A Curious Algorithm

Samuel Everett Grassi

August 2018

Introduction

During a lecture in a summer calculus class, I took to drawing random geometric patterns around the edges of my notes. As I was toying with different shapes and ideas for simple space filling curves, I came across an algorithm that happened to result in consistent patterns when drawn within triangles. Consider this simple algorithm below:

1. Given any triangle, P, pick a point anywhere along one of P's edges and call it `current_point`.

2. From `current_point` draw to the *farthest* edge of P, such that the line drawn is perpendicular to the chosen edge. If faced with two equal distance, pick one randomly.

3. Set `current_point` equal to the end of this drawn perpendicular line, and repeat this recursive process for some number of steps.

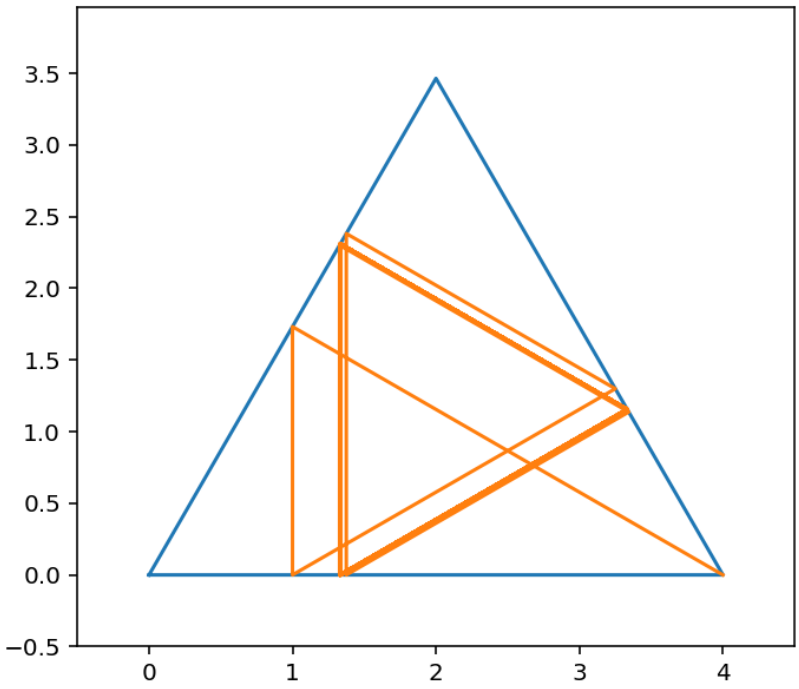
Following this "right angle algorithm" in detail on paper, I noticed a couple curious properties. The most prominent property I noticed was that given any "initial triangle" and any starting point, the algorithm would quickly converge to some repeating shape, which I will call the "ultimate shape".

Interested in finding the exact properties of this ultimate shape and putting my drawing skills to the test, I went home and wrote a program to generate a triangle and implement this right angle algorithm on it, graphing the result. This write up is a description of what I found when testing this algorithm, highlighting its more interesting features.

If you would like to see the code I wrote for this, visit the GitHub repository linked here.

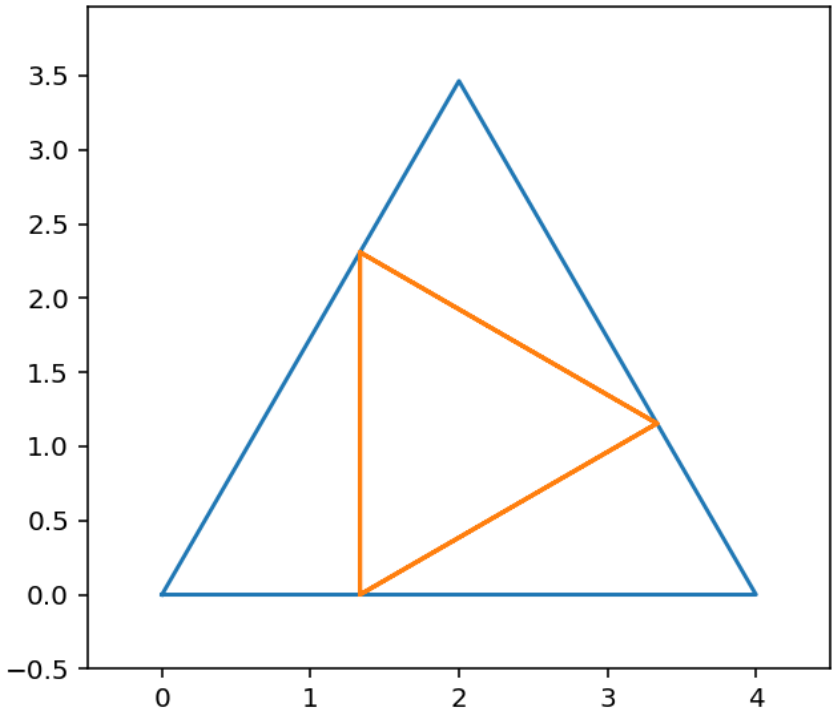
Putting the Algorithm to the Test

Picking an equilateral triangle to begin, I started the algorithm at one of its vertices and let it run for 100 steps, resulting in the following graph.



