(Circle + Square)/Triangle: Using Shape-Based Expressions for Image Creation and Exploration

John Nicholson

Austin Peay State University, Clarksville, TN, USA; nicholsonja@apsu.edu

Abstract

This paper describes a method for evaluating simple math expressions in which the operands are shapes, lines, and curves. The operands can be either stationary or placed in motion through rotation and translation operations. The results of these expressions can be evaluated with the aid of software, allowing them to be rendered as both static images and animations.

Shape-Based Expressions

In basic expressions such as

$$z = i + j \tag{1}$$

values can be assigned to i and j so that a value for z can be calculated. The work described here is based on a simple question: what would the value of z be if i and j were shapes? For example, if i was a square and j was a circle, then in the expression

$$z = \square + \bigcirc \tag{2}$$

what would be the value of z? With this question as a starting point, I developed a method to evaluate simple expressions with shapes as the operands and then implemented software that can render images created from the results of the shape-based expressions.

The first step is to define shape S as any 2-dimensional shape, curve, or line that has the center S_C and that can be expressed with parametric equations $S_x^f()$ and $S_y^f()$, which are used to calculate S's coordinates based on θ:

$$\begin{aligned} S_x(\theta) &= S_x^f(\theta) \\ S_y(\theta) &= S_y^f(\theta) \\ S_C &= (S_{cx}, S_{cy}) \end{aligned} \tag{3}$$

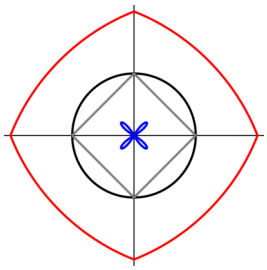
Next is to define a general version of equation (1):

$$Z = I \ op \ J \tag{4}$$

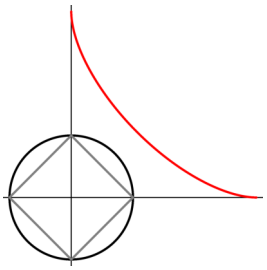
where I and J are shapes, op is an operator such as + or $*$, and Z is the shape resulting from the expression. Applying the equations in (3) to equation (4) leads to

$$\begin{aligned} Z_x(\theta) &= I_x(\theta) \ op \ J_x(\theta) \\ Z_y(\theta) &= I_y(\theta) \ op \ J_y(\theta) \\ Z_C &= (I_{cx} \ op \ J_{cx} \ , \ I_{cy} \ op \ J_{cy}) \end{aligned} \tag{5}$$

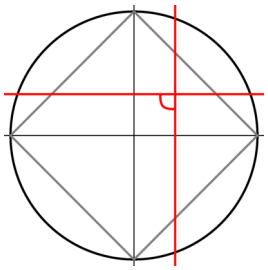
Although not expressed in the equations, the software also contains a rotation function that rotates a shape around its center. Examples of the addition, subtraction, multiplication, and division operations on a circle



(a) *$A + B$ (red outer shape) and $A - B$ (blue inner shape).*



(b) *$A * B$ (red curve).*



(c) *A/B (set of red lines).*

Figure 1: *Results of basic operations on square A and circle B where B's radius is 3, A is inscribed in B, and both are centered at the origin.*

and a square are shown in Figure 1. As the figures show, the results can be closed curves (Figure 1(a)) or open curves (Figure 1(b)). Division (Figure 1(c)) may result in a division by 0 when parts of the shape operand in the denominator intersect the x-axis or y-axis. In these cases, values in Z move towards infinity and result in lines going beyond the "interesting" part of the image, as represented by the horizontal and vertical lines in the figure.

Using these basic concepts, any number of shapes and types of shapes can be combined using any set of operators. For example, Figure 2 shows the result of the expression

$$Z = octagon - circle - hexagon + pentagon + pentagon - square - triangle - triangle \tag{6}$$

where all shapes are inscribed in the $circle$, similar to the square and circle in Figure 1. Figure 3 shows the result of multiplying two more complex curves.

