Then Try This • Algorithmic Pattern Salon

An App for Algorithmic Pattern Generation

Stuart Smith

Then Try This

Published on: Nov 10, 2023

URL: https://alpaca.pubpub.org/pub/9w7bluoj

An App for Algorithmic Pattern Generation

Stuart Smith

Departments of Music and Computer Science

University of Massachusetts Lowell

stu@cs.uml.edu

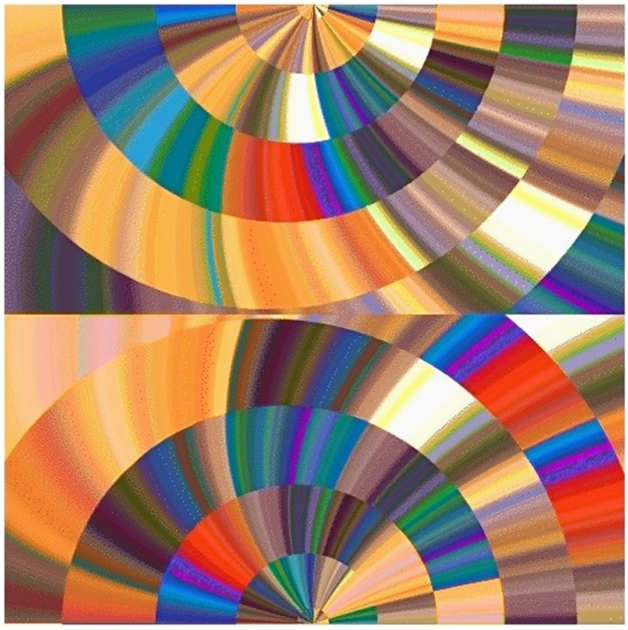

Figure 1
Image created by the app

Overview

This paper is addressed primarily to digital artists who want to create their own custom tools. I describe an algorithmic pattern generator app I've been developing. My goal has been to create an app that would implement the following four-step process:

1. Generate or select a source image.
2. Apply a sequence of one or more transformations to the source image.
3. Display the resulting image and, optionally, make a video from a sequence of images.
4. Select the final images.

The app can perform all four steps either autonomously or under user control. To create a set of images with the app, it is necessary only to fill in one or two text fields on the GUI: the path to a source image and a few numbers that specify a sequence of transformations to be applied to the image. A single mouse click initiates the generation of a run of images. If desired, the images can be saved as frames of a video that will be made automatically at the end of a run.

Example of a single cycle of image generation

Every cycle of image generation begins with a source image. This is usually computer generated, but photographs, manually created images, and live webcam snapshots can also be used. Figure 2 consists of parallel lines of random colors and widths.

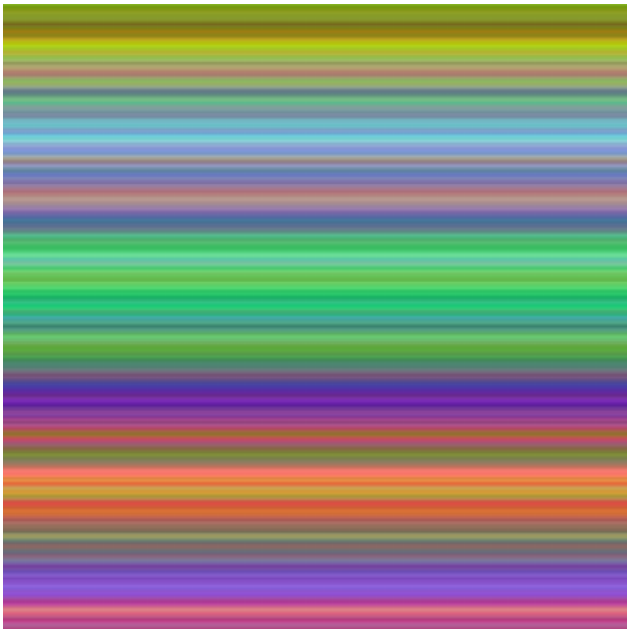

Figure 2
Computer-generated source image

The next step, shown in Figure 3, demonstrates the effect of permuting the R, G, and B channels of the source image. This process often has the effect of increasing contrast and vividness of color, which is why I used it here.