As you can see, after just a few steps the algorithm converged to a different triangle within the outer one. It can be seen that the ultimate triangle has each edge perpendicular to one of the outer triangle's edges, with all three of its vertices touching the initial triangle's edges.

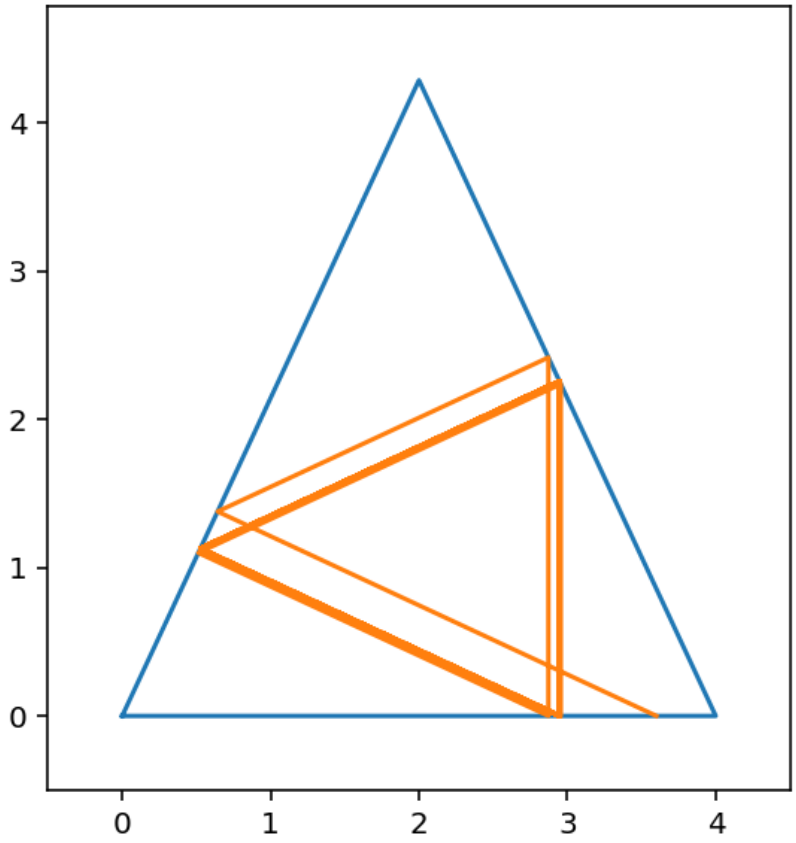
The final ultimate shape is shown below.



To eek some more information out of this pattern, I wrote a function to return the inner angles of this ultimate triangle. It turns out that this ultimate triangle is similar to the initial triangle. In this case, the inner angles of both triangles are $60 - 60 - 60$. This is a rather interesting result.

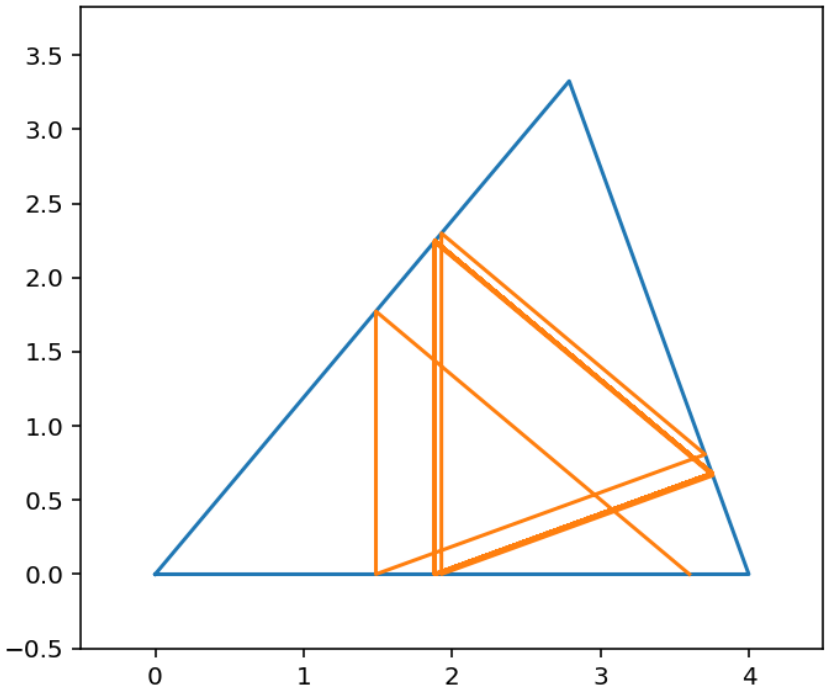
As the initial triangle is scaled up and down, the convergence of the algorithm is not affected; the ultimate shape is always of the same kind (a triangle in this case). However, as the starting point of the algorithm is changed, the orientation of the ultimate triangle changes; in fact the ultimate triangle may "flip" around the axis defined by $x = 2$. Despite a possible internal orientation change wrought by changing the starting point of the algorithm, the inner ultimate triangle is always the same size, with the same interior angles.

