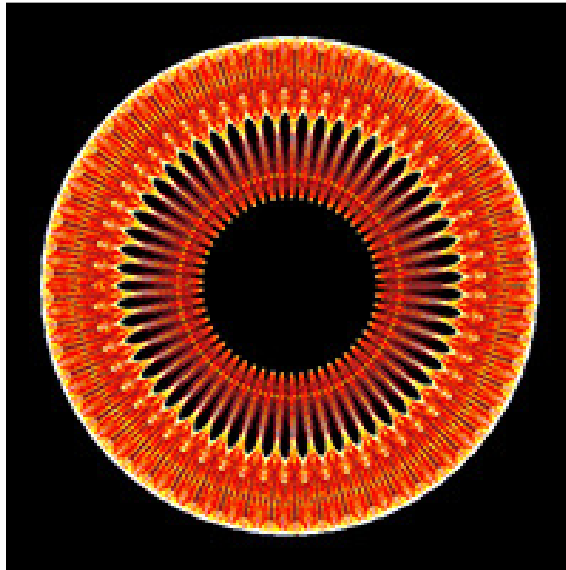


QS SymXaos



Cover Image: Icon, Formula 2 - “Sunflower”

About the Program

This program uses the five attractor formulae described in the book *Symmetry in Chaos*, by Michael Field and Martin Golubitsky (Oxford University Press, 1992). These formulae generate striking symmetric icons, and tiling patterns which the authors refer to as “quilts”.

The program was written in 1997 for Windows 95, with tweaks in 2000 and 2018. It is a 32-bit program which should continue to work in current 64-bit Windows versions.

Running the Program

Images can be rendered in resolutions of up to 4000 x 4000 pixels. Rendering, pausing and resuming are controlled by the **R** key, or by left or right mouse button clicks. Rendering must be paused to access the menu options. There are two default palettes. You also can select any valid Fractint MAP file. The background can be specified as a triplet of RGB values. The scale factor controls the image window; larger values simulate zooming out and yield smaller images or more quilt components.

Images can be saved as TARGA files. Parameter sets can be saved as “SX” files and can be loaded into the program later. After associating these files with the program, you then can load them by double-clicking on their icons to start the program. A set of parameter files for all of the plates in the book is included, along with a collection of additional sample parameter sets.

Each set of parameters which is used to render an image is saved to a cumulative file called "SXHistory.txt". This file can be viewed within the program from the "Parameters" dialog box. However, the file is overwritten each time the program starts.

Parameter sets can be entered in the dialog box, or random sets of parameters can be generated for the currently-selected image type. However, chaotic attractors are by nature unpredictable. Many parameter sets, especially for the two Icon formulae, will overflow and potentially crash the computer. If the program detects that a parameter set will explode, you will be requested to enter new parameters, or a new random set will be evaluated. Additionally, some sets will converge to a small number of finite points, resulting in images of just a few flickering pixels or some lines. These will not overflow, but will not yield very impressive pictures either. Some of the techniques that can be used to screen for stable and interesting images are described in Appendix 1 of this file.

There are two coloring methods. The "hit count" algorithm advances the color of each pixel through a 256-color palette, each time it is hit during iteration of the formula. Therefore it takes time for the colors of the image to emerge, much like a photographic print in developing fluid. So be patient and give the image a chance. However, as iteration continues, progressively more points go beyond 255 hits and are colored with the last palette entry, "overdeveloping" the image. Ultimately, most of the image can assume that one color. To compensate for these trends, palette entries can be redistributed through the normalization and equalization options. Normalization compresses or expands the distribution into the range of 1 to 255. However, this significantly under-represents the upper range of the palette, so the normalization adjustment menu option allows the amount of normalization to be specified. An alternative to linear normalization uses the logarithms of the hit counts, which tends to emphasize the middle range of the palette. Equalization is an attempt to spread the color values evenly throughout the image, making better use of the entire palette. Further discussion of these techniques can be found in Appendix 2 of this file.

The "speed" algorithm is used frequently for chaotic attractors. It uses the distance between the currently-calculated position and the previous one as an indicator of how "quickly" the point had to move to get there, assuming a constant iteration rate, and assigns palette colors accordingly.

After normalization or equalization, a histogram showing the relative number of pixels which are colored by each palette entry can be displayed.

The help file for this program, "SymXaos.hlp", uses a format that Microsoft abandoned after Windows XP. It requires the installation of a file named "WinHlp32.exe", which seems to have different versions for each subsequent flavor of Windows. At least for now, a WinHlp32 installer can be downloaded from the Microsoft Web site:

<https://support.microsoft.com/en-us/kb/917607>

You'll have to select from various versions and hope that one of them works for your system. Since it's quite possible that none of them will, I've included this PDF version of the help file,.

Image Types

Icons, Formula I

$$z = [\text{lambda} + (\text{alpha} * z * Z) + (\text{beta} * \text{Real}(z^n)) + \text{omega} * i] * z + \text{gamma} * Z^{(n-1)}$$

The program iterates the complex number $z = x + yi$ and plots the resulting points. The symbol Z means the conjugate of z ($z\text{-bar}$) = $x - yi$. If ($\text{omega} \neq 0.0$) an element of asymmetry is introduced. The symbol n is an integer which determines the axes of symmetry.

