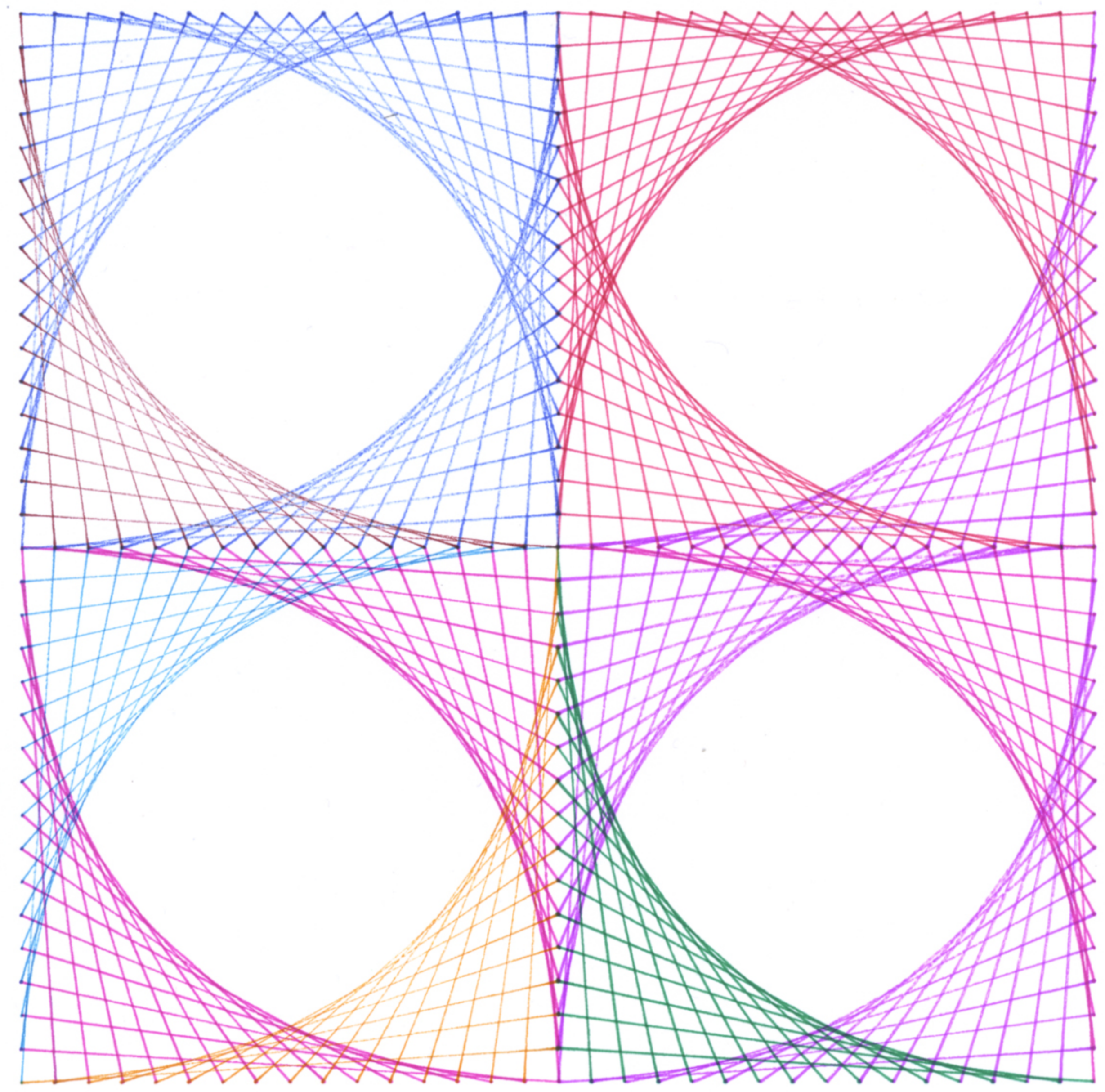
The next 4 string-art curved-grid images were created by DRAW_LINE_ART_11.



The green triangle was drawn by CPlotterShapes.DrawTriangle which drew 3 connected curved-grid images to form the equilateral triangle, in green ink.