Figure 3
Random channel permutation

The next transformation, shown Figure 4, is one of the simpler ones available in the app. Small square replicas of Figure 3 are reassembled into a coarse “weave” pattern.

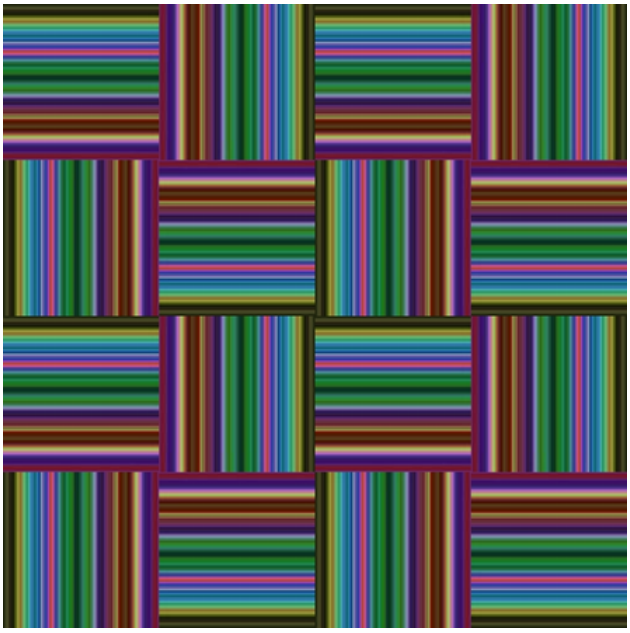

Figure 4
Weave pattern

The original pattern of Figure 3 is in the upper left quadrant of each group of four replicas. Successive quadrants going counterclockwise are rotated 90^{o}, 180^{o}, and 270^{o}, respectively. This simple transformation is followed by one of the most complex, “inversion,” as shown in Figure 5.

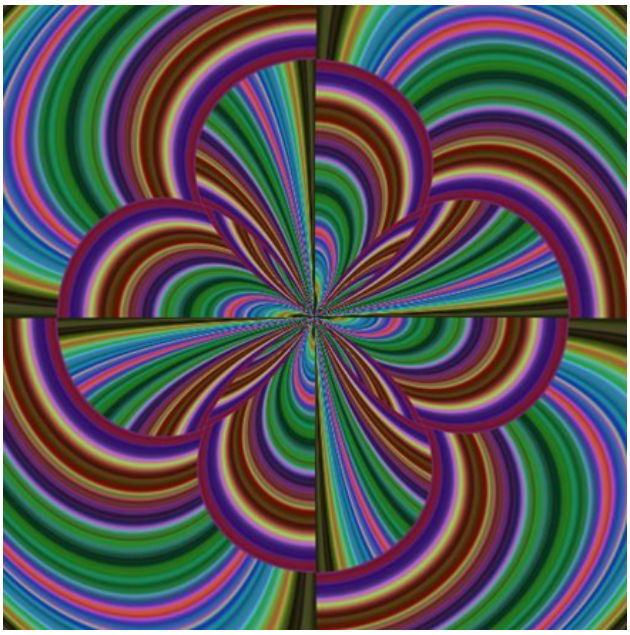

Figure 5
Image “inversion”

“Inversion” here denotes a two-part image transformation [1]. First, the image coordinate system is changed from cartesian to polar: the X and Y coordinates of each pixel are reinterpreted as angle and radius with respect to the origin, which is in the center of the image. Second, pixels near the edge of the image are moved in towards the center, while pixels near the center are moved out towards the edges. This operation in effect turns the image inside out.

The next transformation, “distortion,” is shown in Figure 6.

Figure 6
Sinusoidal distortion

The distortion operation displaces pixels in both the horizontal and vertical directions in a sinusoidal pattern. The amount of distortion in each direction is different, causing the somewhat irregular appearance of the resulting image.

The next filter to be applied recolors the image. The app offers two ways to recolor an image:

1. Select a color map from over 200 available in the app [2].

2. “Borrow” the palette of another image and apply it to the current image.

Figure 7 was recolored by the borrowing method.

Figure 7
Recolor

The recolor operation here “borrowed” the palette of a still life by Cezanne. No picture content—not a single line or brush stroke—was copied from the Cezanne. The operation simply imposes the color scheme of one image on another image.

