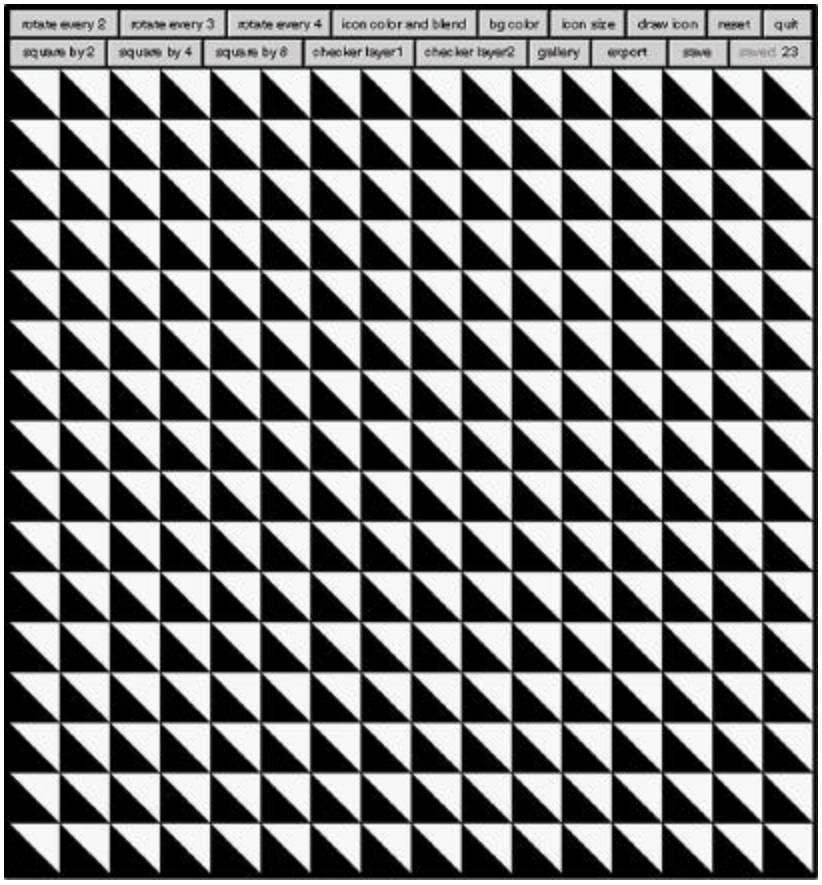
T h e r o t a t o r E n g i n e



Naomi Pitcairn

Contents:

Introduction:

The Rotator Engine is an interactive application that allows users to explore and learn about some specific aspects of two-dimensional pattern. This is achieved by using a controlled set of tools that my application provides. The user interface consists of two layers of a square matrix of interchangeable motifs. A series of operations can be performed on the matrix, that change various aspects of its motifs and their relationships to each other. These operations affect visual changes that can be recorded and replayed. The operations, in brief, are rotation of the various motifs in various groups, changeable motifs and variable size, color and transparency of the motifs in each layer.

Projects:

Illustrations of this type are useful because when parts of patterns can be moved and adjusted in relationship to each other, the user can watch systems reacting and adjusting. This helps to clarify and simplify their underlying relationships. In my research on patterns, I came across much good material, but it was mostly in the form of static documents.

Though patterns may be static in themselves, a tool capable of dynamically showing the steps that produce them can help reveal their structure and organization. Artists who are interested in the possibilities of mathematics but fearful of them might find this type of an application to be a useful visualization tool to aid them in understanding symmetry on a plane and other visually interesting phenomena.

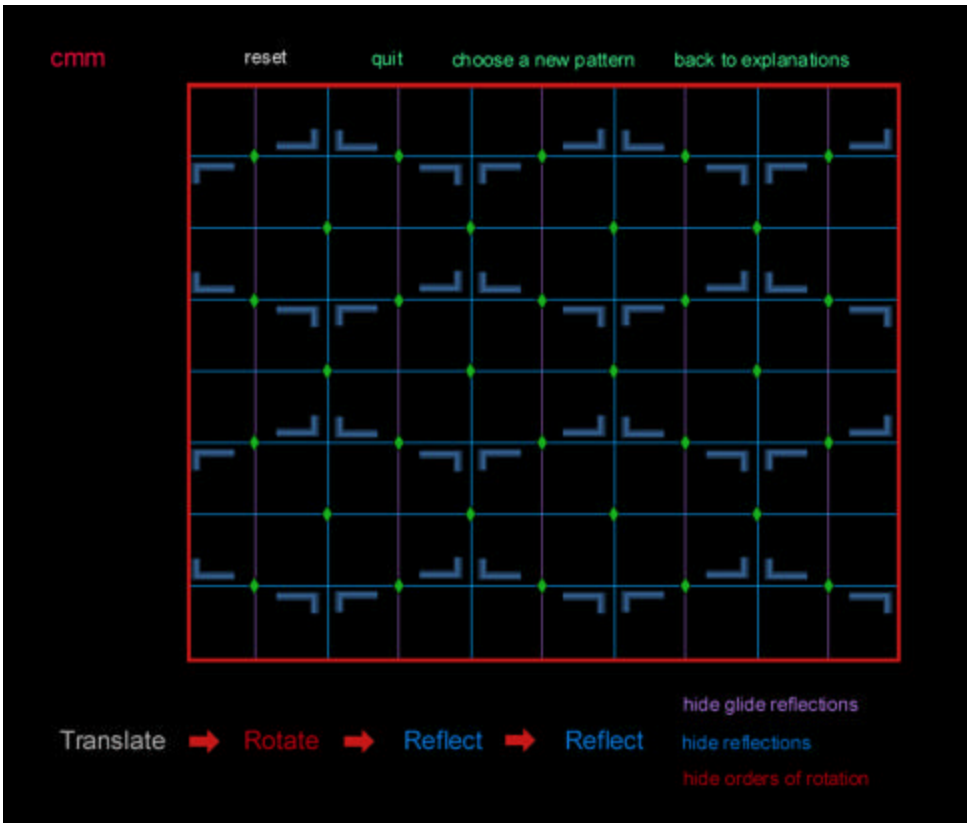
This particular project started out as a personal visualization tool, for illustrating certain mathematical and programmatic concepts. It evolved into a visualization tool that I felt someone else, besides myself could use. Ultimately, it has come to resemble a design tool and that is what I would ultimately like to it to end up being. I would like its final iteration to be a design tool that was easy to use but that also made the underlying, mathematical principles it was based on, transparent to the user. With this application I hope to facilitate the teasing out of the more interesting structures and organizations for the user/designer and to allow for the rapid formation of arrangements that are visually pleasing. This project is a continuation of ideas that I explored earlier in my education.

*****

The first tool I made was a dynamic exposition of symmetry on a plane. It demonstrated the different transformations (translations, rotations, reflections and glide

reflections) as they were applied to form each of the 17 Wallpaper Patterns (which are explained in detail below). The user was able to choose from the list of 17 and watch a simple motif go through the requisite transformations to form the pattern. It was more of a demonstration than a design tool.