Scale range: 0.1 - 5.0

Sample parameter sets:

alpha	beta	gamma	lambda	omega	n (2 - 25)
-2.5	0.0	0.9	2.5	0.0	3
2.32	0.0	0.75	-2.32	0.0	5
-2.5	0.0	0.9	2.409	0.0	23
-2.5	-0.2	0.81	2.409	0.0	24
5.0	-1.9	1.0	-2.5	0.188	5
-2.0	0.0	-0.5	2.6	0.0	5
2.0	0.0	1.0	-1.8	0.0	4
2.0	0.0	1.0	-1.86	0.1	4
-1.0	0.1	-0.8 / -0.805	1.5	0.0	3
-2.5	-0.1	0.9	2.39	-0.15	16
-2.5	-0.1	0.9	2.39	0.0	16
-1.0	0.1	-0.82	1.56	0.0	3
-1.0	0.1	-0.82	1.56	0.12	3
1.806	0.0	1.0	-1.806	0.0	5
1.0	-0.1	0.167	-2.08	0.0	7
10.0	-12.0	1.0	-2.195	0.0	3
2.0	0.2	0.1	-2.34	0.0	5
3.0	-16.79	1.0	-2.05	0.0	9
-1.0	0.03	-0.8	1.455	0.0	3
5.0	1.5	1.0	-2.7	0.0	6
-2.643	2.107	-1.563	2.329	0.615	19

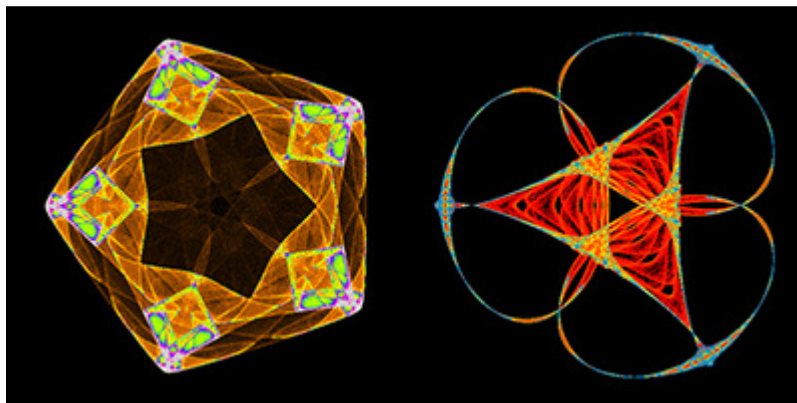
Icons, Formula II

$$z = [\lambda + (\alpha * z * Z) + (\beta * \text{Real}(z^n)) + (\omega * \text{Real}((z / |z|)^{(n * p)} * |z|))] * z + \gamma * Z^{(n-1)}$$

The program iterates the complex number $z = x + yi$ and plots the resulting points. The symbol Z means the conjugate of z ($z\text{-bar}$) $= x - yi$. The symbol n is an integer which determines the axes of symmetry. The symbol p is an integer which introduces a “mild singularity at the origin, but with symmetry preserved.” (Field & Golubitsky, p. 193)

Scale range: 0.1 - 5.0

alpha	beta	gamma	lambda	omega	n (2 - 60)	p (0 - 30)
8.0	-0.7	1.0	-2.5	-0.9	9	0
-2.1	0.0	1.0	1.0	1.0	3	1
1.0	-0.04	0.14	-2.42	0.088	6	0
-1.0	0.03	-0.8	1.455	-0.025	3	0
-1.0	-0.02	-0.75	1.5	0.04	3	24
10.0	-12.3	0.75	-2.38	0.02	5	1
1.5	-0.014	0.002	-2.225	-0.02	57	0



Icon, Formula 1

Icon, Formula 2

Square Quilts

$$f(x, y) = n * (x, y) + (v, v) + \\ \lambda * (\sin(2 \pi x), \sin(2 \pi y)) + \\ \alpha * (\sin(2 \pi x)\cos(2 \pi y), \sin(2 \pi y)\cos(2 \pi x)) + \\ \beta * (\sin(4 \pi x), \sin(4 \pi y)) + \\ \gamma * (\sin(6 \pi x)\cos(4 \pi y), \sin(6 \pi y)\cos(4 \pi x)) - \\ \omega * (\sin(2 \pi y), \sin(2 \pi x))$$

The symbol n is an integer, and v is a floating point “shift factor”: (0.0 or 0.5).

Scale range: 0.5 - 5.0

alpha	beta	gamma	lambda	omega	v	n (-2 - 2)
0.2	0.1	-0.33	-0.59	0.0	0.0	2
0.2	0.1	-0.27	-0.59	0.0	0.5	0
-0.1	0.1	-0.25	-0.2	0.0	0.0	0
-0.3	0.2	0.3	0.25	0.0	0.0	1
0.25	0.05	-0.24	-0.28	0.0	0.0	-1
0.25	0.05	-0.24	-0.28	0.075	0.0	-1
-0.36	0.18	-0.14	-0.12	0.0	0.5	1
0.2	0.1	0.39	0.1	0.0	0.0	-1
0.2	0.04	-0.2	-0.589	0.0	0.5	0
0.08	0.45	-0.05	-0.28	0.0	0.5	0
0.2	0.2	0.3	-0.59	0.0	0.0	2
0.25	0.05	-0.24	-0.28	0.0	0.5	-1
-0.26	0.19	-0.059	-0.11	0.07	0.5	2