Recoloring is often the last step in the process of image generation; however, in this case I wanted to do just a bit of post-processing to enhance the image, if possible. Figure 8 shows the result of applying the “de-hazing” filter to Figure 7. The de-hazing operation reduces atmospheric haze in a color or grayscale image. Here it’s used to make the colors of the image more vivid.

Figure 8
Final image with de-hazing applied

One additional step is sometimes desirable: applying a filter provided by an external app to increase contrast or vividness. I've found the GMIC-Qt "Artistic" and "Details" filters available for GIMP especially good for this.

Fabric and wallpaper designs

I originally envisioned my app as a tool for making colorful images of complex curved figures; however, it's easy to make the more regular patterns typical of textile prints and wallpaper. The easiest way is simply to apply the "wallpaper" filter to an image like Figure 8. This filter makes the image into a tile, which is then repeated and assembled into a 4×4 block of the pattern. The following are some typical examples of designs generated by the app in this way (Figures 9-16):

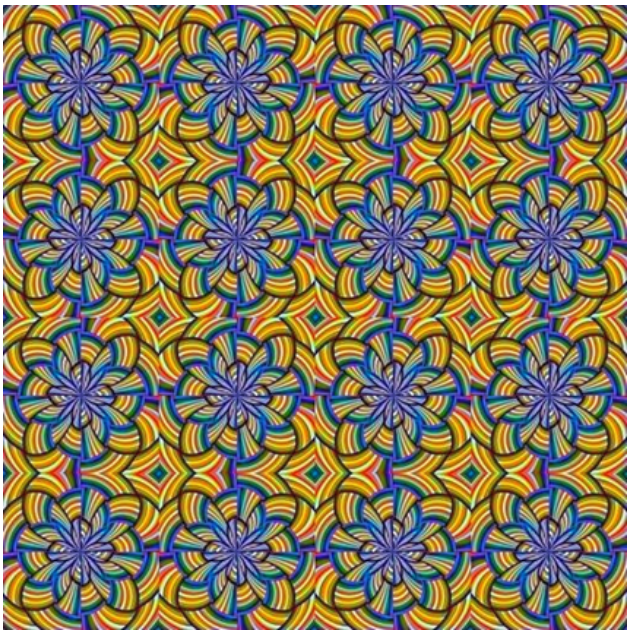

Figure 9

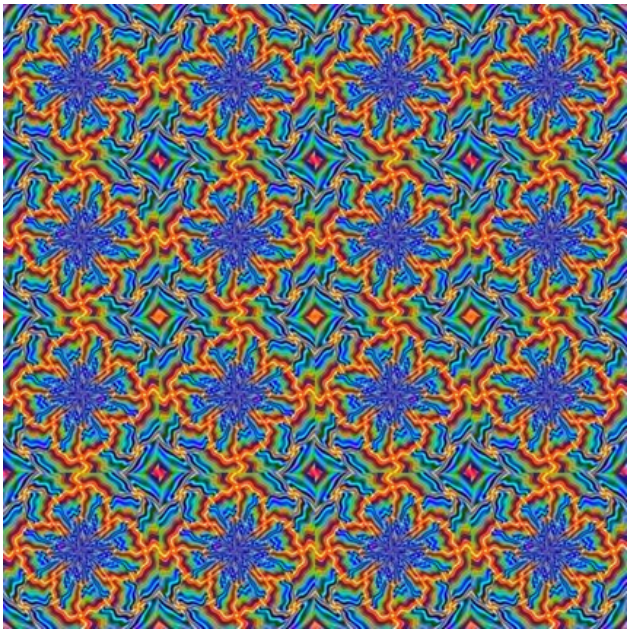

Figure 10

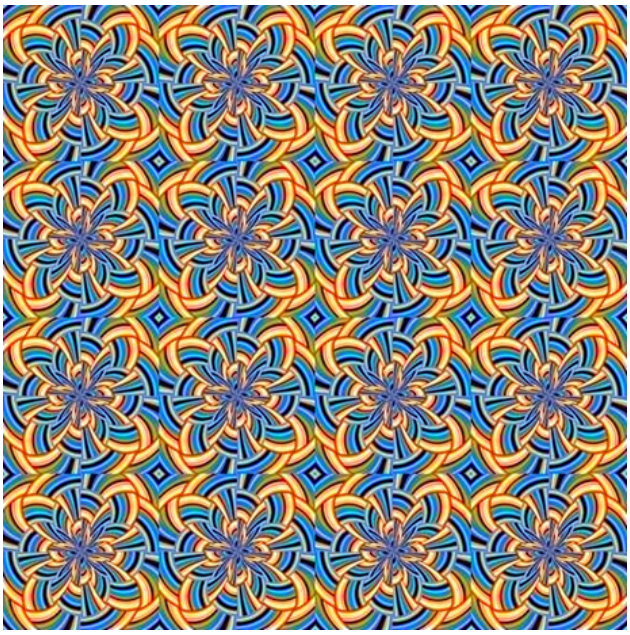

Figure 11

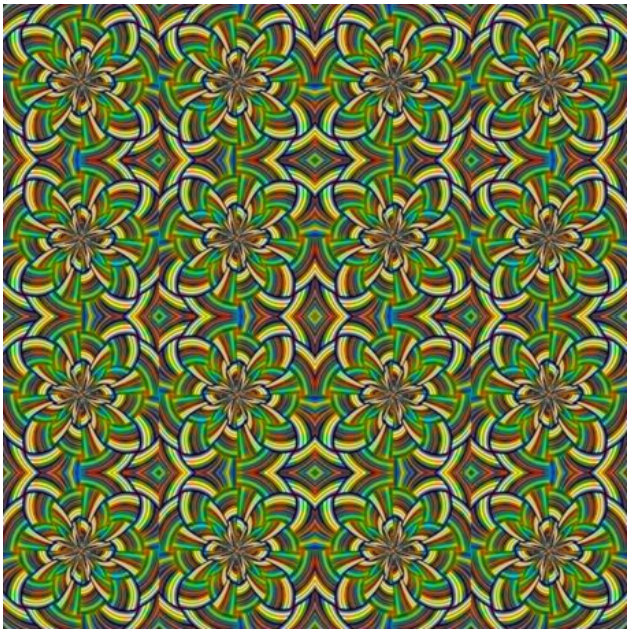

Figure 12

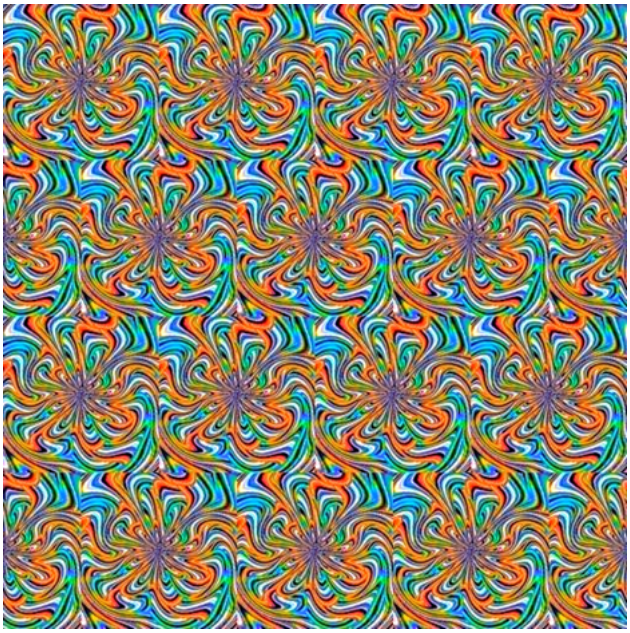

Figure 13

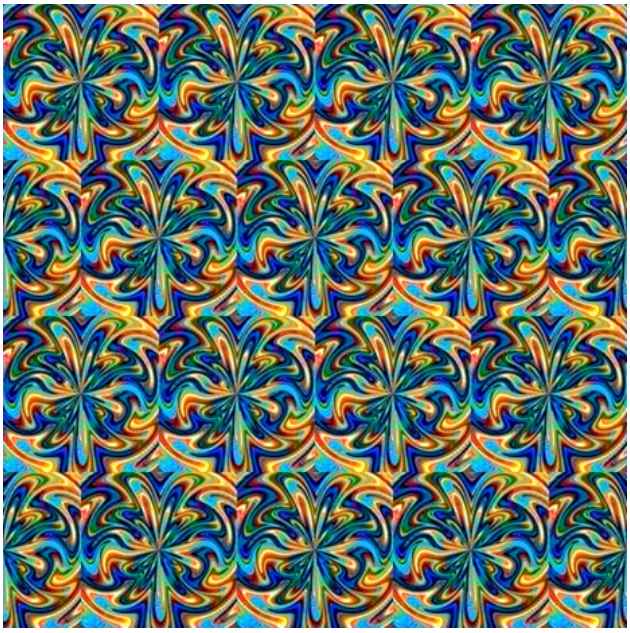

Figure 14

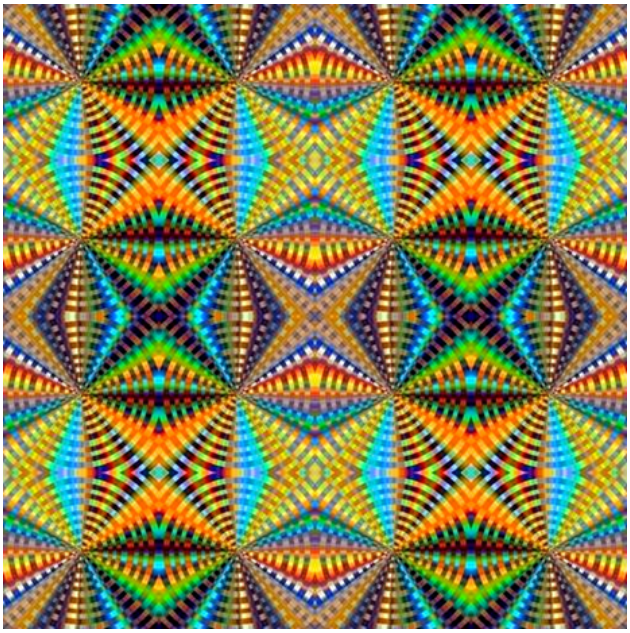

Figure 15

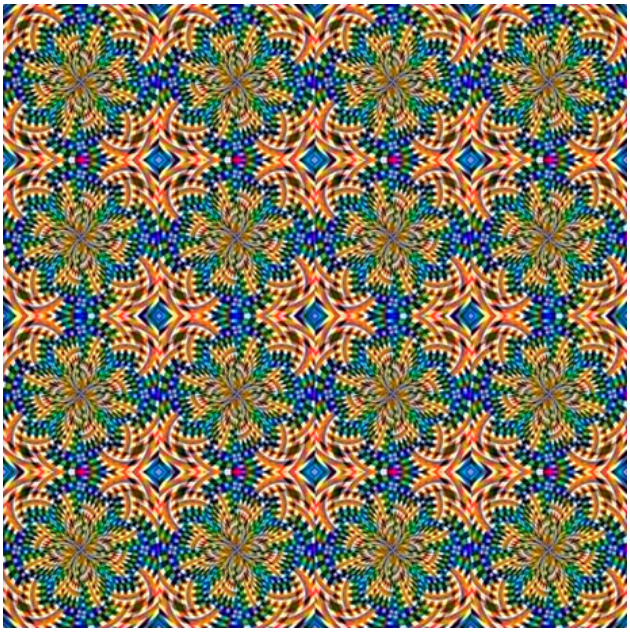

Figure 16

Operation of the app

Creating an algorithm is mostly a matter of setting up a sequence of filters. Any number of filters can be applied in any order, and repeats are allowed. There are 32 “Presets,” small, pre-programmed sequences of filters that generally create pleasing images. A Preset is selected with a single mouse click.

It takes the app on average 7 seconds to generate a 512×512-pixel image on my home system. Each time the size of an image is doubled, execution time goes up by roughly a factor of 4. The largest currently practicable image is 2048×2048 pixels, which can take from one to two minutes to generate. With images of this size, on a large format printer the dimensions of a print are 21”×21” (53 cm×53 cm), not including the border.