Applying the algorithm to an isosceles triangle with interior angles $65 - 65 - 50$ we get a similar result.



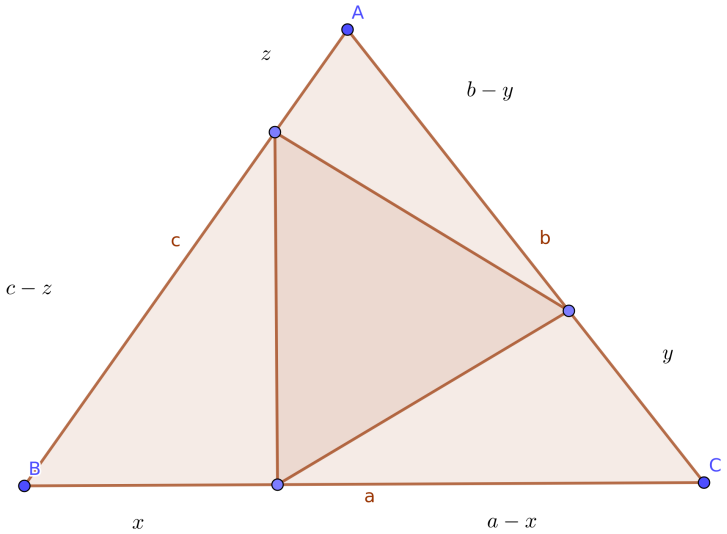
As you can see, the same thing happened for this particular isosceles triangle; using an information program to analyze the ultimate shape, it is found that the ultimate triangle the algorithm converged to is similar to the initial triangle.

At this point, the reader may discern that the ultimate shape may not always be a triangle, depending on the properties of the initial (outer) triangle. I will get to that, but first I am going to show that this property of convergence to similar triangles also holds for scalene triangles.



Once again, it is found that the ultimate triangle is similar to the initial triangle. One natural question may be, "If the computational resources are not available to find this ultimate triangle, how else might it be found exactly?".

The answer would be that this exact inner triangle could be found with relative ease using geometry. Take a look at the *general* diagram below for one of these cases.



We know a, b, c, and A, B, C as shown on the diagram. It is also known that each of the edges of the inner triangle will be perpendicular with its corresponding edge on the outer triangle, and that the inner ultimate triangle will be similar to the outer one (same interior angles). With this information, all the inner angles can be found, giving enough information to solve for x, y, and z shown in the diagram above. After finding these values, the points of the inner ultimate triangle can be found.

One interesting formula that can be used to *approximate* the side lengths of the inner ultimate triangle is given by

$$a' \approx \frac{sin(\gamma) \cdot a}{\varphi}$$

where γ is the peak angle of the initial triangle, a' is the inner side length to be approximated, a is the corresponding outer side length, and $\varphi = 1.61803\cdots$, the Golden Ratio. The resulting error of the approximations when using this formula is highly dependent on the triangle being analyzed. In the example triangles I worked on, I saw side length approximations errors of the ultimate inner triangle between 0.1% and 8%.

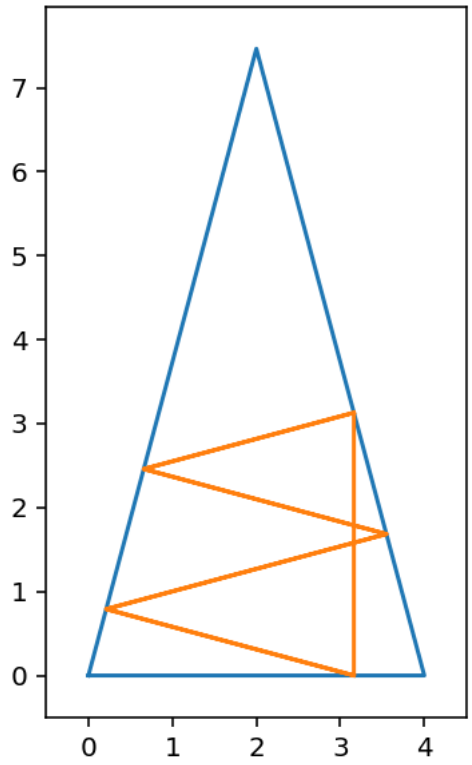
The side length to be solved for, a', can be found exactly with