Hexagonal Quilts

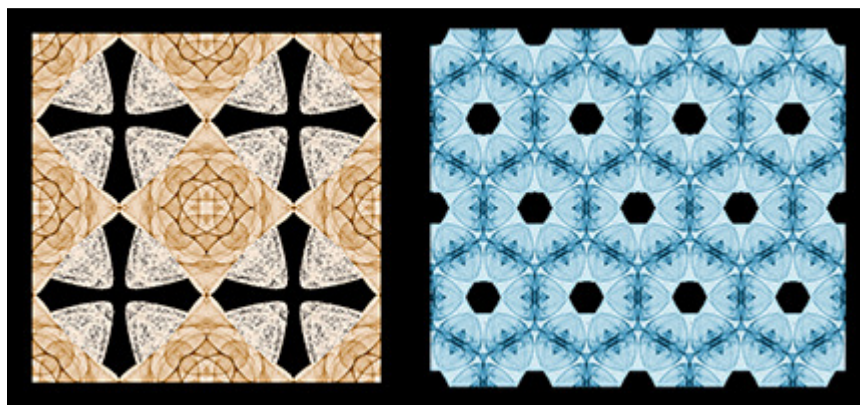
This description is quoted directly from Field & Golubitsky (p. 195).

$$f(X) = nX + C [\sin(2\pi N.X)N + \sin(2\pi RN.X)RN + \sin(2\pi R^2 N.X)R^2 N] + \\ \alpha [\sin(2\pi M.X)M + \sin(2\pi RM.X)RM + \sin(2\pi R^2 M.X)R^2 M] + \\ \sin(2\pi L.X)A + \sin(2\pi RL.X)RA + \\ \sin(2\pi R^2 L.X)R^2 A + \sin(2\pi FL.X)FA + \\ \sin(2\pi RFL.X)RFA + \sin(2\pi R^2 FL.X)R^2 FA$$

“where $X = (x, y)$ is a point in the plane, $L = (3, 1/\sqrt{3})$, $M = (2, 0)$ and $N = (1, -1/\sqrt{3})$. R , F and C denote linear transformation of the plane; R is rotation counter-clockwise by 120° , F is reflection in the x -axis and C corresponds to complex multiplication by $(\lambda + \omega i)$. We have used the dot product notation $N.X = n_1x + n_2y$, where $N = (n_1, n_2)$ and $X = (x, y)$. This formula has five real parameters α , λ , $A = (\beta, \gamma)$ and ω ; it has D_6 symmetry when $\omega = 0$ and Z_6 symmetry when $\omega \neq 0$.”

Scale range: 0.25 - 5.0

alpha	beta	gamma	lambda	omega	n (0 - 2)
-0.076	-0.06	0.1	0.1	0.0	0
-0.076	-0.06	0.1	0.1	0.101	0
0.04	0.1	0.1	0.2	0.0	1
-0.15	0.06	-0.03	-0.105	0.0	0
-0.1	0.14	0.052	0.02	0.0	0
-0.1	0.14	0.052	0.02	0.04	0



Square Quilt

Hexagonal Quilt

Symmetric Fractals

$$f(x, y) = (ax + by + e, cx + dy + f)$$

n is an integer which determines the axes of symmetry and $r = \text{random}(n)$:

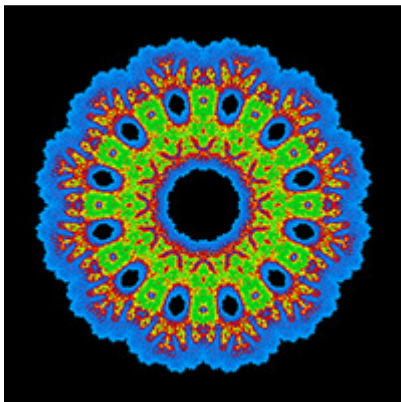
$$\begin{aligned}x &= \cos(2 \pi r / n) * x - \sin(2 \pi r / n) * y \\y &= \sin(2 \pi r / n) * x + \cos(2 \pi r / n) * y\end{aligned}$$

Finally, the sign of y is randomly inverted. The system must be a contraction, i.e.

$$a^2 + c^2 < 1 \quad b^2 + d^2 < 1 \quad a^2 + b^2 + c^2 + d^2 < 1 + (ad - bc)^2$$

Scale range: 0.01 - 5.0

a	b	c	d	e	f	n(2 - 100)
0.5	0.0	0.0	0.5	0.5	0.0	3 / 5 / 6
0.45	0.0	0.0	0.45	0.55	0.0	3
0.46	0.0	0.0	0.46	0.54	0.0	4
0.45	-0.1	-0.31	0.45	0.1	0.2	11
0.45	-0.1	0.3	-0.4	0.15	0.1	4 / 8
-0.1	0.35	0.2	0.5	0.5	0.4	3
-0.25	-0.3	0.14	-0.26	0.5	0.5	12
-0.25	-0.3	0.3	-0.26	0.5	0.5	8
-0.25	-0.3	0.3	-0.34	0.5	0.5	4
-0.15	0.75	0.2	-0.3	0.075	0.4	6 / 50
-0.4	0.75	0.2	-0.3	0.0	0.4	55
0.4	-0.1	-0.35	0.4	0.01	0.2	9
0.4	-0.1	-0.31	0.45	0.01	0.0	5



Symmetric Fractal

Programming and help file authoring by:
Michael Sargent
South Burlington, Vermont
February 2018

msargent@uvm.edu
<http://www.uvm.edu/~msargent>

Disclaimer

This is unsupported, non-standard software. It is provided “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Appendix 1: Evaluating Parameter Sets

Testing for overflow can be approached by checking whether the result of each iteration falls within arbitrarily-chosen boundaries. However, if the boundaries are too restrictive, many well-behaved parameter sets will be discarded. Also, there are some sets which will not cause overflow until many iterations after the limit of the evaluation loop, so the rendering function needs to continue to check each result. In this program, potential overflow while rendering is addressed by resetting the input variables. If this happens, a noticeable shift in the coloring usually will become apparent. Formulae like the quilts and symmetric fractals, which use sines and cosines, are not prone to overflow.