The main reason for the apparently long average time it takes to generate a single image is that certain of the most-used filters (e.g., pincushion distortion and inversion) require that the position of every pixel be recalculated and that appropriate interpolations be done to fill in areas that did not receive updated pixels. By contrast, simple transformations such as flipping an image around an axis of symmetry can be done at video frame rates.

Because all of the filters take one or more random inputs, the exact nature of output images cannot be known in advance. As a result it generally requires a run of 20 to 30 images to get one image worthy of being saved and displayed. The app has a crude accept/reject capability [3], which can automatically weed out unusable images (e.g., visual noise, all-black or all-white images, images with very low contrast, etc.) The final selection from the remaining images must be done by a human who evaluates them one at a time.

More about the app

The app was built partly using powerful data visualization and image processing tools provided by Matlab. A similar app could be developed using other popular programming languages, such as Python [4,5]. The app is presented here not as an innovative tool ready for immediate use by artists but rather as an example of what an artist can accomplish by bringing together capabilities not originally designed for use in making art. It is a work-in-progress that is easily modified to allow changes and improvements that suggest themselves while in the process of making digital art. Despite the app’s origins in scientific/technical pursuits, the aesthetic of its images is distinctly Modernist.

Extending the app

The app currently generates images with 2-fold or 4-fold symmetry. This is because both the structure of computer memory and the coordinate systems of most computer graphics software are orthogonal. Thus it’s easy to create, save, and display square, rectangular, and rhombic images. To obtain images with 3-fold and 6-fold symmetry I’ve used Artlandia’s “Symmetry Mill” [6] app for post-processing some of my images. Symmetry Mill allows me to tile the image plane with regular hexagons and equilateral triangles. Here are a few examples (Figures 17-20):