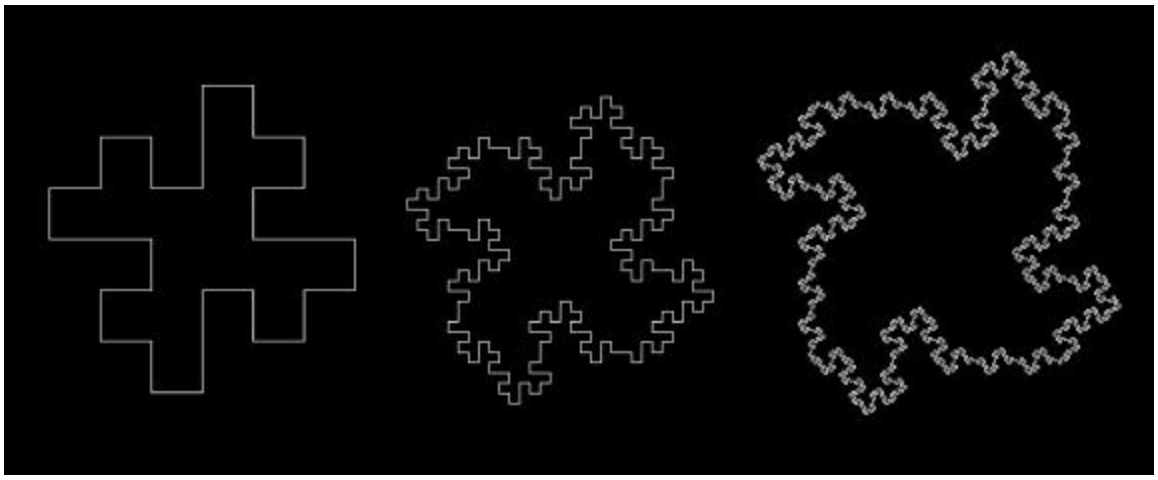
A screen shot from the 17 Wallpaper Patterns:



The second related project I did, called "L-1," (for L-systems # 1) was a dynamic illustration of the Quadratic Koch Curve, one of many fractal generators. It enabled a user to draw a line using 90 degree angles and then re-draw the perimeter of a square using that line. They could then repeat a series of iterations, each one redrawing each segment of the last line with the original line until the pattern became too complex for the screen to handle. The resulting image was complex and fractal.

A Screen Shot from L-1:

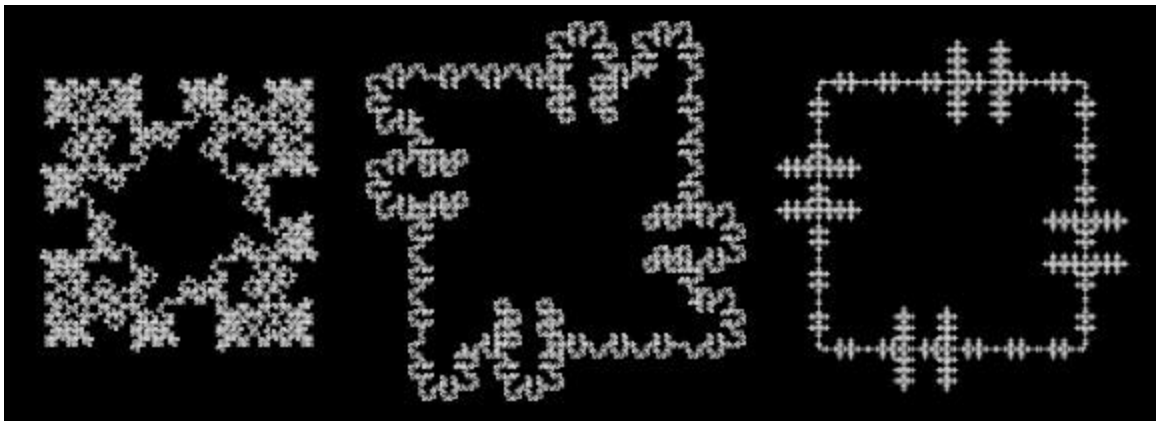
1. One series of iterations of increasing complexity



first iteration **second iteration** **third iteration**

The L-1 project lies in between a teaching tool and a design tool. It has a fairly large range of images it can create since the final outcome depends on user input. Nonetheless, it is of limited use for practical design purposes.

A selection of final images from L-1



Available Pattern Design Software:

There are a of variety pattern design software packages already available, most notably PrimaVision and U4ria, both distributed by a company called Lectra Systems. Both applications create repeats for textile design, but are very expensive (~$10K) often require special cards and do not offer a comprehensive package of the possible transformations that are mathematically possible. A more comprehensive package would greatly enrich the collection of possible designs a user might create.

I learned Prima Vision quite thoroughly over the summer of 1998. It did not conform well to traditional graphics conventions. For that reason and others, I think it has bad interface design. An application geared as a PhotoShop plug-in, would be far more practical for most users. I would be interested in taking such a project on some time in the future.

An interesting small application I found, was an applet called Java Kali that lets a user interact with the 17 wallpaper patterns.

Printouts of some images created in Java Kali

Origins:

Most of my recent work comes out of an interest in natural phenomena, organic forms, organization, tilings and crystallization patterns. More specifically this work comes out of 2 general lines of interest and one specific subject of study. The 2 major lines of interest are: 1. An interest in pattern and 2. An interest in computer programming. The specific subject of study is the work of P. Dominique Douat.

1. An interest in pattern:

Visual patterns, besides being beautiful, are both complicated and simple at the same time. Our brains are so constructed that we can make out the slightest irregularity in a pattern design instantly but when we sit down and try to analyze what we actually see, it quickly gets confusing. A small, discreet set of mathematical rules combined and applied to individual images, can create an endless variety of visual patterns.

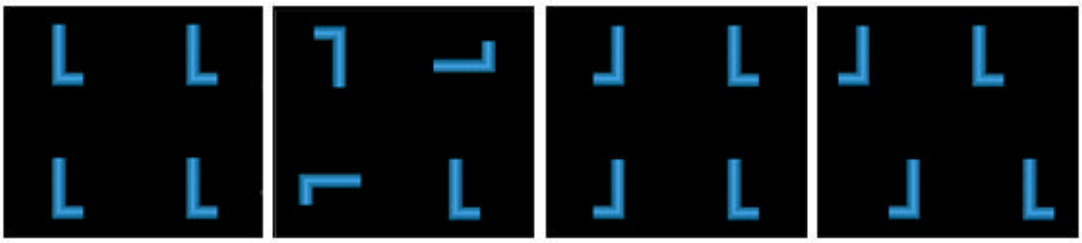
What makes a pattern interesting and visually pleasing is symmetry, which can be thought of, in this case, as geometric regularity. The dictionary defines symmetry as: "the correspondence in size, form and arrangement of parts on opposite sides of a plane, line or point." Symmetrical arrangements are of interest to a wide range of people in different disciplines, for various reasons. Among them, are mathematicians, interested in the mathematical rules that underlie the arrangement and formation of these patterns, chemists and crystallographers, interested in how molecules fit together, and artists and designers like M. C. Escher, who are attracted to the innate beauty of these arrangements and use their knowledge of them to design art, decorative fabrics and wall coverings.

What are the 17 Wallpaper Patterns?

The 17 wallpaper patterns, otherwise known as the 17 symmetries on a plane, are the finite group of symmetries possible on a 2 dimensional plane. In other words there are 17, and only 17, possible arrangements of elements on a 2 dimensional plane that can be repeated infinitely in a uniform (or symmetrical) way on that plane. There are other symmetries in other dimensions other than 2 dimensions, and these too have finite limitations. There are only 2 possible kinds of linear (or 1 dimensional symmetry) and 32 possible kinds of symmetry in a 3 dimensional space. (This piece restricts itself to one flavor of planar symmetries.) There are algebraic proofs of how and why this is so, but they are beyond the scope of this paper.

The 17 wallpaper patterns are formed by combining a very limited set (4) of possible transformations of elements (images, atoms, molecules, whatever) on one of 5 potentially symmetrical matrices (a square matrix, a rectangular matrix, a parallelogramatic matrix, a rhomboid matrix or a hexagonal matrix).

The 4 possible transformations that can symmetrically take place on these matrices (only certain ones on a given matrix) are: **translation**, the simple moving-over of an element, **rotation** of an element, **reflection** of an element and **the glide reflection** of an element (a combination of reflection and translation).



Naming Conventions:

The 17 different symmetries can be designated by several different naming conventions. These conventions are based on rather complex mathematical characteristics of the arrangement of the pattern's elements in relationship to each other and the lattices that they are arranged on.