Attractors can contract to a fixed point or a periodic cycle, such as those found in the Mandelbrot set. In contrast, *strange attractors* are systems whose final state is complex and chaotic. These alternatives are characterized by the *Lyapunov exponent*. Calculation of this value is described by Julien Clinton Sprott in various publications, including

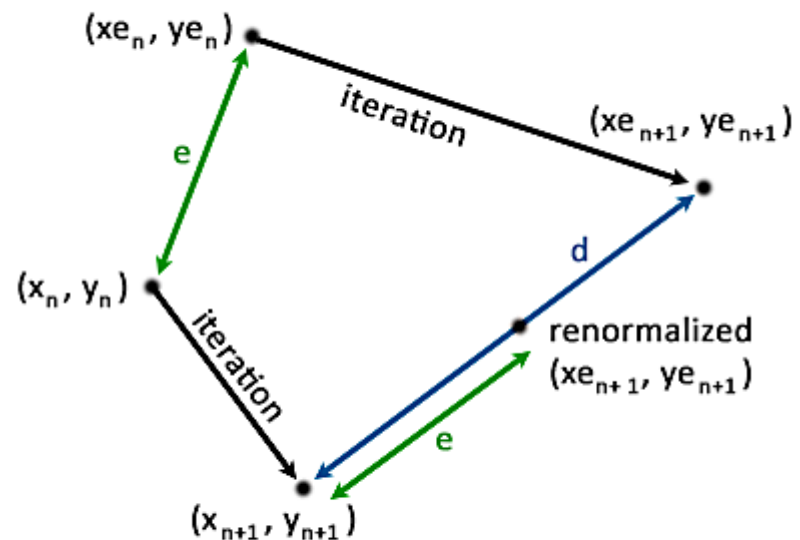
<http://sprott.physics.wisc.edu/chaos/lyapexp.htm>

and by Heinz-Otto Peitgen *et al.* in their text *Chaos and Fractals* (Springer-Verlag, 1992), Chapter 12.5. The process can be summarized as follows: Pairs of close points which are further apart after iteration tend to show chaotic behavior, but if they are closer, they tend to converge to an attracting system. Assume that an attractor formula yields a sequence of points (x, y) . After a hundred or more iterations to establish that subsequent points are on the attractor, the next point (x_n, y_n) is perturbed by a small error “ e ” at an arbitrary angle. The value of the angle does not matter. In one of the simplest instances, if the angle is 0.0, the perturbed point (x_{e_n}, y_{e_n}) will be $(x_n + e, y_n)$. After one iteration, the resulting points (x_{n+1}, y_{n+1}) and $(x_{e_{n+1}}, y_{e_{n+1}})$ will be a distance “ d ” apart. The logarithm of the ratio d/e will be negative, zero or positive, depending on whether $d < e$, $d = e$ or $d > e$. The average of this logarithm over many iterations yields the Lyapunov exponent.¹ After each iteration, the point (x_{e_n}, y_{e_n}) should be “renormalized” so that it is the original distance e from (x_n, y_n) , and in the same direction. In *Chaos and Fractals*, a

¹To be rigorous, the value should be calculated as e approaches 0.0, which requires using the derivative of the attractor's transformation. This involves much more math than is necessary for a recreational computer program.

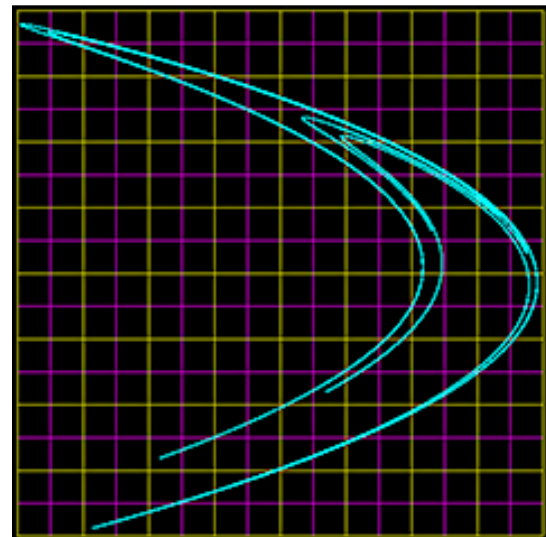
diagram (Figure 12.43) similar to this one is used to represent a step in this process.

Ultimately, if its Lyapunov exponent is positive, a system will exhibit chaotic behavior, as the distances between the iterations of nearby points will, on the average, diverge. Thus, images of such systems should produce aesthetic, or at least interesting, patterns. In this program, parameter sets are accepted when the Lyapunov exponent arbitrarily is greater than 0.075. The smallest value from the images in *Symmetry in Chaos* is 0.19.