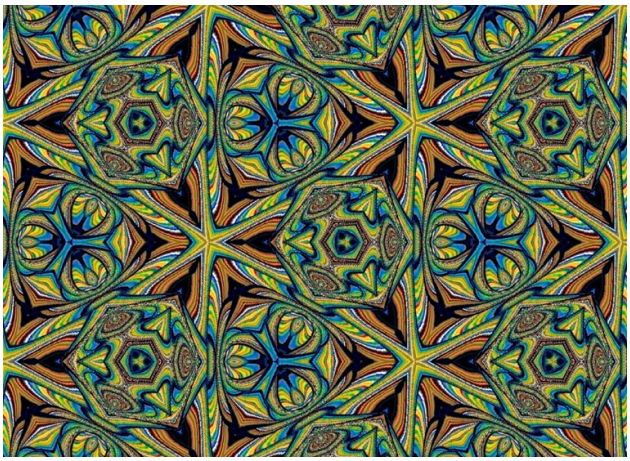

Figure 17

Figure 18

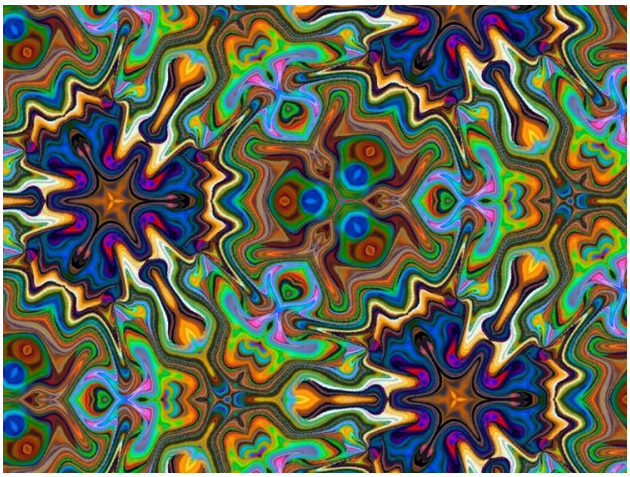

Figure 19

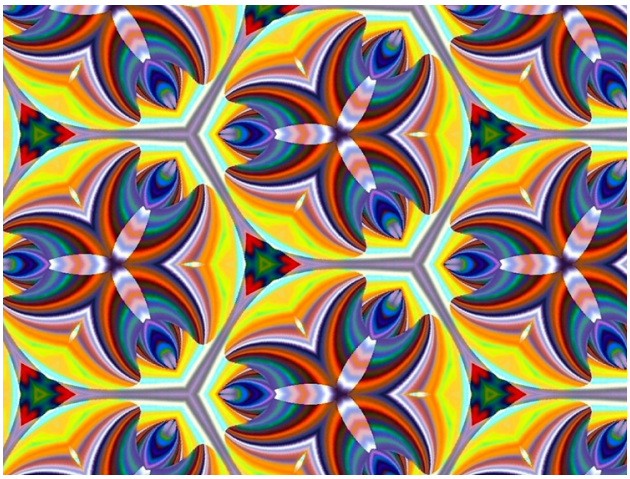

Figure 20

I plan to implement 3-fold and 6-fold symmetry transformations in the app in the near future.

Why MATLAB?

The core Matlab language includes 2-D and 3-D plotting functions to visualize data and communicate results. The add-on Image Processing Toolbox provides a wide range of operations, including image transforms, edge detection, smoothing and blurring, color manipulations, common geometric transformations, and creation of videos. These capabilities can be augmented by user-written functions and by packages available from the Matlab File Exchange [7,8]. Matlab's built-in "App Designer" makes it easy to bring all of these capabilities together to make a unified application. The user interface (Figure 21) was created with App Designer by simply selecting a widget from a menu, dragging it over to the background, and dropping it in the desired place. Text was entered into the fill-in form associated with each widget.

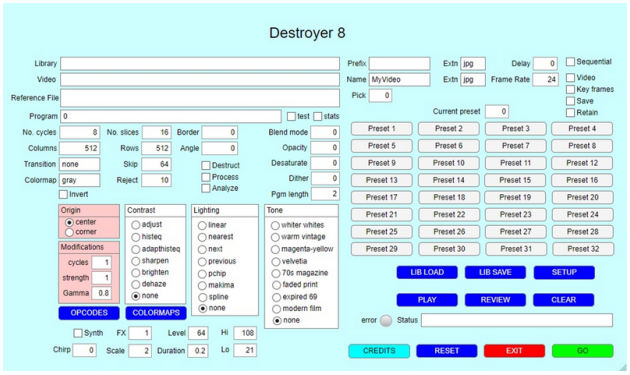

Figure 21

Matlab is not free, but a "home" license for the full language can be purchased as of this writing for USD $149 + tax. Add-ons such as the Image Processing Toolbox are usually priced at USD $45 each.

This app vs. other tools

The app provides many capabilities one would expect to find in any image manipulation application. It can, for example, perform the following operations just as in GIMP, Krita, or Photoshop:

- blend and composite images as layers
- adjust contrast
- add lighting effects
- apply various geometric filters

Perhaps the app's most distinctive feature is its ability to create source images: there are 24 different algorithms for generating source images, plus the ability to use image files of many types as well as live images from a built-in camera.

References

[1] MIMT: Matlab Image Manipulation Toolbox. https://www.mathworks.com/matlabcentral/fileexchange/53786-image-manipulation-toolbox?s_tid=srchtitle 2021. Accessed 12/25/2022.

[2] Woodford, O. J. SC-Powerful image rendering. https://www.mathworks.com/matlabcentral/fileexchange/16233-sc-powerful-image-rendering?s_tid=srchtitle 2020. Accessed 12/25/2022.

[3] Machado, Penousal and Amilcar Cardoso. All the Truth About NEvAr. *Applied Intelligence* 16: 101-118. 2002.

[4] https://www.knowledgehut.com/blog/business-intelligence-and-visualization/python-data-visualization-libraries 2022. Accessed 12/31/2022

[5] Image Processing in Python: Algorithms, Tools, and Methods You Should Know. https://neptune.ai/blog/image-processing-python 2020. Accessed 12/25/2022.

[6] https://artlandia.com/products/SymmetryMill/ accessed 10/23/2023

[7] MIMT: Matlab Image Manipulation Toolbox. https://www.mathworks.com/matlabcentral/fileexchange/53786-image-manipulation-toolbox?s_tid=srchtitle 2021. Accessed 12/25/2022.

[8] Woodford, O. J. SC-Powerful image rendering.
https://www.mathworks.com/matlabcentral/fileexchange/16233-sc-powerful-image-rendering?s_tid=srchtitle
2020. Accessed 10/22/2023