$$a' = \frac{sin(\gamma) \cdot a}{c}$$

where $c \approx \varphi$. I have a hunch c is given by a function of the peak angle γ, however I have yet to find this function exactly, assuming it exists.

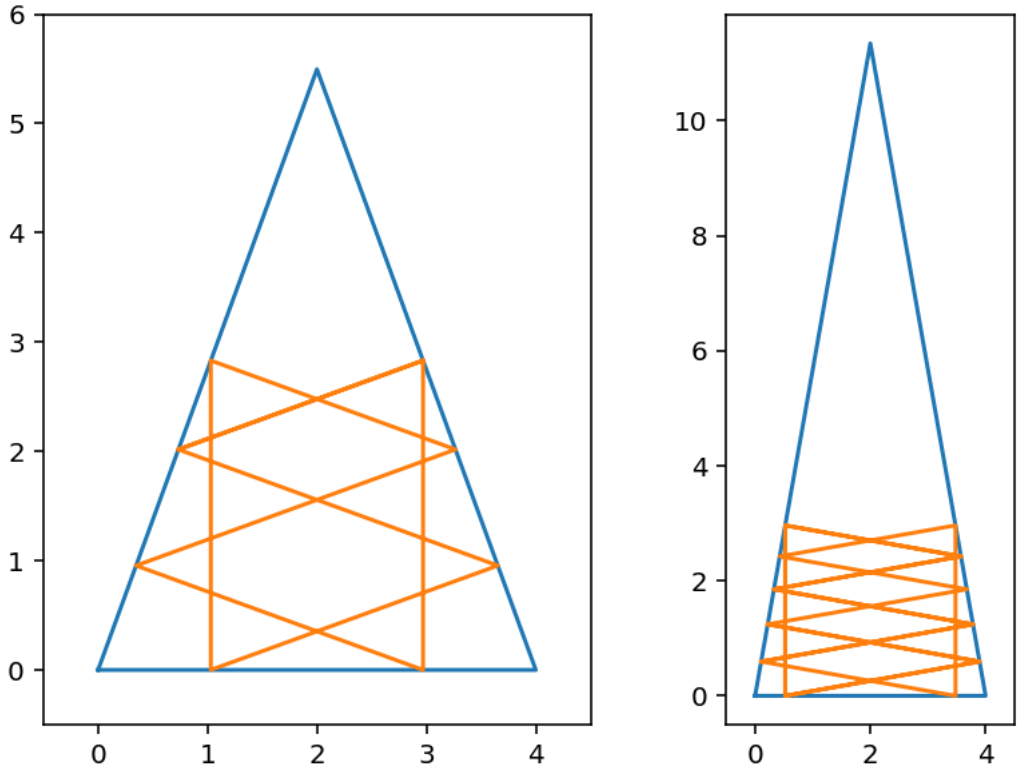
Other Ultimate Shapes

As alluded to earlier, when the angles of the outer triangle are adjusted, the algorithm can generate other interesting repeating shapes. One of the simpler repeating shapes is given below.



