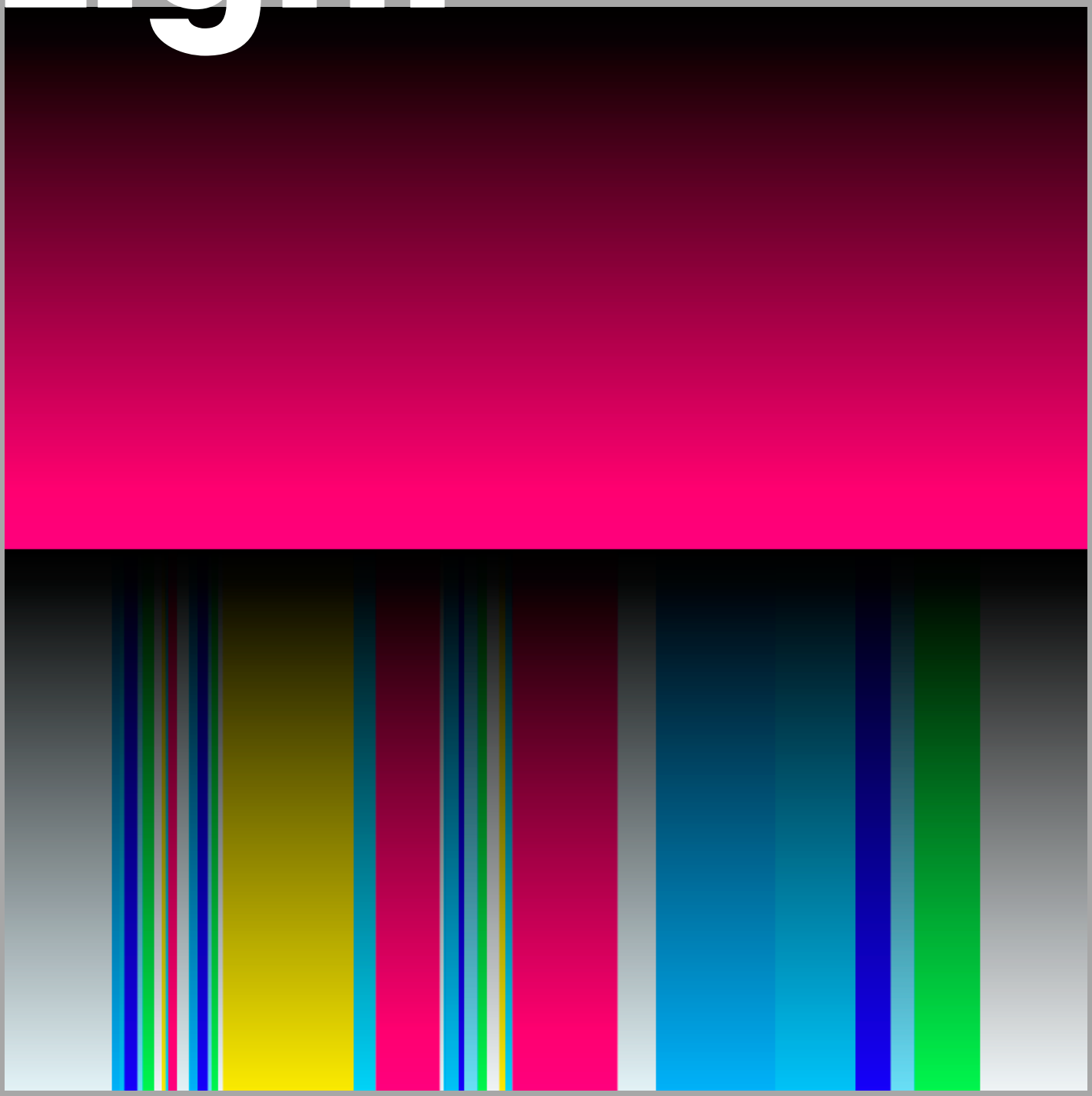
Generative Design **Variations 1**

Light



Henk Lamers

Discovering programmed design

www.generative-gestaltung.de
Special thanks to Julia Laub, Hartmut Bohnacker, Benedikt Gross and Claudius Lazzeroni