The most accessible notation system is based on notation developed by the IUC (or International Union of Crystallography). There are other systems, including orbifold notation which is based on the mathematical discipline of topology, which studies those properties of geometric forms that remain invariant under certain transformations, such as bending or stretching. The crystallographic notation includes the following set of names for the 17 symmetries: {p1, p2, pm/p1m, pg/p1g, pmm/p2mm, pmg/p2mg, pgg/p2gg, cm/c1m, cmm/c2mm, p4, p4m/p4mm, p4g/p4gm, p3, p3m1, p31m, p6, p6m/p6mm}

This crystallographic notation consists of 4 symbols which identify the conventionally chosen cell or unit, the highest order of notation (or in other words, the highest number of possible rotations for that particular group), and the other fundamental symmetries (reflections and glide reflections).

The letter "p" or "c" denotes the cell (or unit) which can be "primitive" or "centered." A centered cell is chosen in 2 cases so that the reflection axis will be normal to one or both sides of the cell.

The numbers that sometimes follow the cell letter denote the highest order of rotation, or in plainer English, the number of repeated images rotated around each other in each, individual unit or cell. The "m" denotes a mirror, or reflection and the "g" denotes a glide reflection. A "1" denotes no symmetry axis for the reflection or glide

reflection of the preceding letter. (The bottom line is that it is that you can just ignore the names and have fun playing with the symmetry anyway.)

The Patterns that The Rotator Engine can make:

The Rotator Engine deals only with p4 symmetry (which is the four-fold rotation of elements about points on a square matrix). This is because it is based on the work of P. Dominique Douat (who's work I discuss in the third part of this section) who worked only with these elements. Rotations on a square matrix may not seem very complex, but the possible number of different arrangements of a grid of 16 x 16 individual motifs (which is what I used) is 4 to the 256^{th}. Not all of these many configurations are visually symmetrical or interesting but certain groups of rotations produce very aesthetic symmetries and this project attempts to tease those interesting ones out.

2. An interest in computer programming:

One of the attractions of using a computer to study patterns is because it can compute and set arrangements incomparably faster than any person could do manually, revealing design arrangements and their underpinnings that might not otherwise be apparent. In the study of pattern, as with many other things, computer programs are able do things that someone would otherwise have to do a much more tedious way.

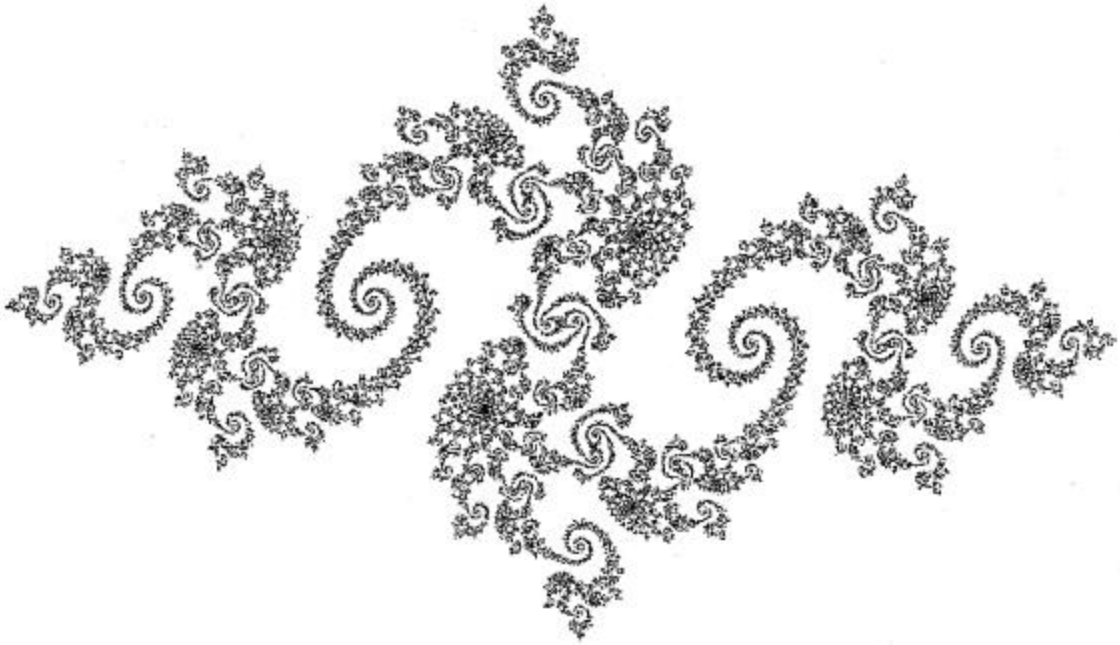
When I was in art school, trying to master the finer points of draughtmanship, I believed that the camera was the ultimate drawing. The camera designer, instead of

working to acquire the laborious skills that I was learning, instead invented an elegant and brilliant shortcut. It might take some doing to invent a camera, but once it is done, every picture from then on, is "drawn" for you. There's something incredibly beautiful in that. Computer algorithms open up a world of possibilities for doing this kind of thing. They can function with an elegance similar to that of the simple rules that form snowflakes or allow plants to grow into so many diverse and beautiful forms. They seem custom made for projects that allow designers to take alternate routes in the pursuit of their design goals.

When you program something you come to understand it on a different level, and working with the finished application can further that understanding by producing emergent behavior, or in this case a visual illustration that is created in an intuitive rather than a literal manner.

I cannot help but think of Gaston Julia who first experimented with the now eponymous Julia sets. Julia sets are the plotted graphs of a series of results of repeated calculations, where the x-axis are real numbers and the y-axis imaginary. Each graph point is drawn black or white based on the input and whether or not the results of the calculations are finite. (Julia sets are now known to be parts of the massive and extremely popular Mandelbrot set.) Julia never lived to see the beauty of the complete visually plotted results of his mathematical experiments because he had no recourse to computers. Though he did extensive calculations of his equations, the sets were simply too massive for someone to calculate in one lifetime. What would he have thought had he seen these visual representations?

A Sample Julia set:



Fractal illustrations are, of course, but one example of the beautiful visual expositions that the use of algorithms can generate. The L –System work of Aristid Lindenmayer is another good example of using computers to generate interesting imagery based on simple mathematical rules and there are many more.

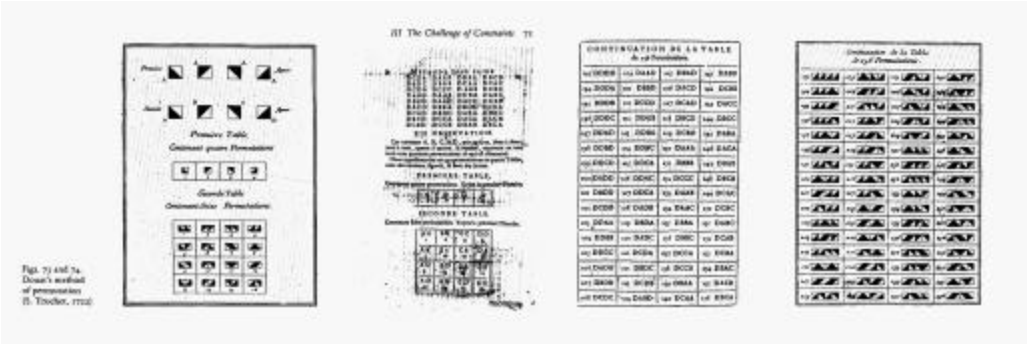
Specific subject of study:

The specific idea for this project came out of some reading I was doing in a book by E. H. Gombrich called "*The Sense of Order*." In the book Gombrich discusses a treatise by P. Dominique Douat with the lengthy title*: Methode pour faire un infinite de desseins differents avec des carreaux mi-partis de deux couleurs par une Ligne diagonale, ou observations du P. Dominique Douat Religiuex Carme de la Province de Toulouse sur un memoire insere dans l'histoire de l'Academie Royale des Sciences de*

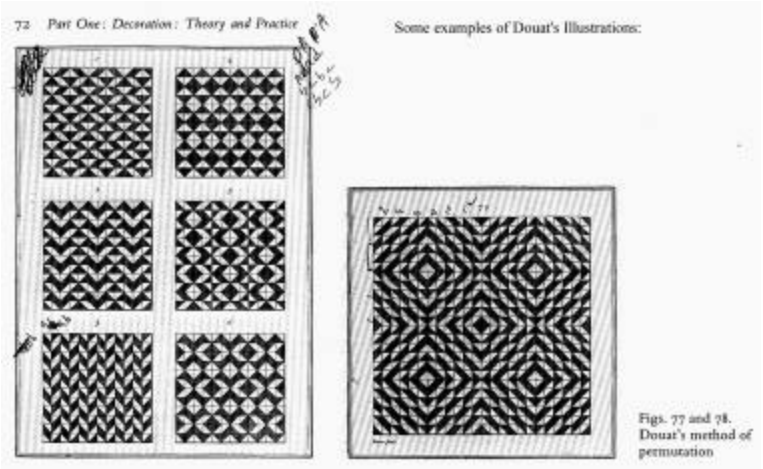
Paris 'annee 1704, presente par R. P. Sebastien Truchet Religieux du meme order, Academicien honoraire (Paris, 1722). For the sake of brevity, I'll call it "Douat's Permutations."