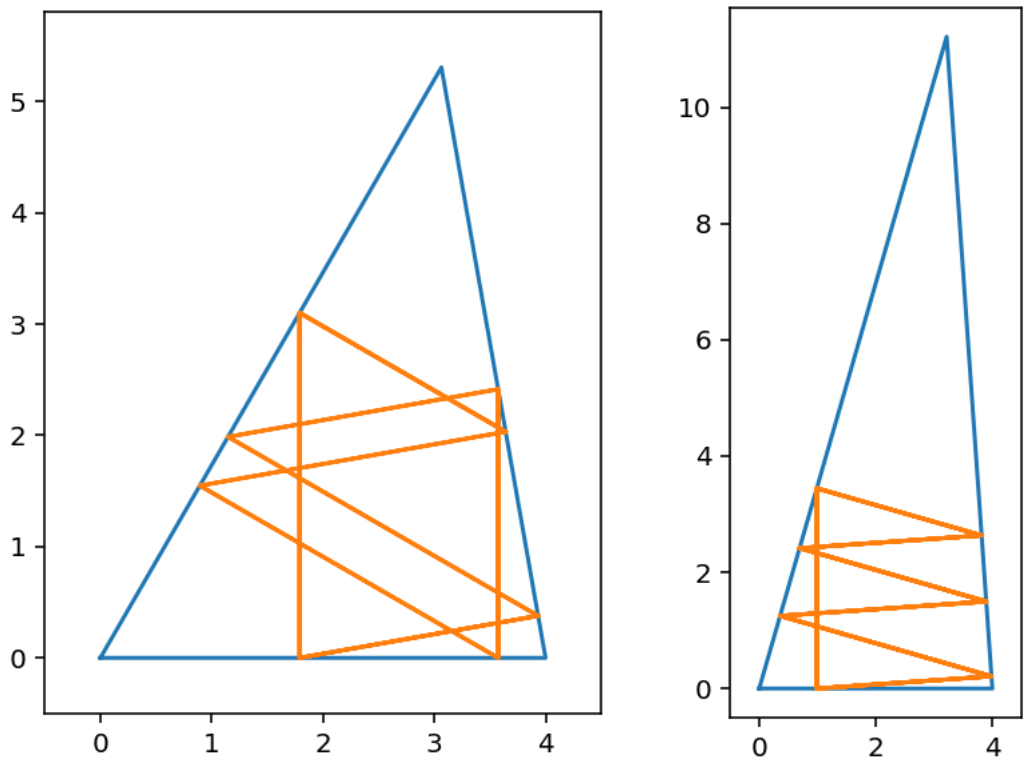
This is the first repeating ultimate shape we have seen that produces something other than a triangle. This ultimate shape is nothing to get too excited about however, so lets start by slightly increasing and decreasing the interior angles of the isosceles triangle to see what else the algorithm comes

up with.