Another means of assessment of an attractor is its *dimension*. Mandelbrot described three types of *fractal dimension*, one of which, the *box-counting dimension*, is actually applicable to systems which are not self-similar. The method consists of placing a sequence of grids over an image of the system. Typically these grids are spaced at intervals of 2^{-k} , where $k = 0, 1, 2, \dots$

In this image of the Henon attractor, the k values are 3 (yellow) and 4 (magenta), so that there are 64 yellow and 256 magenta boxes. For each iteration, the boxes in which the resulting point lands are recorded. When iteration is finished, all boxes which contain points are counted. Peitgen's terminology in Chapter 4.4 of *Chaos and Fractals* for the total number of boxes of each grid is $N(2^{-k})$. The logarithms of these values can be plotted against the logarithms of the grid sequence (2^k). The slope of the line connecting the points will yield the dimension. In this instance, the box counts N are 30 and 71 for 10,000 iterations, so that



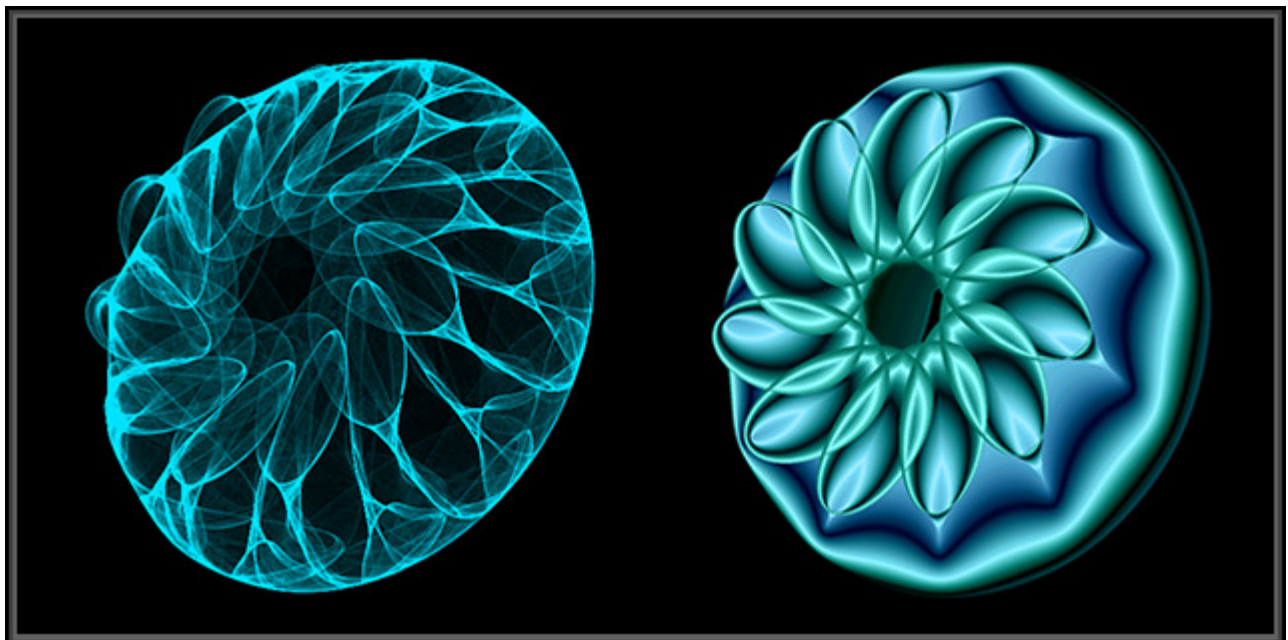
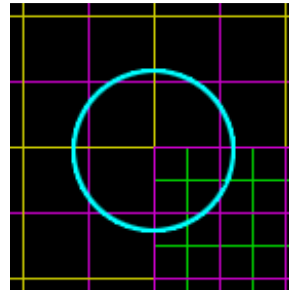
$$\frac{\log_{10}(71) - \log_{10}(30)}{\log_{10}(16) - \log_{10}(8)} = 1.2429\dots$$

Any base logarithm can be used. The formula can be simplified, using base 2 logarithms, as follows:

$$\log_2 \frac{N(2^{-(k+1)})}{N(2^{-k})} = \frac{\log_{10} \left(\frac{N(2^{-(k+1)})}{N(2^{-k})} \right)}{\log_{10}(2)}$$

Dimensions determined by this method cannot exceed 2. For rigorous purposes, the sequence of k would have many more members, of much higher values. For the purpose of this program, a simple pair of 3 and 4 seems sufficient. Parameter sets are accepted when the dimension arbitrarily is greater than 1.5, which will eliminate most, though not all, sets that yield small clusters or lines. Thus, the Henon attractor would not be considered “attractive” enough.

However, the box-counting dimension sometimes can be misleading. This circular attractor crosses 4 large and 12 small boxes, yielding a dimension of 1.5849.... But using a smaller grid (lower right) yields a ratio of 12 large to 20 small boxes, and a dimension of 0.7369....