Douat's Permutations, or what I could find on it. (there is very little available information) consisted of tables of different arrangements of diagonally divided squares rotated in one of 4 ways. There were some illustrations of these in the book and I was amazed at the variety of designs in just those few examples and wondered what others were possible. The original idea for this piece was simply to make an engine that could explore the geometric possibilities of this simple family of forms. What evolved out of it, was the idea to make an engine that would make a variety of patterns.

Douat's charts of Permutations:



Some examples of Douat's Illustrations:



Figs. 77 and 78. Douat's method of permutation

How the project was built:

An application always has two, often conflicting, design issues: the construction of the interface and the structure of the underlying program. I audited an interface design course at NYU's graduate computer science department and discussion of that conflict of interest between functionality and usability came up in almost every class. The final consensus is though, that usability comes first and the onus is on the programmer to make the application work in the context of its being intuitive and easy to use.

Design considerations:

One of the big challenges with the Rotator Engine was fitting everything into a manageable space. There was the creeping featurism aspect, as well as the fact that the more motif units I could fit in, the more visually interesting the piece would be. That had to be weighed against clutter and the fact that if the motifs were too small they would become less visible. The memory limitations and speed of Macromedia Director (which the project is implemented in) also effected the design decisions. This is one reason that I think the application could benefit from being re-written in another, faster language.

The result, as often is the case, was a compromise. In this case, the compromise was between features vs. simplicity. A balance had to be effected between openness (being able to see all your options at glance) and visual clutter. In the end I limited myself to two rows of buttons and made choices between features when the rows became full. I originally had separate windows that opened with the program and stayed open, but ended up changing the design to windows that popped up when a button was clicked in

order to save space, eliminate clutter and allow the project to run “as is” on the web. I tried to keep buttons that did similar things close to each other.

I needed an even number of motif units so the design would be symmetrically balanced, and in fact having a number of units that was a multiple of four made the symmetry work best. I settled on a 16 x 16, two layer grid which made the motifs a little small but allowed much more symmetry and was manageable (barely) given the memory considerations.

I found the limited user testing I did to be invaluable and was very glad to have done it early on. A big issue was that it was hard for the users to understand how the program updated when they clicked a square and a symmetry button was turned on. I had to change it from the user choosing which quarter of the matrix was the “leading” quarter to having whichever quarter they clicked in become the leading quarter automatically. Later that seemed painfully obvious.

I also found the design of the individual motifs to be surprisingly challenging. Douat was smart to choose the diagonally divided square as he did, because it really highlights the patterns that the rotations produce. One of the first additions to my original idea though, was the interchangeability of motifs. When I tried other designs, I found that many of them were either already symmetrical (so that rotating them made no visual change) or that they did not line up visually, and so looked bad when they were rotated in relation to each other. I would have liked to have made the designs with vectors instead of bitmaps but Director doesn’t handle vectors particularly well and ground to a virtual halt, when I tried to use them.

Programming considerations:

I did my project in Macromedia Director because I knew it well. I wanted to be able to concentrate on the application and the more complex aspects of the code and not have to worry too much about syntax.

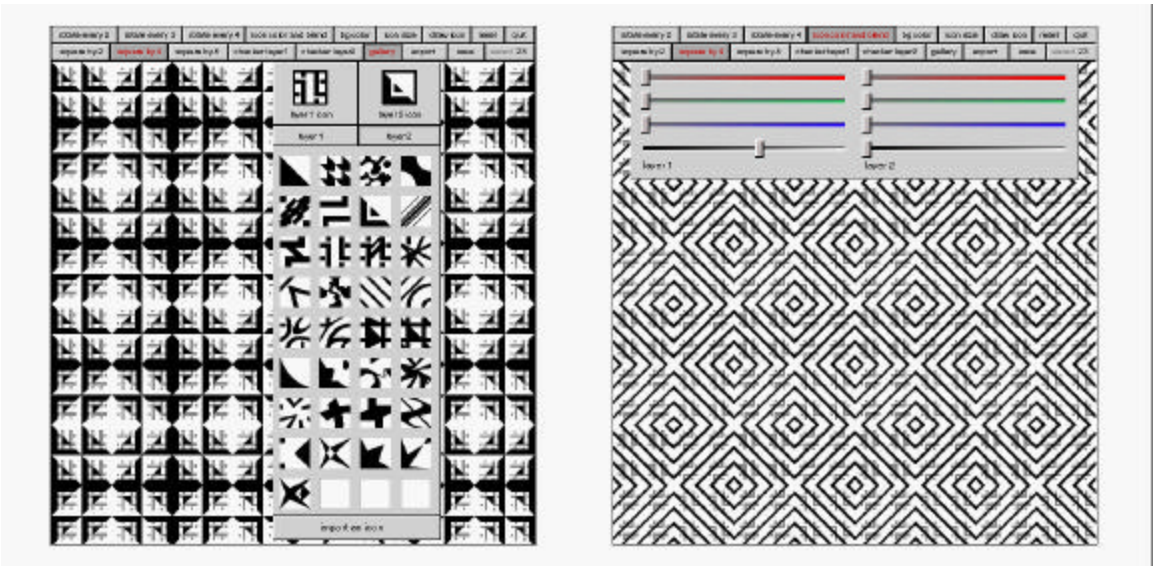
The first thing I did with this engine was to make each, individual square on my square matrix rotate 90 degrees when it was clicked. This already made the generation of patterns much faster than if rotation had to be done by hand in the real world or in Photoshop. Then I made buttons that would rotate groups of squares, and then buttons to set things into symmetrical patterns and update those patterns when individual squares were clicked. This was difficult as I was having problems with the program choosing which quarter of the matrix to follow and changing things that were already correct, based on other changes that had happened. I ended up resolving this by having the application change a 2-D array that held degrees of rotation and then updating the rotations based on the array.