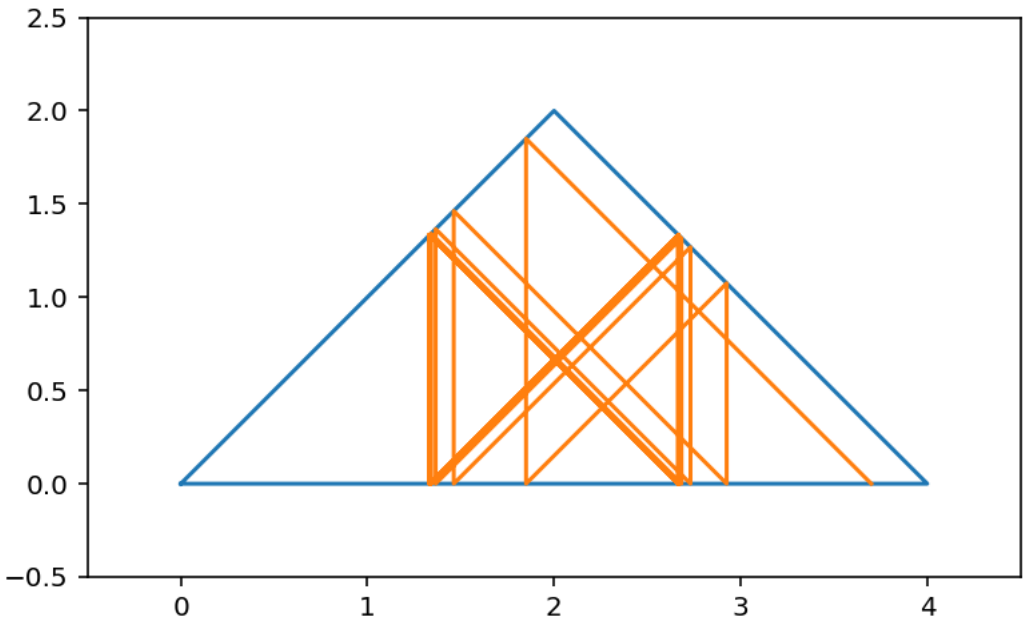


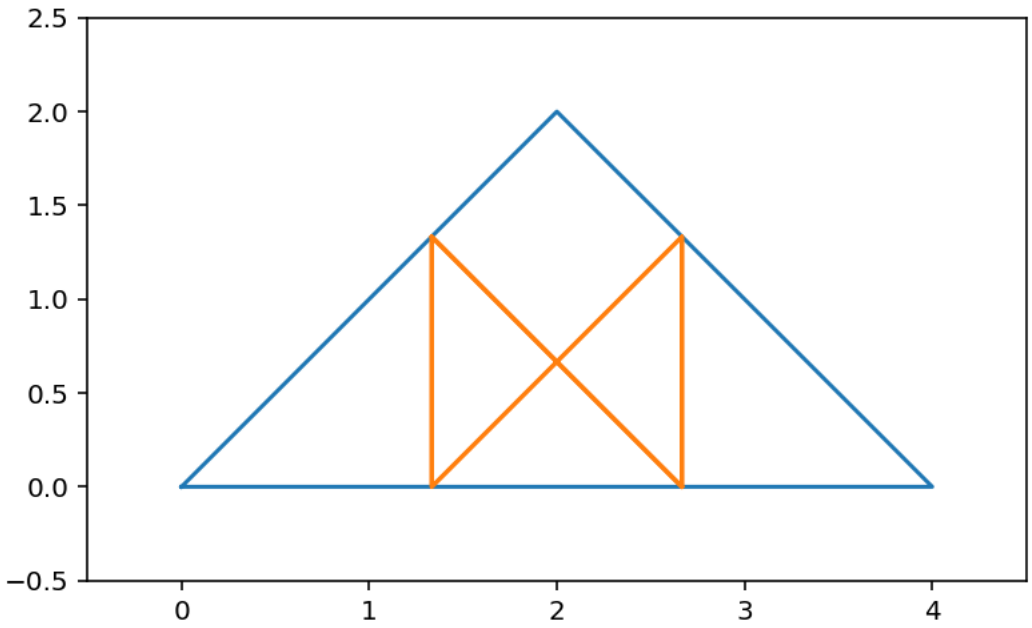
The patterns illustrated above are far more interesting. Recall that the algorithm will always draw the longest possible line it can while still obeying the right angle rule, as we can see the result of this taking form in the previous two examples. As the slope of the sides becomes steeper, the number of steps up the algorithm will take increases.

As shown below, scalene triangles succumb to a similar phenomena as well, producing different repeating shapes.



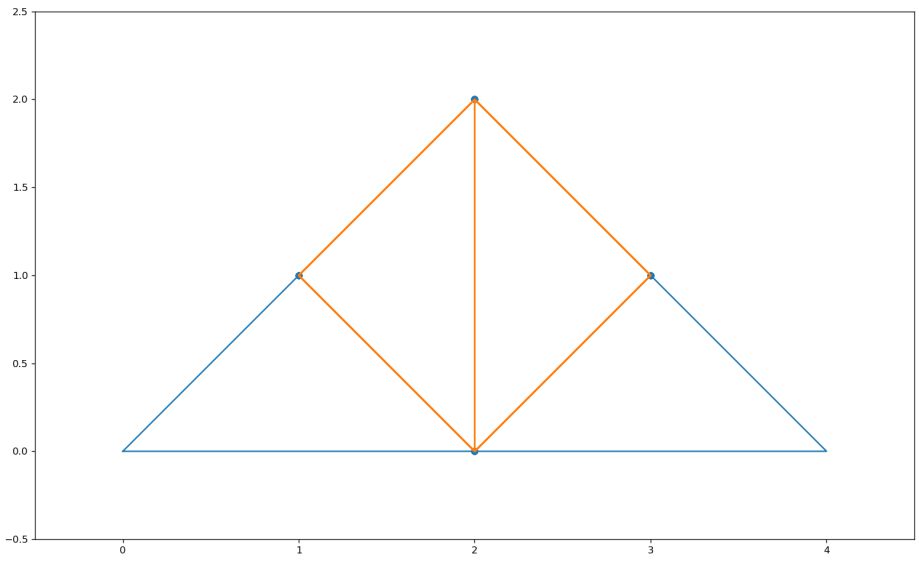
When we move the top angle in the other direction, starting with a $45 - 90 - 45$ triangle, the algorithm produces the other interesting pattern plotted below. The starting point is $(3.7, 0)$.