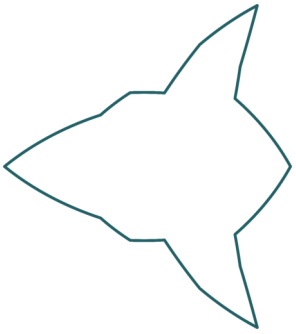
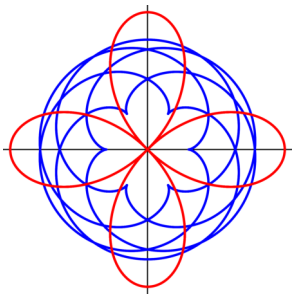
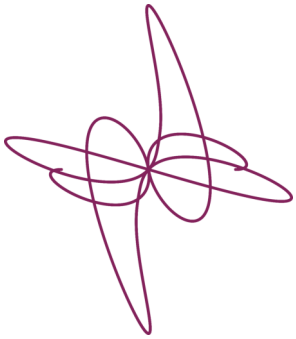


Figure 2: *Spaceship-like result of adding and subtracting multiple shapes.*



(a) *Epicycloid (blue) and rose curve (red) centered at the origin.*

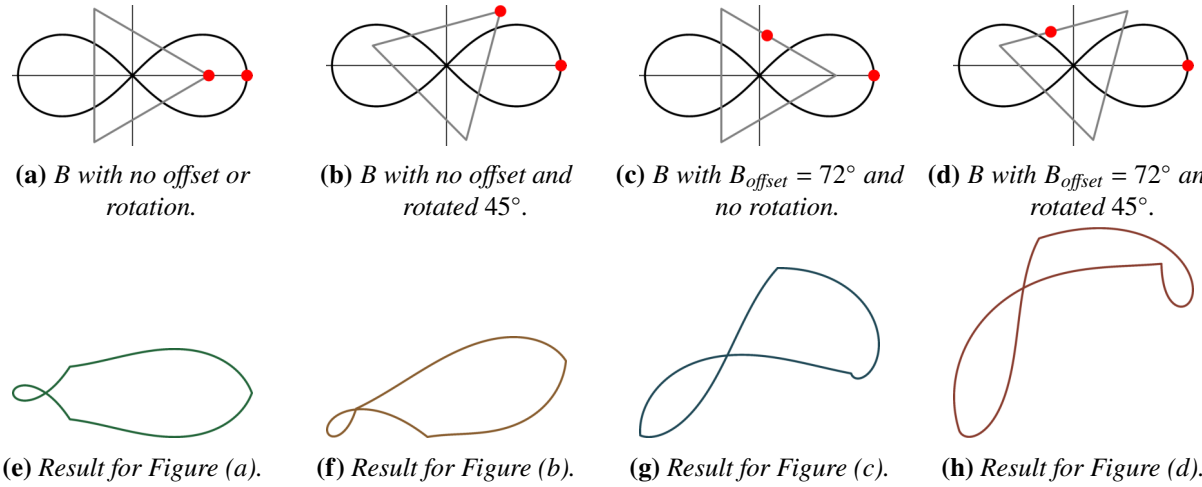


(b) *epicycloid $*$ rose curve.*

Figure 3: *Multiplying more complex curves.*

Since this project is software-based, drawing a shape requires starting at some point on the curve, usually at $\theta = 0$. The initial point can be changed by adding a starting offset S_{offset} to θ, which affects how the shape interacts with the other shapes in the expression. Adding offsets for I and J to the equations in (5) leads to

$$\begin{aligned} Z_x(\theta) &= I_x(\theta + I_{offset}) \ op \ J_x(\theta + J_{offset}) \\ Z_y(\theta) &= I_y(\theta + I_{offset}) \ op \ J_y(\theta + J_{offset}) \\ Z_C &= (I_{cx} \ op \ J_{cx} \ , \ I_{cy} \ op \ J_{cy}) \end{aligned} \tag{7}$$