Results can be found here:
mycodehistory.wordpress.com
www.loftmatic.com
www.flickr.com

Text and design:
Henk Lamers and Jeanne de Bont

Copyright © 2015 Loftmatic, The Netherlands. All rights reserved

Introduction

During 2010 and 2011 I went through Casey Reas's and Ben Fry's book 'Processing, A Programming Handbook for Visual Designers and Artists'. While doing the exercises of the book I tried to make four different versions. Here is the link to the first page of the 532 exercises I managed to complete. When I was finished I sent the results to Casey Reas and on January 22, 2011 I received an e-mail from him: Thank you for the email. I've seen your website in the past, but I took more time with it today. I think your exercises are impressive. I started in a similar way in 2007. Programming didn't come naturally and I didn't have the resources that are available now. I produced many small programs to work through the basics and over time, it started to come easier. Now, it's fluid after years of practice. You might want to look at the Generative Gestaltung site and code as a way to make a next step. Good luck and thank you for sharing your journey. I replied: Thanks for your reaction. I will follow your advice to take a look at the Generative Gestaltung site. I will start with it when they issue the English book version. Meanwhile I followed a course by Andrew Glassner about 2D Animation and Interaction.

Anyway... when I saw the impressive and very well designed 'Generative Gestaltung' book in 2011 I contacted the makers of the book to ask them if and when an English version would become available. I received an answer of Julia Laub that it was hard to find a publisher. So I bought the German version and started to translate it into English. But in August 2012 I received the English version of 'Generative Gestaltung' now called 'Generative Design'. I couldn't wait to start. I began a new project and called it 'Generative Design Variations'. I estimated that I could do one project per week. My idea was to keep on going until I would run into problems which I couldn't solve myself. I was not sure if I could simply change the code, the variable names and the functions to make different variations of the graphical output. I also wondered if the makers of the book would allow me to publish my variations on my Loftmatic site and write about my experiences on a blog? So again I wrote to the authors of the Generative Design book. Julia Laub replied on my questions within 30 minutes.

Thank you for your email. We´re glad you like our book and we're happy to hear, how enthusiastic you are about working with generative design methods. We published the code under the creative common license. So feel free to use the code and change anything you want. We'd appreciate, if you refer to the book´s site www.generative-gestaltung.com when publishing your experiments on your site. We wish you good luck and have fun! Best, Julia for the Generative Design book team, Hartmut Bohnacker, Benedikt Gross, Julia Laub and Claudius Lazzeroni.

I was all set for the next step into learning to program with Processing. I had the intention to make at least five variations of each Generative Design program. I also knew that it would be hard to create better results because the images and the original programs in the Generative Design book were already on a very high level. It was not my intention that my changes would lead to an improvement of the original code and images.

My goal was to extend my basic knowledge of programming. I created 1219 modified programs, 5957 visualizations and I wrote 83 articles about the design process. In these publications I share my entire experience and results. I hope you will enjoy reading them as much as I have enjoyed it to create them.