Icon, Formula 1, rendered with QS SymXaos 3D
Lyapunov exponent: 0.41
fractal dimension: 1.85

Appendix 2: Image Enhancement

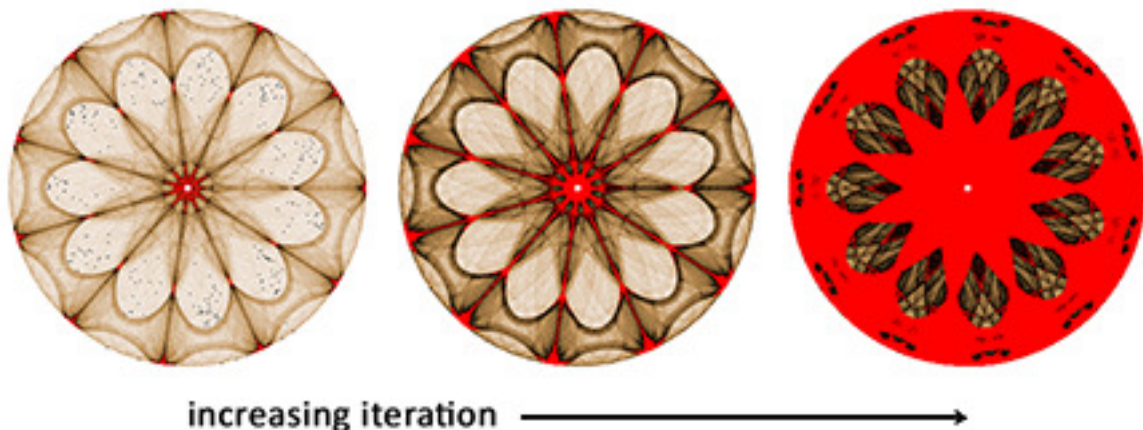
For images of chaotic attractors, an effective coloring method assigns an entry from a color palette according to the number of times the pixel has been “hit” by an iterating point.²

This is a representative palette of 256 colors, with indices of 0 to 255, which is derived from a text file called a MAP file.



These files were developed for the pioneering fractal rendering software *FractInt* as an easy way of creating, using and exchanging palettes. The first color of the palette is pure black. This is commonly done so that palette[0] can be used for the background by a program. Therefore, *QS SymXaos*, like many other programs, starts coloring images with palette[1] and ignores all background pixels, i.e. those that are not hit at all and therefore have a hit count of zero.

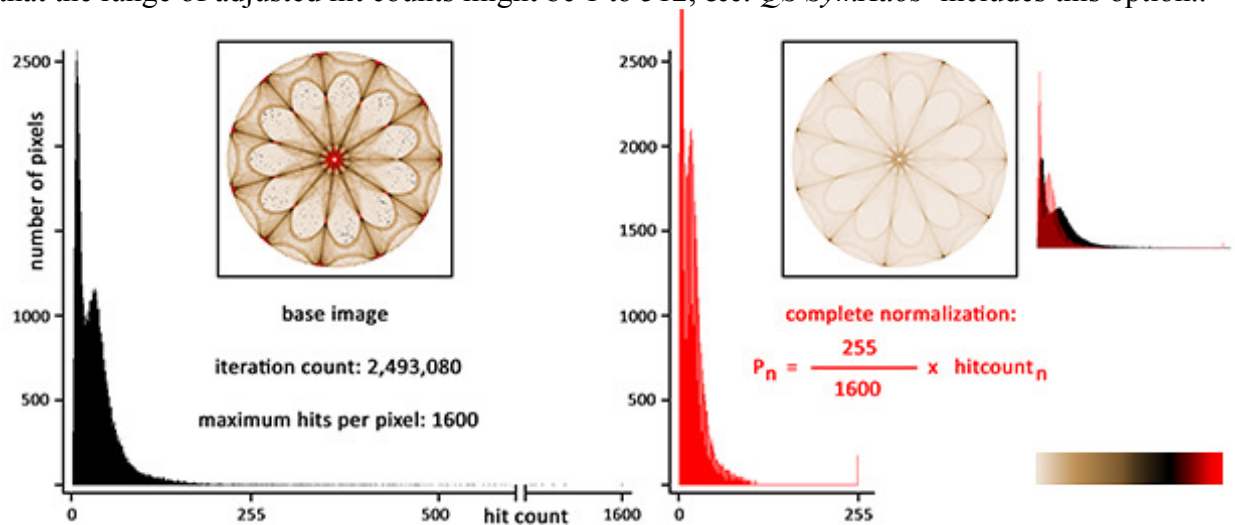
Using this palette with the formula for the program’s opening icon, we can generate images like these. The image on the right demonstrates a liability of this method. As iteration continues, more and more pixels will be hit in excess of 255 times, and so the number of pixels colored with palette[255] will increase until potentially all of the image could be engulfed. Alternatively, the palette index for the 256th hit could be “wrapped” back to 1, but arguably this results in less appealing images.



One method of compensating for this situation is *normalization* of the hit counts, in which the count for each pixel is multiplied by a factor consisting of 255 divided by the maximum hit count of all the pixels. Thus, all hit counts will be normalized to a range of 1 to 255. But this method also has potential liabilities. The following example shows an image which has been iterated 2,493,080 times, resulting in a maximum hit count of 1600. This means that at least one pixel has been hit 1600 times, and it turns out that there is only one. The next highest hit count is 1522, again only for one pixel. The highest hit count shared by 2 pixels is 1087.

²It should be understood that as iteration continues, an infinite number of different points can land within the boundaries of a single pixel. However, the likelihood of this happening will diminish gradually as the pixel resolution of the image increases.