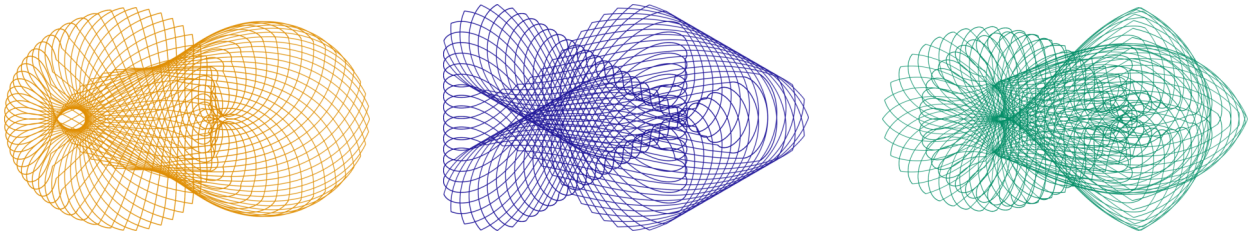
(a) *B with no offset or rotation.* **(b)** *B with no offset and rotated* 45°. **(c)** *B with* B_{offset} = 72° *and no rotation.* **(d)** *B with* B_{offset} = 72° *and rotated* 45°.

(e) *Result for Figure (a).* **(f)** *Result for Figure (b).* **(g)** *Result for Figure (c).* **(h)** *Result for Figure (d).*

Figure 4: *$Z = A + B$ where A is a lemniscate and B is a triangle. The results in the second row vary depending on B's starting offset location (red dots) and rotation in the first row.*

Motion

Once starting offsets and rotations were implemented, as demonstrated in Figure 4, it became apparent that motion could be used to create more complex images, both static and animated. A shape's starting offset could move along the entire path of a shape, and shapes could make complete rotations from 0° to 360°. Combining the results at multiple offset positions and rotation angles created interesting results. Unlike the results in Figure 4 that were computed using single rotation angles and offset positions, Figure 5 shows how moving the offset, rotating a shape, and combining both movements result in different images when results from multiple discrete steps are combined into a final image.

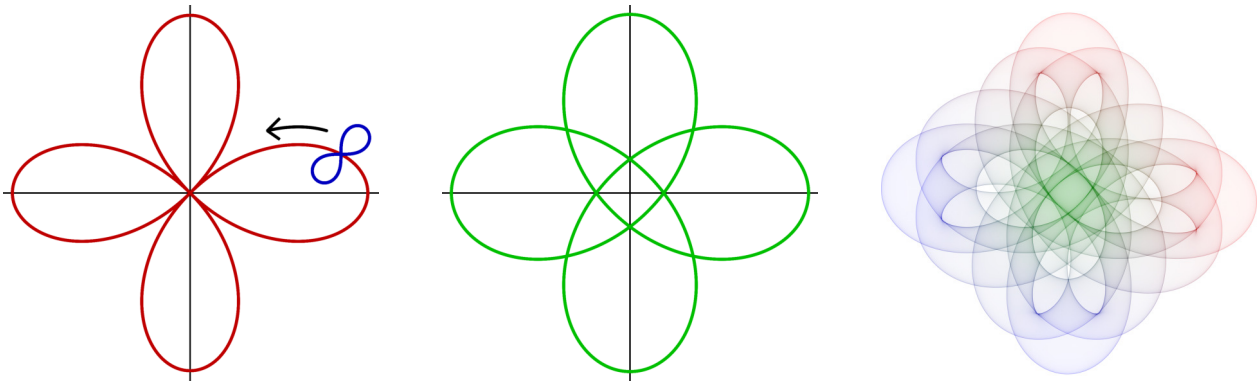


(a) *B rotates around the origin. See Figure 4(b).* **(b)** *B's offset moves around its perimeter. See Figure 4(c).* **(c)** *B rotates around the origin while its offset also moves. See Figure 4(d).*

