Code journal: Geometric Spiral

21.07.2020, Mariana Ávalos Arce

What is constant and what is not?

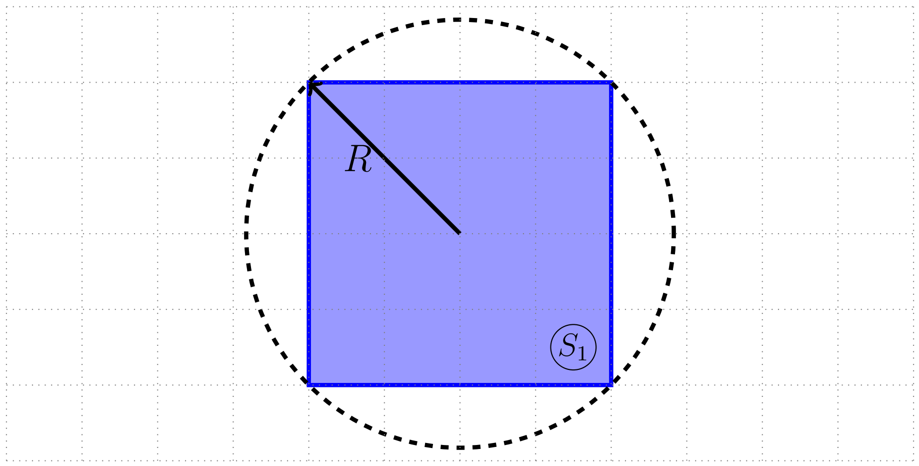
A **Geometric Spiral** is basically the well-known image where a regular polygon rotates and gets bigger and bigger as the image is filled. The image might look deceitful at times, which made me doubt: are both the angle and the size increasing, or just one of them?

A regular polygon

The first thing to build the image is to create the central **regular polygon**. To build any regular polygon, the whole 360 degrees must be divided by the number of sides in the desired polygon. During this report, the example will be done with a square. Therefore,

$$\text{step angle} = \frac{360}{sides} = \frac{360}{4} = 90\,\text{deg} \tag{1}$$

Which tells us that every 90 degrees a polygon vertex is positioned. Then, we just need a custom radius or polygon size to draw square 1 (S_1) inside a circumference of radius R (or size), following the idea below. A simplified code for the mentioned concept is also shown further below.



```
acc_angle = 0.0; acc_size = size
last_point = [0.0, 0.0]; first_point = [0.0, 0.0]
    for i in range(num_sides):
    angle = i * step_angle - (step_angle / 2.0) - acc_angle
    x = fig_center[0] + (acc_size) * math.cos(angle * PI / 180.0)
    y = fig_center[1] + (acc_size) * math.sin(angle * PI / 180.0)
    if i == 0:
            first_point = [x, y]
    if i > 0:
            ln.line([x, y], last_point, thick, color, img)
    if i == num_sides - 1:
            ln.line([x, y], first_point, thick, color, img)
    last_point = [x, y]
```

Resulting in the first square at the center of the image, which will be also the smallest. The following iterations of squares in an external loop answer the first question: what is constant, the angle or the size increase in each square? If we look at the figure below, the blue lines are the Δa or the slight change in the rotation angle of each square, and we see that **each square follows a constant angle increase** when compared to the previous one.