2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4

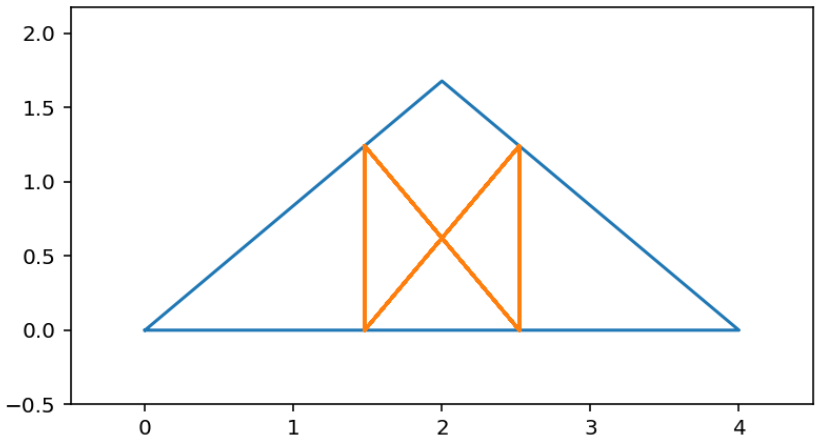

2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4

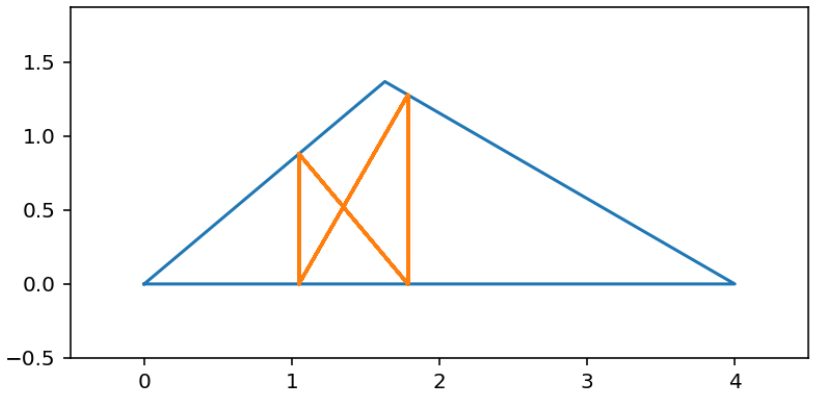
The most immediately noticeable result is the “hour-glass” ultimate shape that forms. In fact, this ultimate shape is built out of two $45-90-45$ triangles, as found through an information function.

In the current implementation of the algorithm I have prevented it from drawing along edges or meeting the triangles vertices, to force it into developing the hourglass ultimate shape seen above. It is for this reason that on what looks to be the fourth step above (shown on the first graph of the triangle), the algorithm chose to draw down instead of up to the top vertices of the triangle. If the algorithm had been allowed to do so, the resulting inner shape would have been as pictured below.



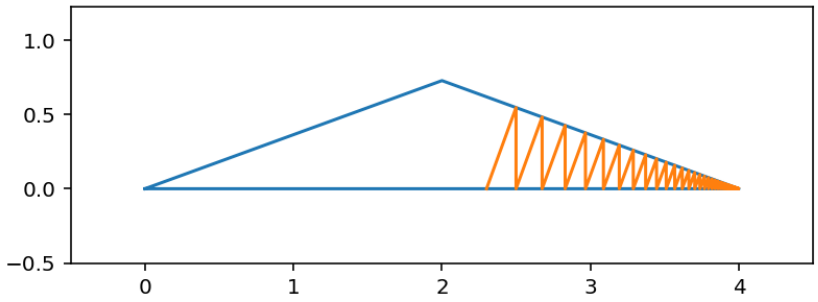
The algorithm is applied to a few more obtuse triangles to demonstrate this property of the algorithm converging to an ”hour-glass” ultimate shape, composed of two similar triangles.





While this property is true for all obtuse triangles, note what happens as γ, the peak angle, approaches 180 degrees. As $\gamma \to 180$, the range of starting points from which the algorithm will converge to this ultimate hour-glass shape decreases, approaching the center of the triangle.

Notice what happens if the algorithm starts outside of this "critical range".

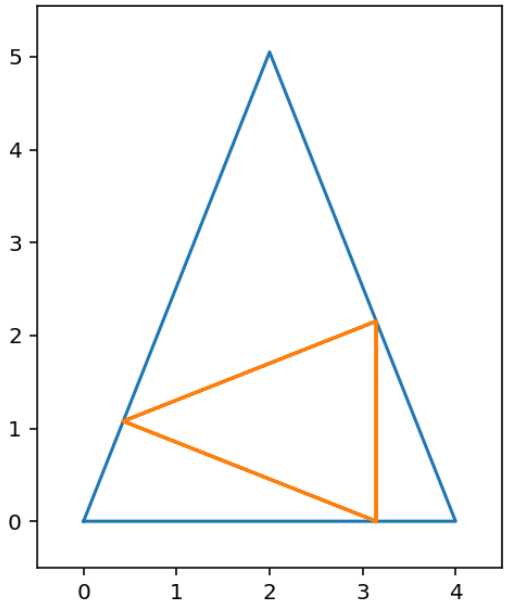


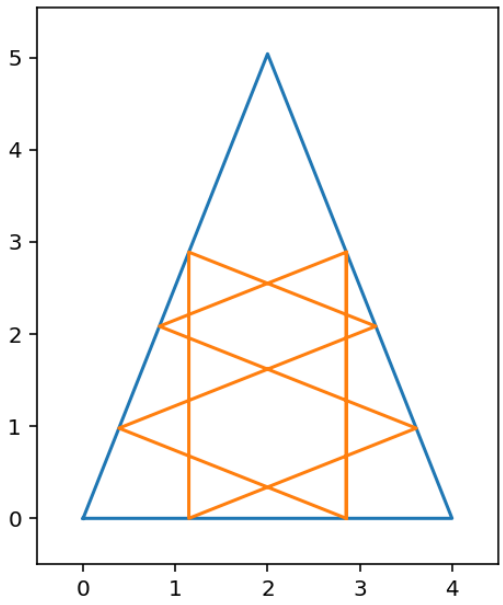
The algorithm gets "stuck" in a corner and approaches the left/right endpoints as the number of steps approaches infinity.