Figure 5: *Results for shapes in Figure 4 using 40 discrete steps for offset motion and rotation.*

Although all shapes in the examples up to this point have been centered at $(0, 0)$, this is not a requirement and shapes can be centered at any coordinate. If a shape can be located anywhere, it is a simple extension to move a shape along a path P^S, which is another shape itself. For example, a circle could move along the path defined by a square. Applying this to Z_C in the equation set (7) means that each operand can have its own unique path, P^I and P^J.

$$\begin{aligned} Z_C = &((I_{cx} + P^I_x(\theta))\ op\ (J_{cx} + P^J_x(\theta))\ , \\ &(I_{cy} + P^I_y(\theta))\ op\ (J_{cy} + P^J_y(\theta))) \end{aligned} \tag{8}$$



(a) *A lemniscate moving along a path defined by a rose curve.*

(b) *Hypotrochoid.*

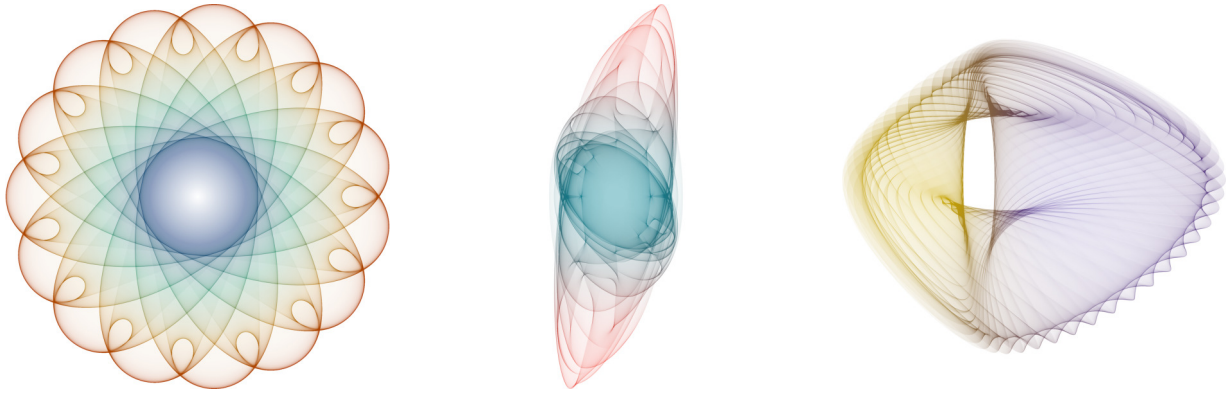
(c) *lemniscate + hypotrochoid.*

Figure 6: *Example of adding a moving shape, a lemniscate, to a static shape, a hypotrochoid.*

Figure 6 shows an example of an image that can be created when one of the operands is moving along a path. The result in Figure 6(c) was rendered using a density-plot approach similar to the one described in [1].

Conclusion

Treating curves and shapes as operands in math expressions has allowed me to explore and create new mathematical-based images. Figure 7 shows some results, including one possible rendering of the paper's title in Figure 7(b). Higher resolution versions of these images, as well as additional images, are available in the paper's supplementary materials. The next step in this project is to develop a web-based interface for exploring these expressions, images, and animations.



(a) *epitrochoid + hypotrochoid.*

(b) $(circle + square)/triangle$.

(c) $hypotrochoid * oval$.

Figure 7: *Examples created from various shape expressions.*

References

[1] J. Nicholson. "Curve Stitching Density Plots." *Bridges Conference Proceedings*, Linz, Austria, July 16–20, 2019, pp.351–354. https://archive.bridgesmathart.org/2019/bridges2019-351.html.