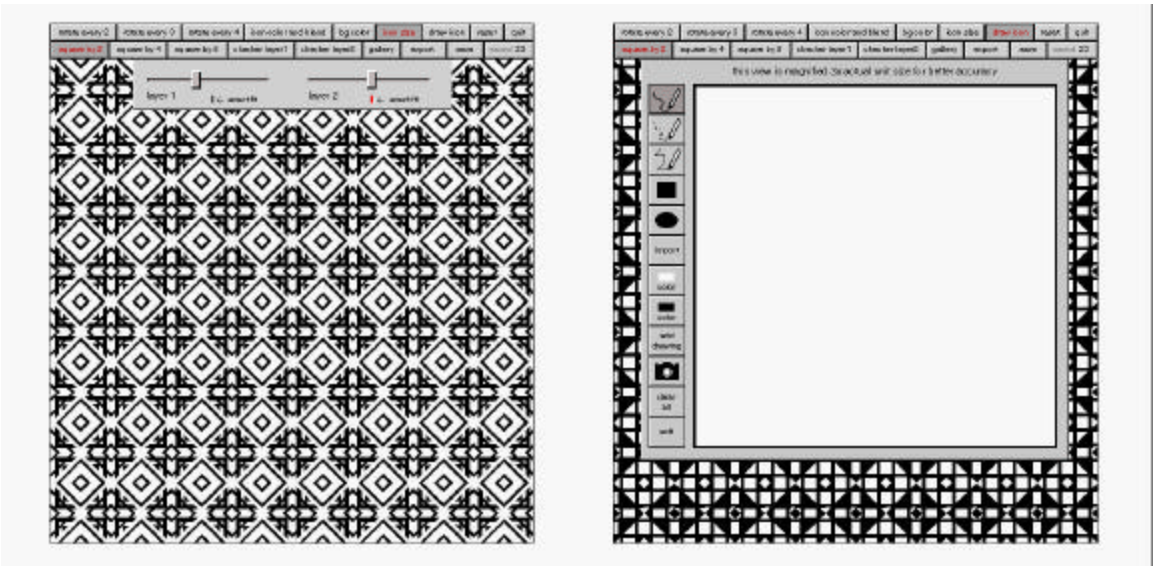
The “save” function was difficult because I had to make the program hold onto results after shutting down. I was able to write a list of commands to a cast member that endured, and after some challenges with parsing out the commands from the list, I was able to make it work using the “do” function.

The final major programming challenge was to get the image into and out of the machine’s clipboard so that it could be used in Photoshop or another imaging software. This ended up being easy when I found that Director already had a functions that did that, copyToClipboard() and pasteClipboardInto().

I had to keep track of the states of many different elements to accommodate the design since I had opted for having many pop-up windows, but this was more tedious than difficult.

Screen shots





Instructions for Using the Rotator Engine

To rotate and individual square, click on it.

There are 2 layers. You cannot see this until you change the motif of at least one layer using the "gallery" button.

Description of button functions:

1. "rotate every 2" does just that. It rotates every 2 squares.
2. "rotate every 3" rotates every 3 squares.
3. "rotate every 4" rotates every 4 squares.
4. "icon color and blend" allows you to adjust the blend and color of the icons/squares on each of the 2 layers.
5. "bg color" allows you to adjust the background color.
6. "icon size" allows you to make the icons/squares bigger or smaller. When "exact fit" is lit red, it means that your icons are their original size, touching but not overlapping.
7. "draw icon" allows you do draw your own image/motif. You may prefer to import your own from Photoshop though. This can be done both in "draw icon" and in "gallery."
8. "reset" returns the application to its original state.
9. "quit" closes the application.
10. "square by 2" makes every unit of 4 symmetrical. It will update when

you click on individual squares as long as it is depressed.

11. "square by 4 makes every unit of 16 symmetrical. It will update when you click on individual squares as long as it is depressed.
12. "square by 8 makes the entire pattern symmetrical. It will update when you click on individual squares as long as it is depressed.
13. "checker layer 1" makes every other square in layer 1 invisible.
14. "checker layer 2" makes every other square in layer 2 invisible.
15. "gallery" gives you a choice of motifs for each layer and allows you to import your own motifs from the clipboard with the "import an icon" function at the bottom.
16. "export" lets you save the patterns you make into the clipboard.
17. "save" allows you to save the patterns you make within the application so that you can go back and review them with the "saved" button.
18. "saved lets you toggle through all the patterns saved to date in the application.

Conclusion:

Pros:

I think that the Rotator Engine is fun to play with and in some senses, educational. I learned a great deal making it, both about patterns and programming which was a big part of my goal in making it.

Cons:

Where I think it falls short is that the images are crude and the program has to be used outside of a mainstream image processing software. It is also a little slow and the transformation options are limited due to it being based on Douat's work.

If I were to do it over, I would base the operations on all of the 17 symmetries on a plane. I would allow the user to choose between matrices. I would allow them to import any kind of image they wanted and have the output resolution be device independent.

Resources:

Books:

Flake, Gary William. *The Computational Beauty of Nature*, MIT Press, Cambridge, Massachusetts, 1998

Ghyka, Matila. *The Geometry of Art and Life.* Dover Publications Inc., New York, 1977

Glassner, Andrew. *Andrew Glassner's Notebook*, *IEEE Computer Graphics and Applications Magazine*, July/August 2000

Gombrich, E. H. *The Sense of Order, A Study in the Psychology of Decorative Art*. Ithaca, New York: Cornell University Press, 1979

Prusinkiewicz, Przemyslaw, *The Algorithmic Beauty of Plants.* Springer-Verlag - New York, New York, 1996

Weyl, Hermann. *Symmetry*, Princeton University Press, Princeton, 1952

Websites:

http://www.geom.umn.edu/java/Kali/welcome.html

http://www.geom.umn.edu/

http://aleph0.clarku.edu/~djoyce/home.html

http://www.geom.umn.edu/docs/doyle/mpls/handouts/handouts.html

http://www.geom.umn.edu/docs/reference/CRC-formulas/node12.html#wallpaper1b

http://www.ics.uci.edu/%7Eeppstein/junkyard/tiling.html

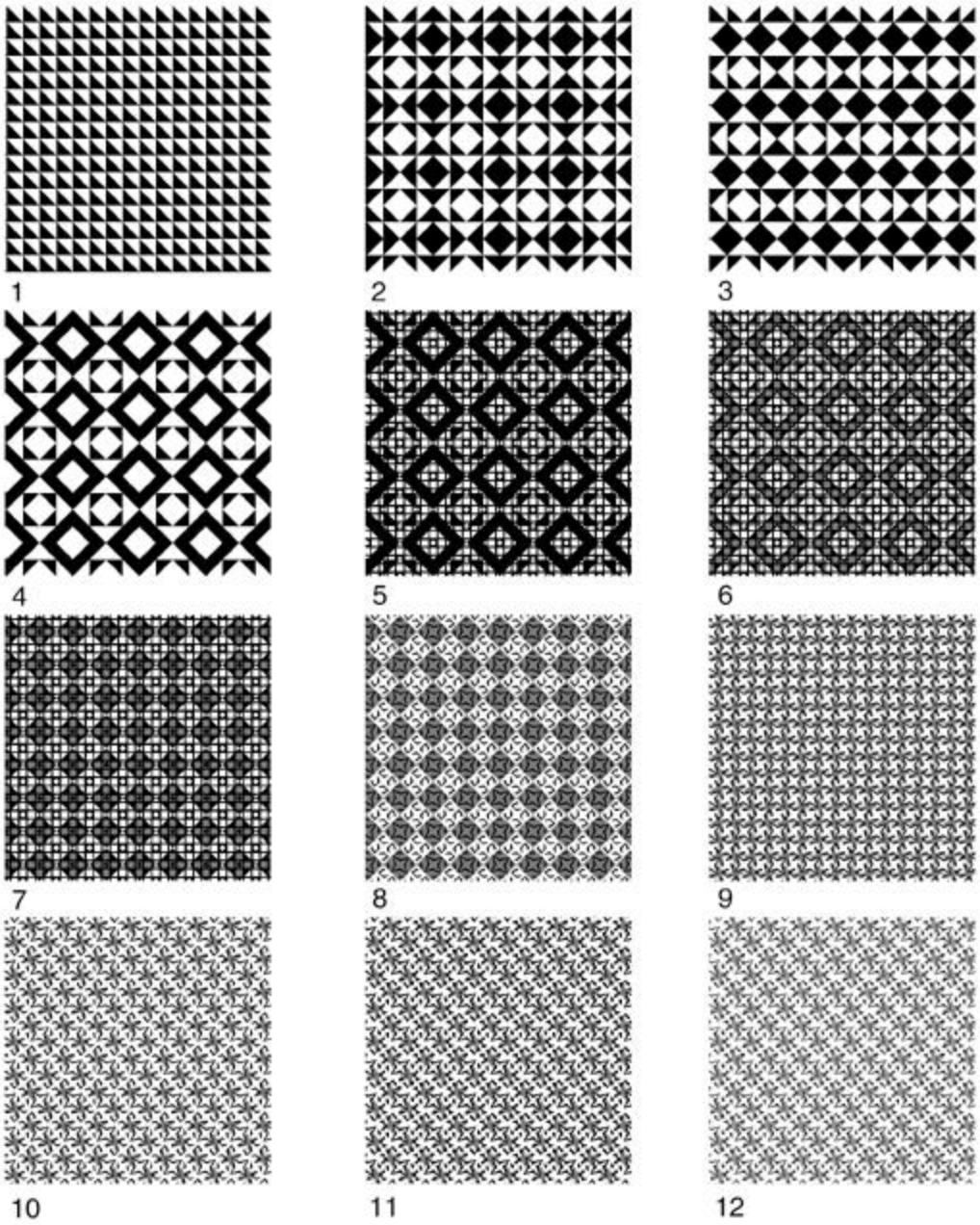
http://xahlee.org/Wallpaper_dir/c0_WallPaper.html