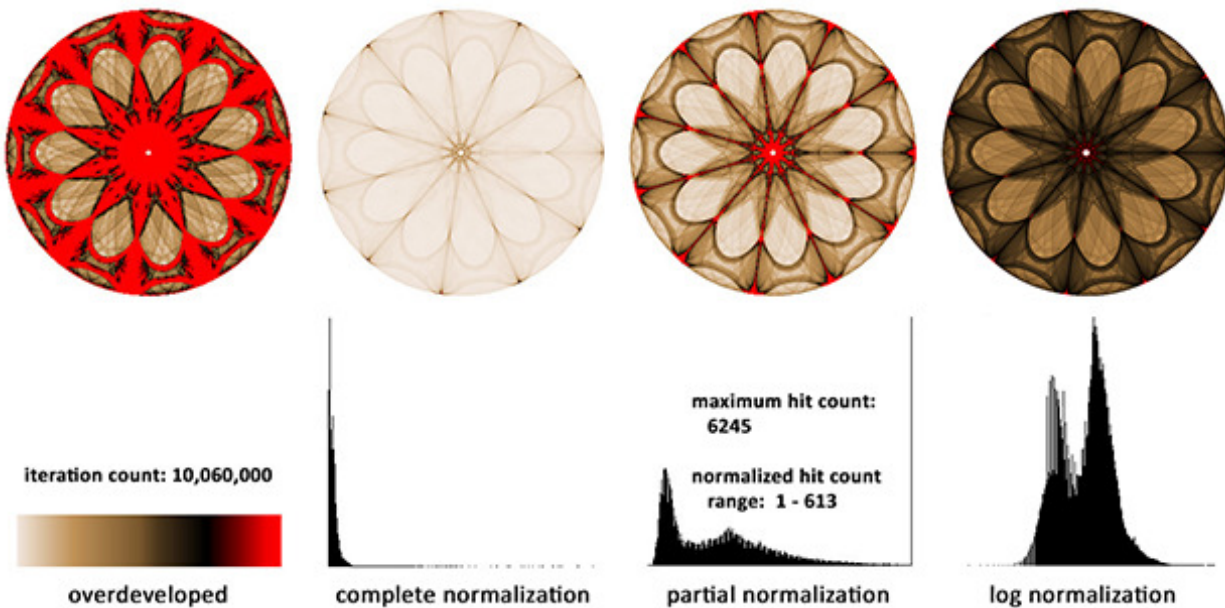
On the left side, a histogram shows the number of pixels for each hit count. Clearly, the majority of pixels have a relatively low count, and therefore colors from the lower portion of the palette predominate. On the right side, the hit counts have been normalized. The histogram, now also corresponding directly to palette indices, is the same shape, but it has been “squeezed” even further to the left, and since the number of pixels has not changed, the height of the histogram is about twice that of the original. The resulting image has even less representation of colors from the higher portion of the palette. Although this sometimes results in appealing images, it would be desirable to see a more even balance of colors. One way to do this is to normalize partially, so that the range of adjusted hit counts might be 1 to 512, &c. *QS SymXaos* includes this option..



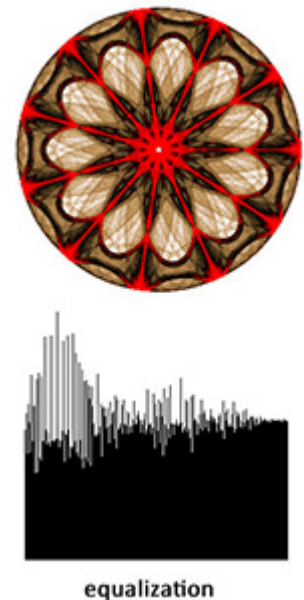
It should be noted that these histograms are not like those that are used by image processing software for contrast enhancement. The histogram to the right, from PhotoShop, plots the number of pixels corresponding to the tonal range from pure black to pure white, and is completely dependent on the actual image. But the hit count and palette index histograms described above are only dependent on the results from iterating the formula. They are equally applicable to any palette and actually are calculated before any pixels are plotted.



This example shows how an overdeveloped image can be adjusted. We already have seen how complete normalization affects the range of the palette, which again is confirmed by the histogram. With partial normalization, the palette is more spread, but the upper end is still under-represented, except that all the pixels with hit counts between 614 and 6245 have been assigned a palette index of 255. In logarithmic normalization, 255 is multiplied by the logarithm of the hit count divided by the logarithm of the maximum hit count. This results in emphasis on the middle range of the palette. If the two logarithms are squared, a similar histogram shape is seen, but it is shifted more toward the lower end of the palette.



Finally, we see an *equalized* image with an even distribution of colors across the palette. The method is described in the next two pages. (Note that these histograms represent the same number of total pixels, so to fit them into the same-sized spaces, their Y-axis scales have been adjusted).



