Symmetry and Trigonometry

Steve Whealton
1725 Kilbourne Place, NW
Washington, DC, 20010, USA
E-mail: swhealton@ctsmd.com

Abstract

The focus of this paper is on the use of trigonometric functions, specifically the arctangent and arcsine functions, to produce an overall design that suggests mirror symmetry, but is not fully symmetrical. The formula is shown, and discussed.

1. Using Trigonometric Functions

1.1. Symmetry and Scalability. When making a bitmap image using trigonometric functions, the most important preliminary questions have to do with symmetry and scale. Symmetries are built into each of the familiar trigonometric functions, and these must be accommodated. Likewise, the numbers going into and coming out of the main formula must be scaled properly.

1.2. Imperfect Symmetry. To remain largely, but incompletely symmetrical, an image must feature formulas that give mostly symmetrical results within the domain and range considered. In this image, the square with the four corners (-1,-1), (-1,1), (1,-1), and (1,1) was used. Note that the origin is in the center of this square. The X-axis divides it into two halves horizontally, while the Y-axis bisects it vertically.

1.3. Scale. Having staked out such a square, the next step is to examine it in small increments. The horizontal component has been divided into 1,800 equal parts, and the vertical component into 1,440 equal parts. This results in a non-square rectangle of 1,800 X 1,440 or 2,592,000 pixels.

1.4. Complicated formula. Nested loops cause the computer to visit all 2,592,000 pixel locations in the final picture, one by one. The fairly complicated formula is not laid out succinctly. Notice in the formula below that the variable, "s1," is set equal to the variable, "s3," plus the same variable, "s3." This awkward formulation is typical. It has been used to make it easy for the programmer to alter this formula when the time comes to create the next, variant, image.

1.5. Code in e-run. Here is the code from the file that creates Trig0095. The variables, xx and yy, directly represent the x-coordinate and y-coordinate for each pixel. The variable, col, is used to determine the color assigned to the pixel at that location. The software used is Bob Brill's e-run.

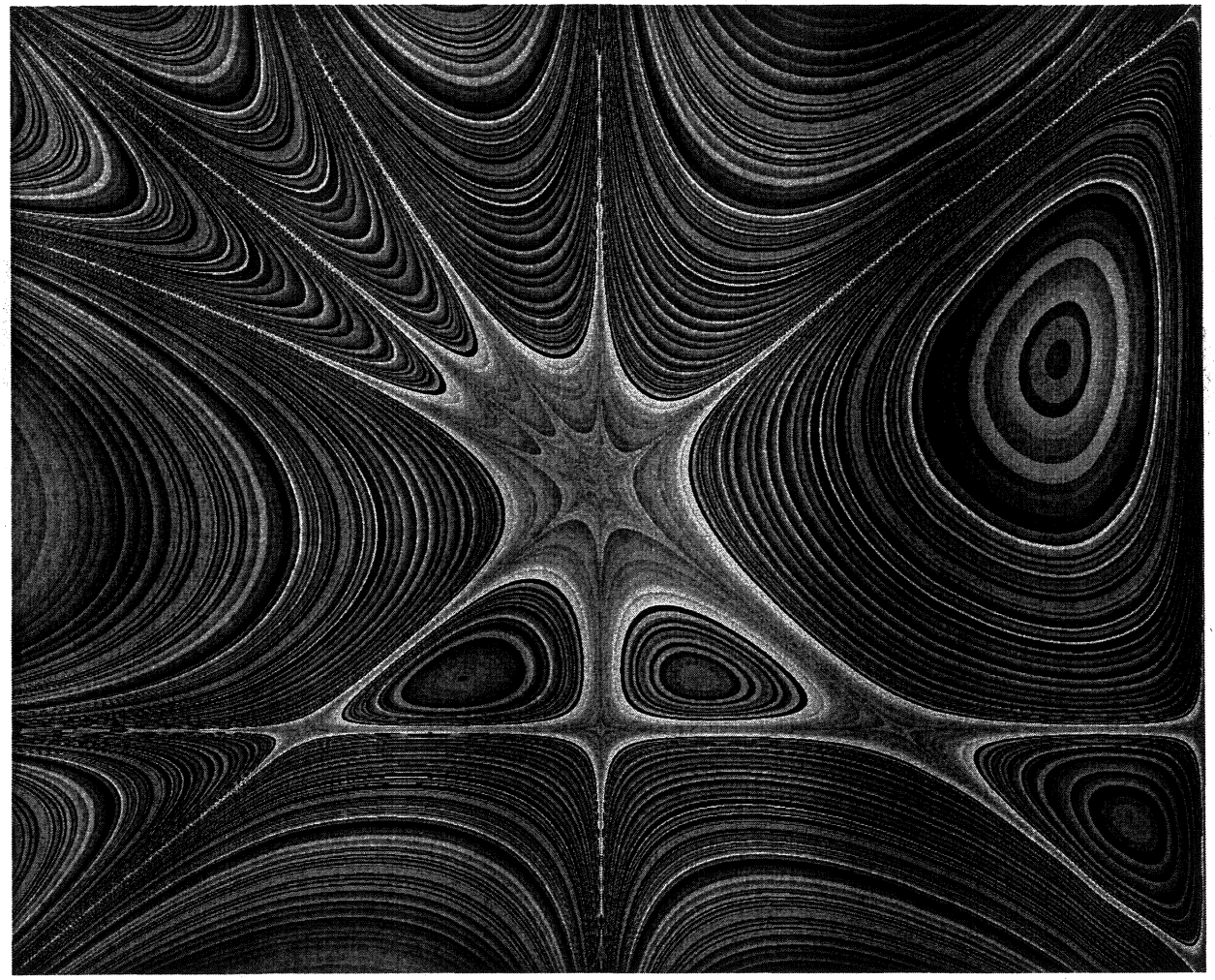
```
t3 <- arctan[-xx-xx-xx]*17/24
t1 <- t3+t3
t2 <- arctan[(yy+yy+1)*(xx-1)]*17/24
t <- sqrt[abs[t1*t2]]
c1 <- arcsin[(xx-yy)/2]*17/24
s3 <- abs[arctan[-yy-xx]*17/24]
s1 <- s3+s3
s <- sqrt[abs[(s1+c1-1)*(s1*c1+1)]]
col <- sqrt[abs[t*s]]
```

1.6. **Imperfect Symmetry.** In the nine lines of formula shown, symmetries and asymmetries are juggled and misaligned only slightly. The final image was arrived at not by a process of careful analysis. Rather, it was built up over time through a series of intermediate images and steps.

1.7. **From 360 to 256.** Bob Brill's e-run software has degrees, rather than radians, built into its trigonometric functions. This means that every arctangent and arcsine calculation produces a number between 0 and 359. For each pixel, an index number between 0 and 255 will be needed, so a re-scaling calculation is necessary. Multiplying by 17 and dividing by 24 does trick.

1.8. **Formula as art.** The image below represents a manifestation of the long and unwieldy formula shown on the previous page. This formula is not efficient, nor is it novel or insightful. Instead, it produces an image.

2. Image – Trig0095



References

Brill, Bob, "The Endless Wave" in Bridges Conference Proceedings, 2002.