http://library.thinkquest.org/16661/

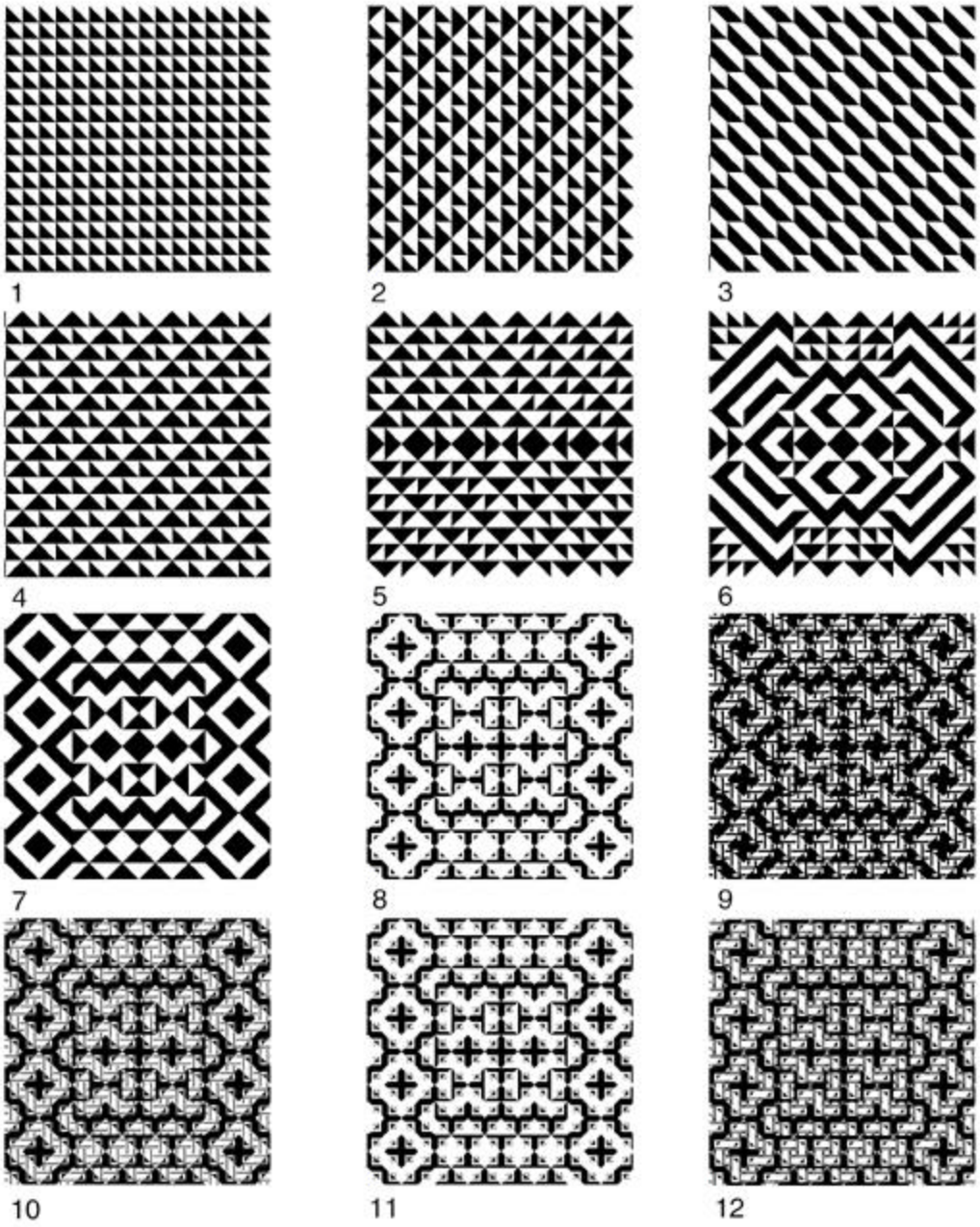
http://www.lectra.com/en/solutions/produits

Some Example Patterns Made With the Rotator Engine

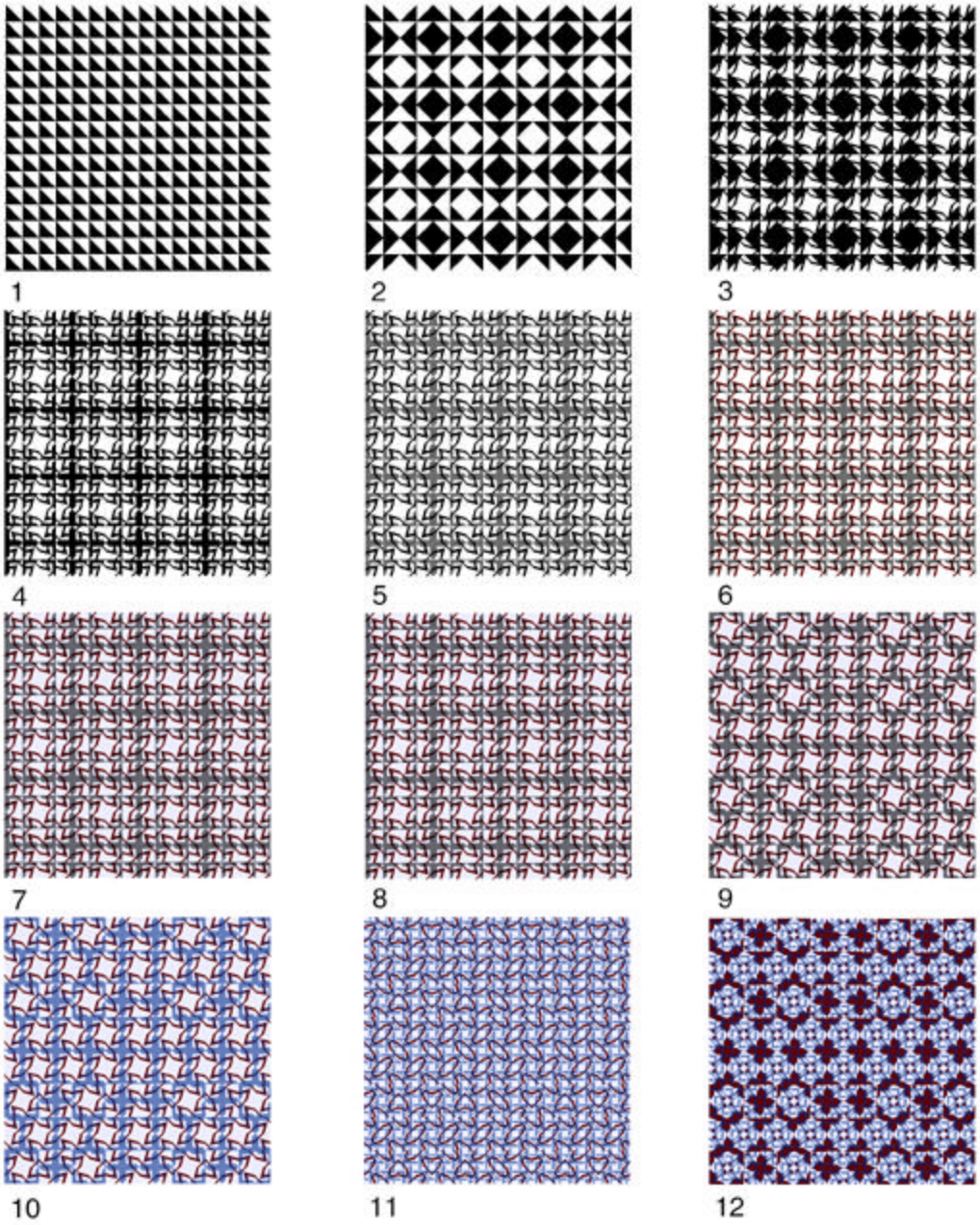
Pattern series created with the Rotater Engine /#1