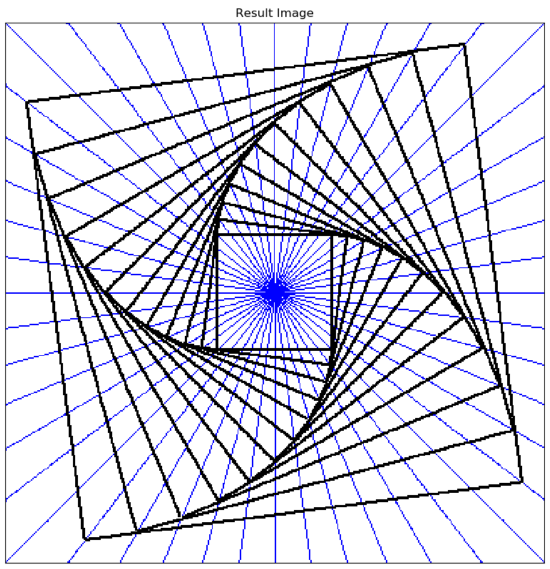
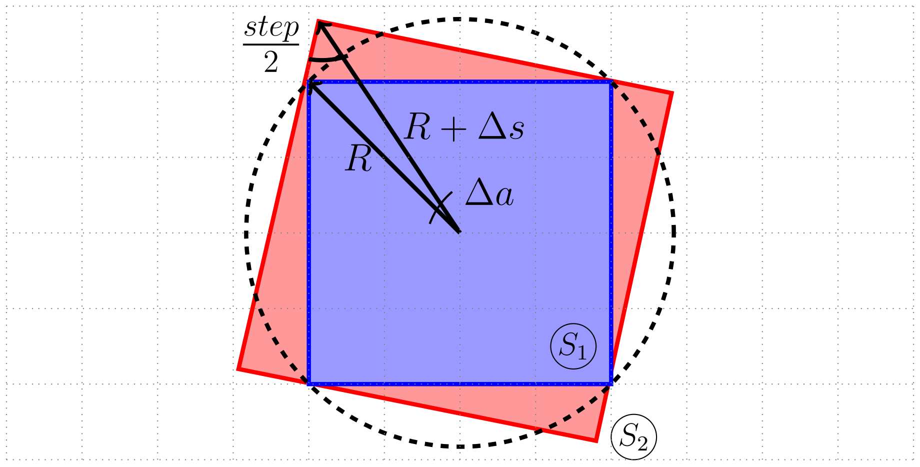


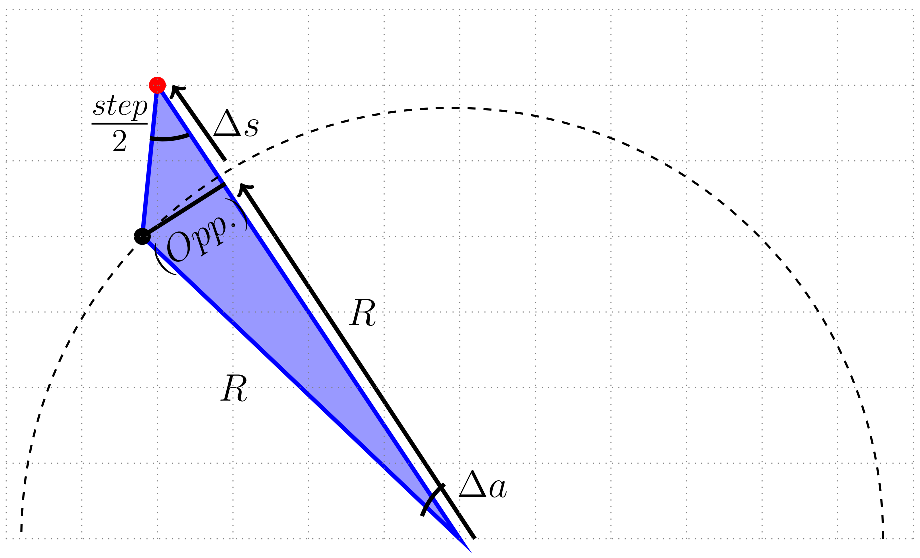
Figure 1: Test showing constant angle

The test image also shows that each of the blue lines slides the `step_angle` in half, which will be crucial to calculate the **increase in size** or Δs of each square so that its sides exactly coincide with the last square's sides. This increase in size (or radius) will then be **larger** in every polygon drawn. To explain this, let's take an example of a two-square spiral, as in the diagram below.



Thus, in order to calculate the exact change in size *for the next polygon* so that the sides touch each other exactly, we must take on account that we know the value of two crucial angles: the custom Δa and the angle $\frac{step}{2}$. If we take the triangle and amplify it a bit as in the following diagram, we get a more evident discovery to know the value of Δs, and with this, the red point will be given now that the new accumulated radius or size gets increased with the new value;

allowing the value of the next polygon to be known at the *end* of each iteration.