Finding the Ultimate Triangle

As described in this article, the ultimate shape of different initial triangles may or may not be a triangle. An important question is "can we define a set of angles for each type of triangle, where the algorithm will always converge to an ultimate triangle?".

Lets begin with finding rules for when an isosceles triangle will or will not result in an ultimate triangle. Consider the two initial isosceles triangles below. Notice the minute difference in their interior angles (about .1 degrees), and the large difference in the ultimate shapes.





What we have found is a "critical point" where the algorithm will converge to something other than a triangle. We already know of the first critical point for isosceles triangles; it is the $45-90-45$ triangle, where when the angles are at that point the ultimate shape will not be a triangle. The graphs above show us that the inflection point on the other end is when the interior angles are about $68.35-43.3-68.35$. Now that we know the two critical points, we can begin to define a range on which the initial triangles will have triangular ultimate shapes.

We will define each of the "foot" angles in an isosceles triangle as x, and the "peak" angle as y. From this we deduce the following rule for the interior angles of isosceles triangles

$$2x + y = 180$$

Treating y as an unknown, we can solve for it and graph the equation

$$y = -2x + 180$$

Using this function and the found critical points, we know that when $45 < x < 68.35$, and x is passed through the function $y = -2x + 180$ the returned triangle defined by x and y will always result in an isosceles triangle with

a triangular ultimate shape. When $x \leq 45$ and $x \geq 68.35$, the triangle produced with the angles given by the function $y = -2x + 180$ will have an ultimate shape equal to something *other than* a triangle.

A similar process can be used to find the subset of angles that define a scalene triangle with an ultimate shape equal to a triangle. For example, if we take the line perpendicular to $y = -2x + 180$ that passes through $(60, 60)$ we get $y = \frac{1}{2}x + 30$. The points along this line give interior angles that can be used to construct scalene triangles, except when $x = 60$.

Now, a pair of critical points can be found for scalene triangles to bound the range of points along the line defined by $y = \frac{1}{2}x + 30$. Then, we know that the scalene triangles within this range will have triangular ultimate shapes.

We now have an easy method for determining if a given initial triangle will have a triangular ultimate shape or not.

Given these findings, we are in a position to make some simple conjectures and explore other questions.

Conjecture

For any given triangle P, there *may* exist *a single* similar triangle P', such that when P' is placed within P, P' can touch all three of P's edges, and will form a right angle with each of P's edges. Such a triangle P' does not exist for triangles with an angle greater than or equal to 90 degrees, and does not exist for specific acute triangles. A P' does exist however for every equilateral triangle.

Perhaps the best or only way of testing this conjecture is by using this "right angle algorithm" on a triangle P to see if such a triangle P' can be found.

Further Exploration

This project brought up many more ideas I would like to explore and questions I would like to have answered. I list some of these below.

1. Is it possible to create chosen shapes or patterns by passing the some unique engineered polygon to the algorithm?

2. How would this algorithm respond when applied to higher dimensional shapes? For example, given some 3D shape, and applying some form

of this algorithm to it, will other repeating shapes emerge? Other repeating shapes similar to the "outer" 3D shape?

3. In the cases when the algorithm does converge to a triangle, why does it do so? Specifically, what is special about the "molding" characteristics of certain initial triangles that force the ultimate shape into some specific repeating pattern?

4. Do the ranges I define along the lines describe all the isosceles/scalene triangles that have triangular ultimate shapes?

5. Is there anything particularly special about the equilateral triangle and its predictable ultimate shape? Or is the equilateral triangle just a triangle that *happens* to have a triangular ultimate shape?

6. Is there anything particularly special about the critical points that bound the ranges?

7. One of the next things I would like to explore is what kind of repeating shapes, if any, this algorithm would produce on any polygon other than a triangle. For every polygon, does there exists at least one repeating shape that can be developed by applying this algorithm to it?

8. I would like to continue searching for an exact formula that can be used to find the side lengths of any interior ultimate triangle given the initial triangle (assuming a formula of this kind exists).

9. A fun project I am currently working on is a second program that can be used to recursively nest these "ultimate triangles" within each-other, to create some rather visually appealing nested patterns.