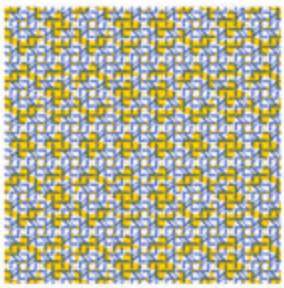
Pattern series created with the Rotater Engine /#2

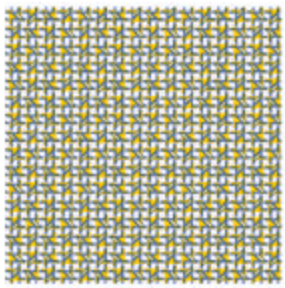


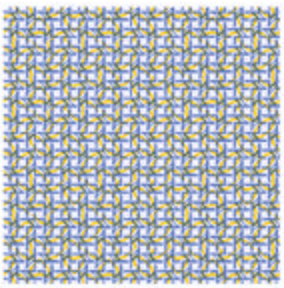
Pattern series created with the Rotater Engine /#3 p1

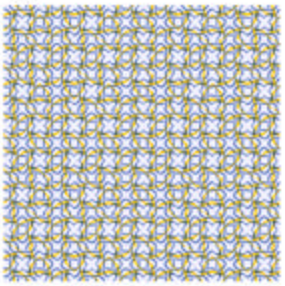


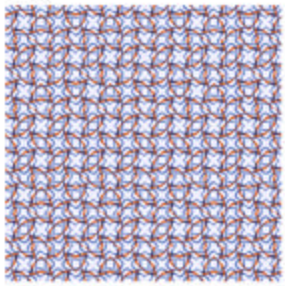
Pattern series created with the Rotater Engine /#3 p2