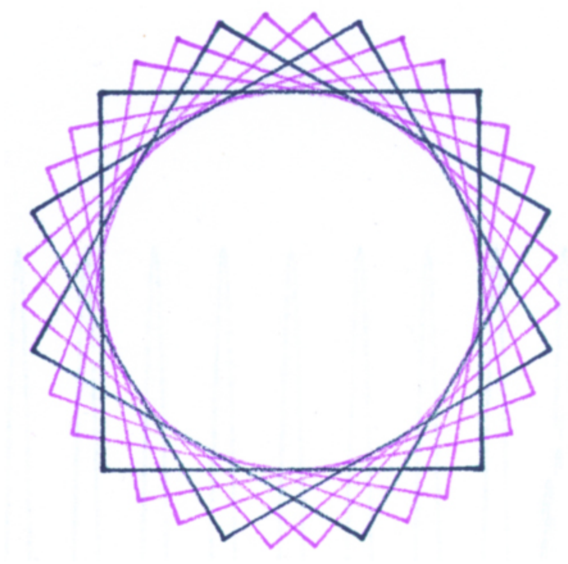
CPlotterShapes.DrawFourQuadrants used 8 different color pens randomly selected to draw this series of 16 connected curved-grid images. The image is based on a hand-drawn picture found on the Internet, having been drawn by hand by a very talented seventh-grade student.

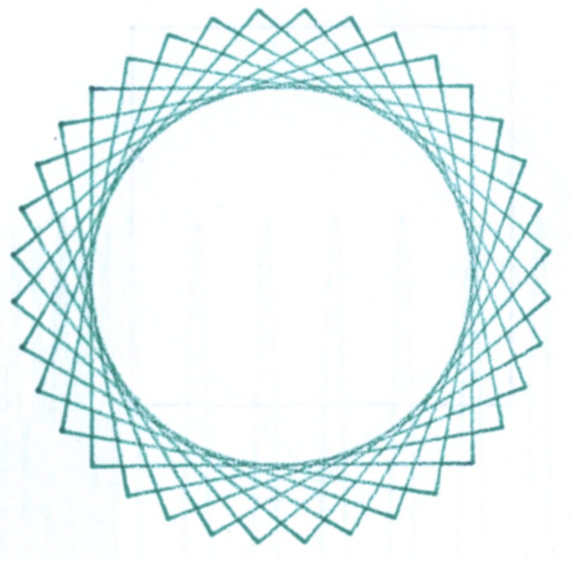


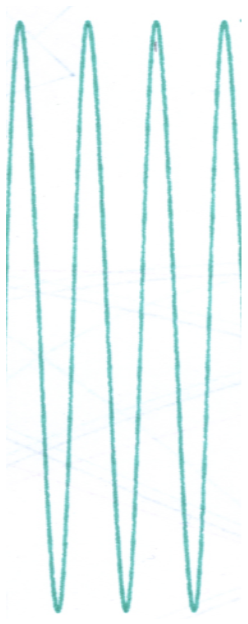
The next 2 images were drawn by the ROTATE_SQUARE test case. The image on the left is actually 2 images from 2 separate tests. The blank ink shows the original square with 2 copies of it, rotated 30 degrees and 60 degrees. The purple ink shows the same square rotated 10 degrees per rotation.

The image on the right shows the same square rotated 10 degrees per step from 0 degrees to 80 degrees. Since the shape is a square, 9 steps are all that are needed to complete the image.

This code made use of the PolarToCartesian and CartesianToPolor methods to rotate the images.







Next is the sine wave example. Like an oscilloscope, the image is drawn with a steadily increasing value in the horizontal direction, with the vertical position being determined by the sine function of the value of the horizontal position.

This image was created by the DRAW_SINE_WAVE test case.

The last 2 images are Lissajous curves, named after the French mathematician, Jules Antoine Lissajous, who studied them in depth. The sine curve above was drawn with a consistent left-to-right horizontal motion with the sine values plotted in the vertical dimension. In contrast, a Lissajous curve is drawn with sine values in both x- and y-axis. By plotting sine values of differing wavelengths, these plots become more interesting.

The next image was drawn by the DRAW_LISSAJOUS test case, using the method CPlotterShapes.PlotLissajousCurve. The test case uses 3 different pens for each of the separate wavelength combinations.

The last image was created while experimenting with other wavelength combinations, also using the CPlotterShapes.PlotLissajousCurve method.