Thus, taking into account some trigonometry,

$$\text{Opp.} = \text{size} \times sin(\Delta a), \tag{2}$$

Which leads to

$$\frac{tan(\frac{step}{2})}{1} = \frac{Opp.}{Adj.}, \tag{3}$$

$$\frac{Adj.}{Opp.} = \frac{1}{tan(\frac{step}{2})} \tag{4}$$

$$\boxed{Adj. = \frac{Opp.}{tan(\frac{step}{2})} = \frac{\text{size} \times sin(\Delta a)}{tan(\frac{step}{2})}} \tag{5}$$

Indicating that

$$\Delta s = |\text{size} - Adj.| \tag{6}$$

So now we can complete the code with Δs (ca) as the following:

```
for j in range(num_iter):
    last_point = [0.0, 0.0]
    first_point = [0.0, 0.0]
    for i in range(num_sides):
        angle = i * step_angle - (step_angle / 2.0) - acc_angle
        x = fig_center[0] + (acc_size) * math.cos(angle * PI / 180.0)
        y = fig_center[1] + (acc_size) * math.sin(angle * PI / 180.0)
        if i == 0:
            first_point = [x, y]
        if i > 0:
            ln.line([x, y], last_point, thick, color, img)
        if i == num_sides - 1:
            ln.line([x, y], first_point, thick, color, img)
        last_point = [x, y]
        ca = (acc_size * math.sin(delta_angle * PI / 180.0)) / (math.tan((step_angle / 2.0) * PI / 180.0))
        ca = abs(acc_size - ca)
    acc_angle += delta_angle
    acc_size += abs(acc_size - ca)
```

This sample code creates a number of iterations where each makes a square slightly rotated by a custom `delta angle` that remains constant during all the iterations. Also, each square or regular polygon should grow its size or radius, but not by a constant number each. The increase in size of the polygons should obey the relation in Eq.(5) and, more precisely, accumulate the size with Eq.(6) at the end of every polygon drawing. The result of the program in Python 3.6.2 is the image shown below.

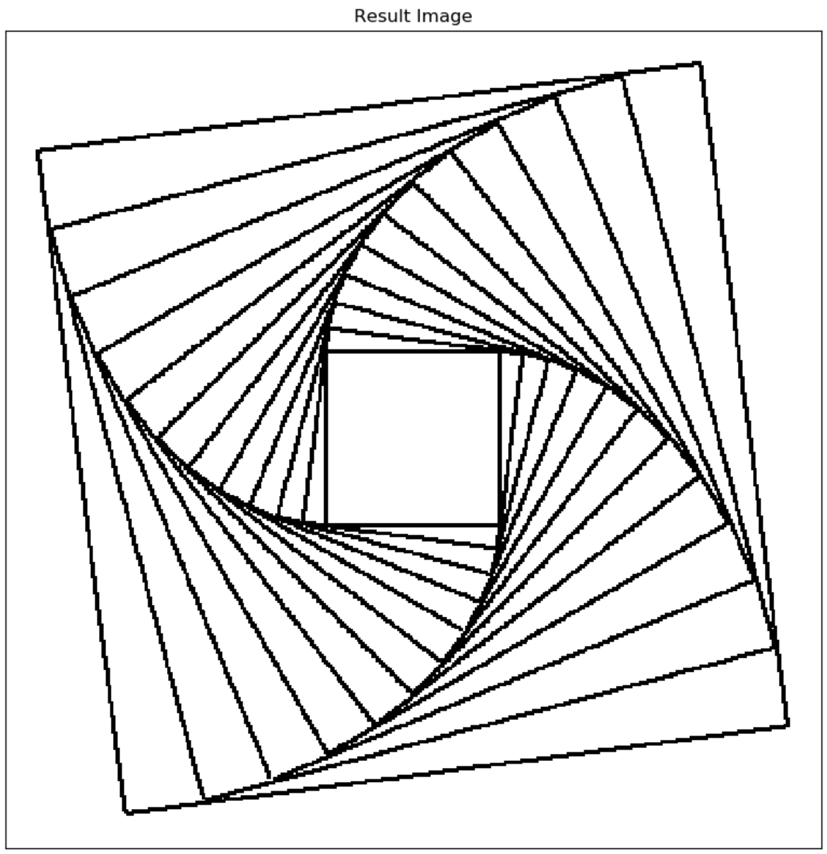


Figure 2: Test showing constant angle