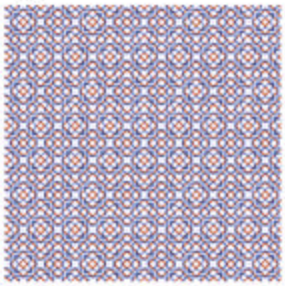

13

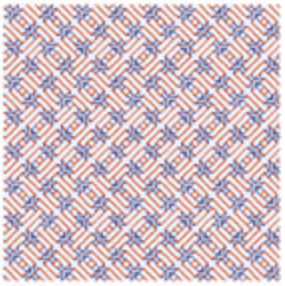

14

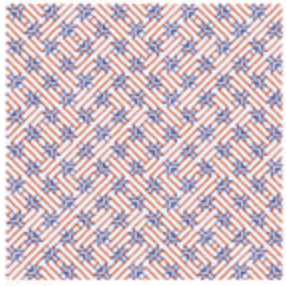

15

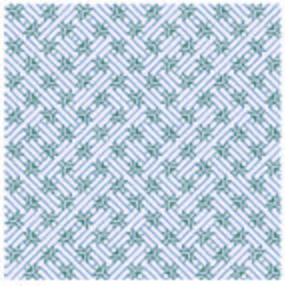

16

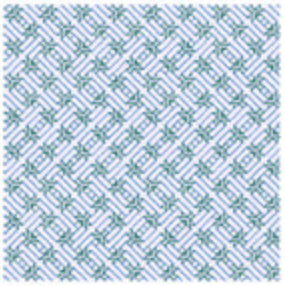

17

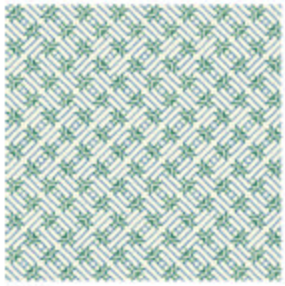

18

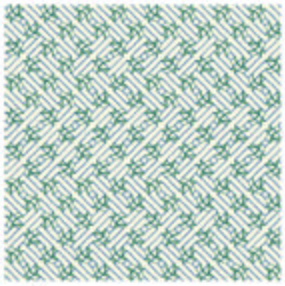

19


20

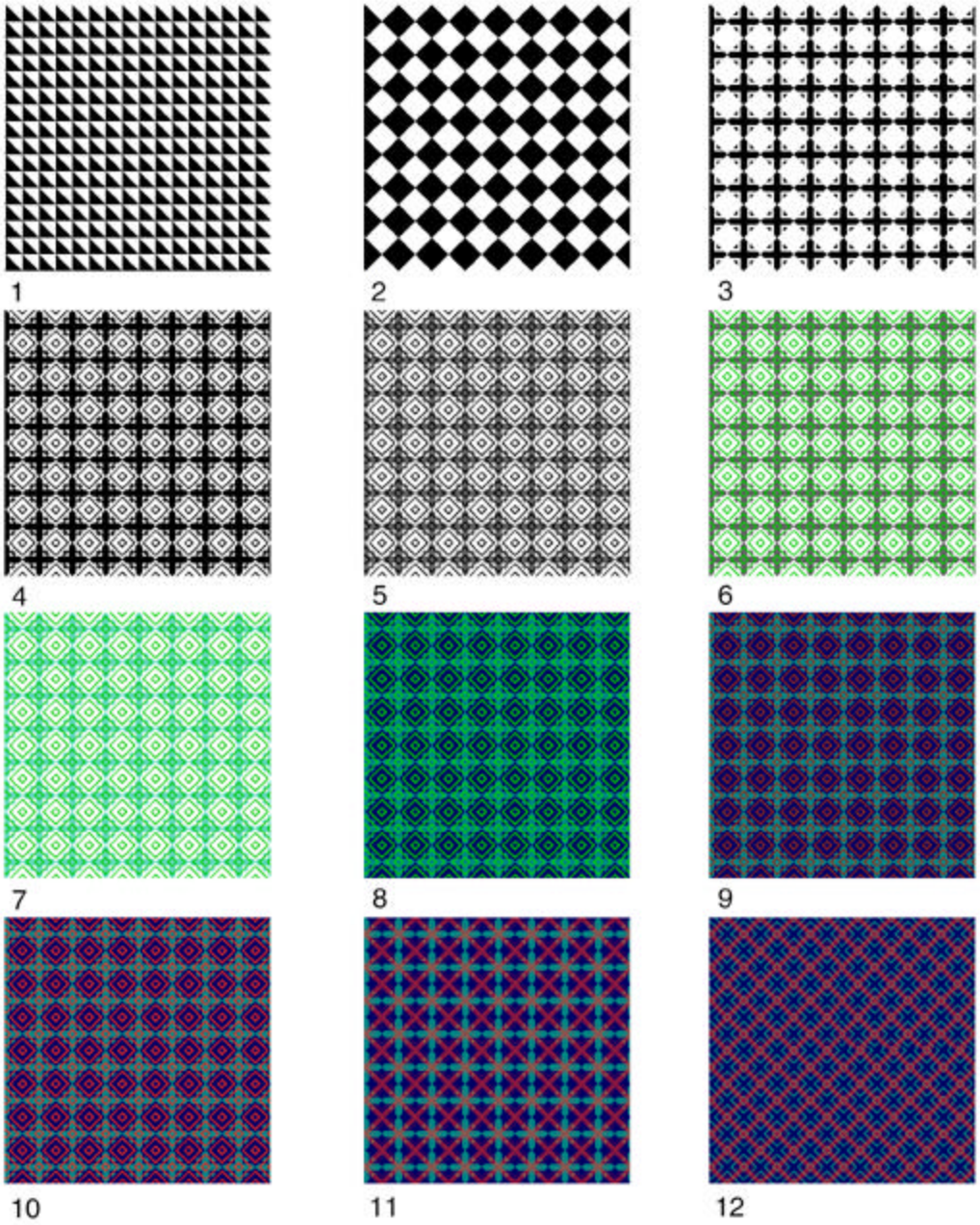

21


22

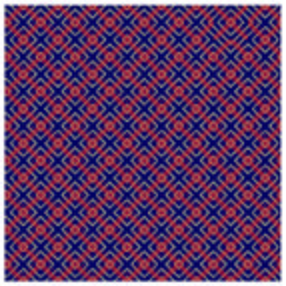

23

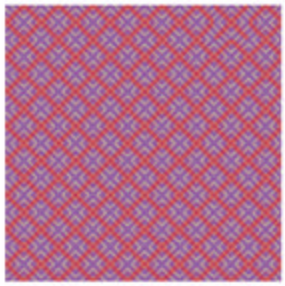

24

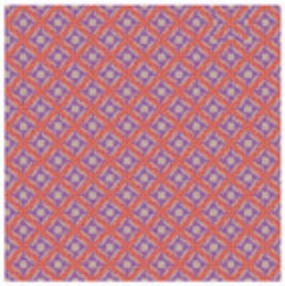
Pattern series created with the Rotater Engine /#4 p1

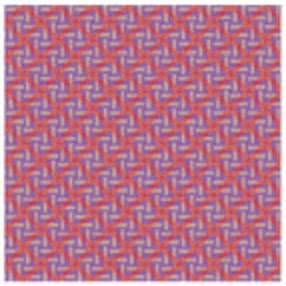


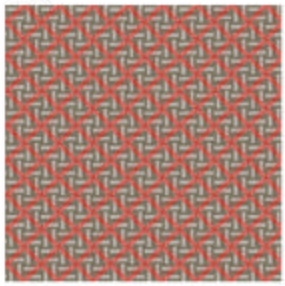
Pattern series created with the Rotater Engine /#4 p2

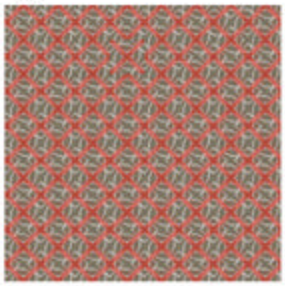

13


14

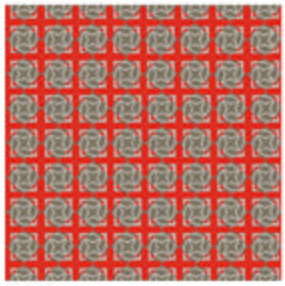

15

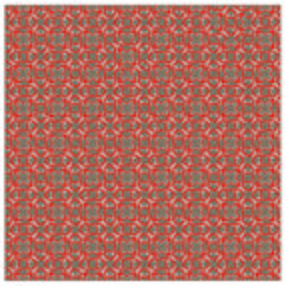

16

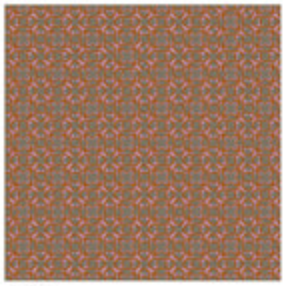

17

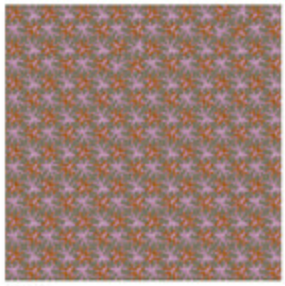

18

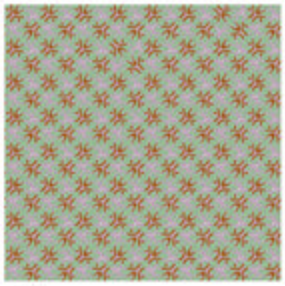

19


20

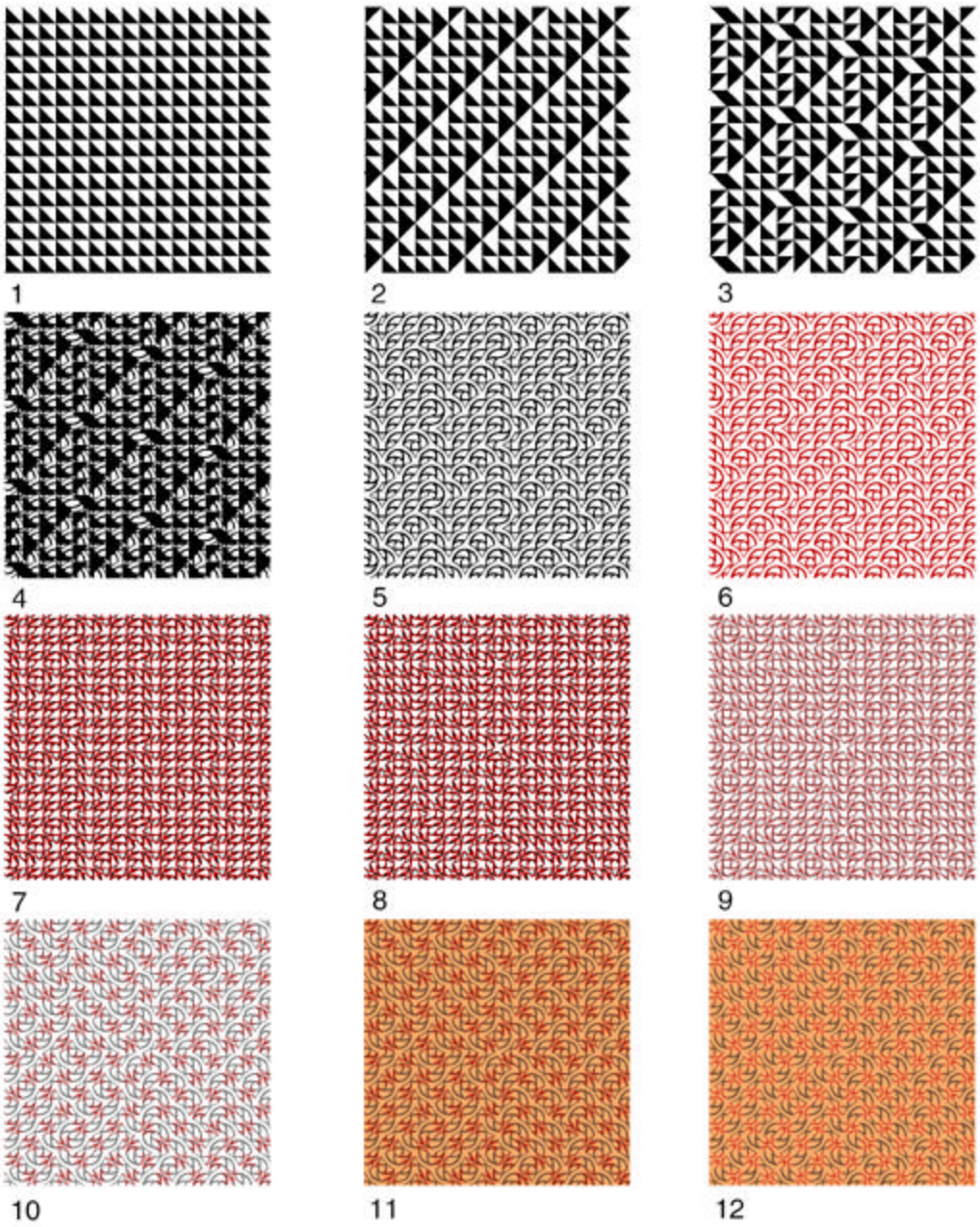

21


22


23


24

Pattern series created with the Rotater Engine /#5 p1



Pattern series created with the Rotater Engine /#5 p2