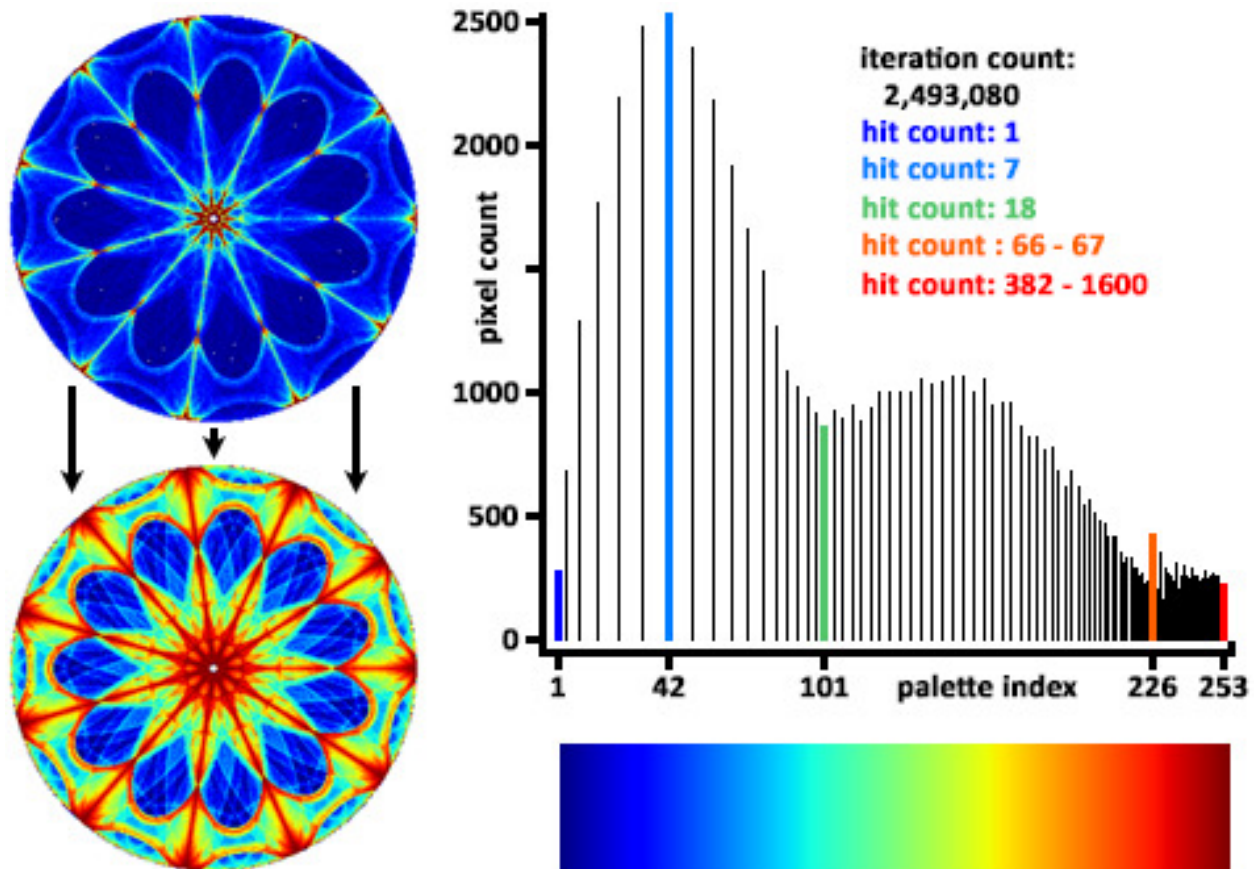
The process of histogram equalization is similar whether a luminance histogram is used to enhance the contrast of a radiographic image, or, in this case, a hit count histogram is used to distribute the colors of a palette evenly across an image. An array of the number of pixels associated with each hit count is converted to a “cumulative distribution function” (cdf) or “cumulative histogram”, in which each hit count is associated with its own number of pixels added to the sum of all the preceding numbers. Thus, in the following formula, $\text{cdf}_{\min}$ is the lowest non-zero value in the histogram, and $\text{cdf}_{\max}$ is the last value, corresponding to the total number of pixels of the image. P is the number of palette entries (256). An adjustment is made because background pixels with a hit count of 0 are ignored, so that the desired range of palette indices is 1 to 255.

$$\text{index}_n = \frac{\text{cdf}_n - \text{cdf}_{\min}}{\text{cdf}_{\max} - 1} \times (P - 2) + 1$$

For the example on the next page, the table shows representative data from various hit counts for an image in which rendering again was stopped at 2,493,080 iterations.

$$\text{For hit count 4,} \quad \text{index}_4 = \frac{4047 - 295}{67415 - 1} \times (256 - 2) + 1 = 15$$

hit count	number of pixels	cdf	palette index
1	295	295	1
2	690	985	3
3	1294	2279	8
4	1768	4047	15
5	2187	6234	23
6	2475	8709	32
7	2530	11239	42
8	2386	13625	51
-	-	-	-
17	921	26167	98
18	872	27039	101
19	931	27970	105
-	-	-	-
65	261	59814	225
66	224	60038	226
67	214	60252	226
68	223	60475	227
-	-	-	-
1599	0	67414	253
1600	1	67415	253



The example shows that a “mere” two and a half million iterations is not sufficient for the equalization algorithm to be completely successful. Numerous palette indices are skipped, and the distribution is far from even. Notice that as the hit counts increase, the algorithm assigns the same palette index to pixels with different hit counts if the numbers of pixels are small. Index 226 is assigned to pixels with hit counts of 66 and 67, and index 253 is assigned to all pixels with hit counts from 382 to 1600. The fact that a different palette is used to render the images has no bearing on the calculations.

Compare these results to the previous example (lower image) in which the iteration count is over 10 million and the maximum hit count is over 6 thousand. In that instance, all the palette indices are used for the equalization, and the distribution is much more even.

However, from an esthetic viewpoint, equalization is not necessarily “better” than normalization, logarithmic normalization, equalization of a normalized image, or any enhancement at all. Ultimately, any of these approaches has the potential to produce subjectively pleasing images, depending on the number of iterations and the selected palette.