Henk Lamers

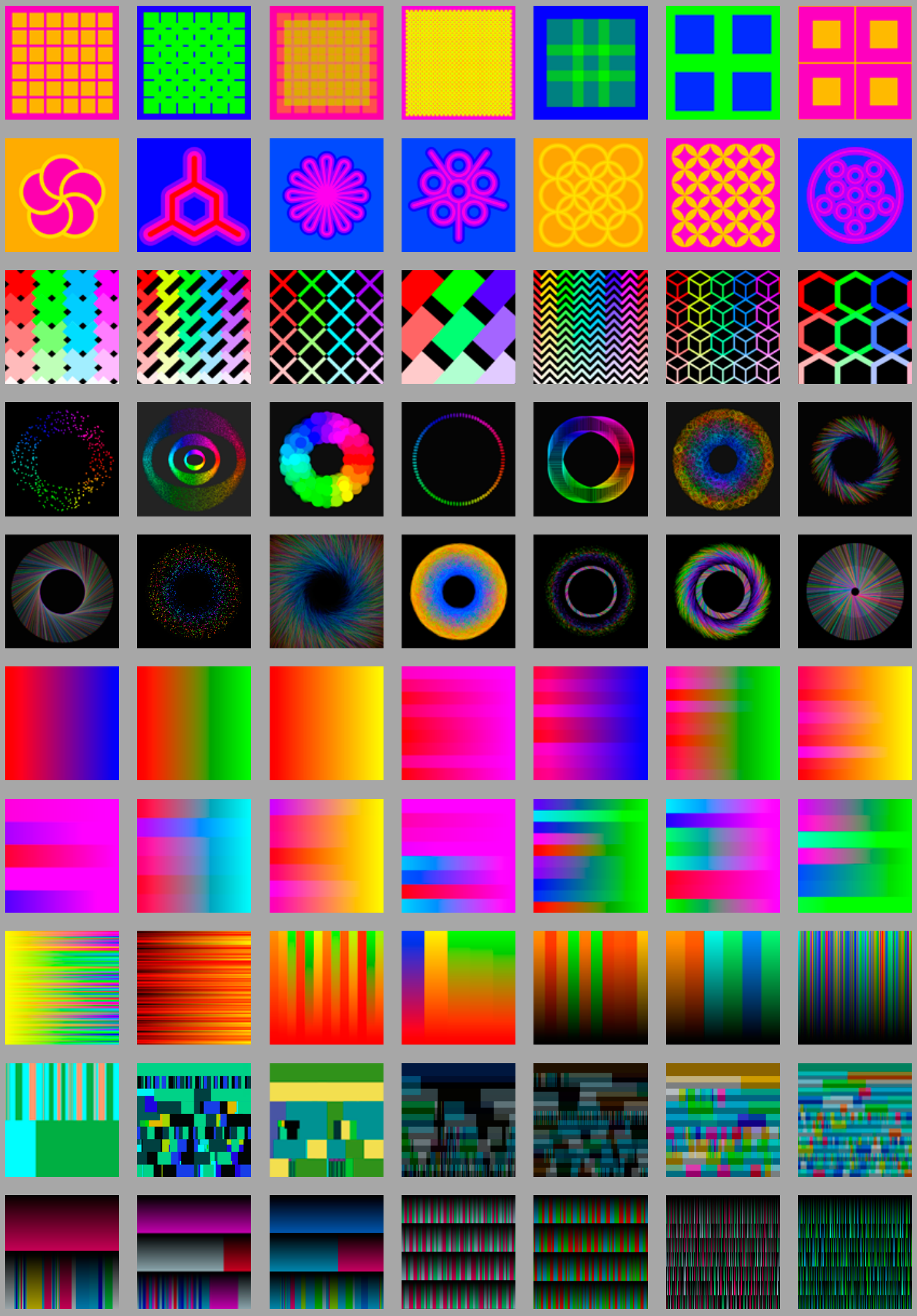
Light

MyCodeHistory: 3 November 2013

It is difficult to say what is more important: light, color or geometry. Light is connected to color as color is connected to geometry. When I started with this Generative Design Variations project it was very unpredictable which colors I was going to use. You might think: Well this is a self-study so than it really doesn't matter what colors you use. But that is not entirely true. Because when I am completely free in choosing colors I will limit my choice. Why? I think this has something to do with the nature of a project. Color can add a functional dimension to the use of form. Likewise form can add a dimension to color. For instance: silver-colored cars are least likely to be involved in a car accident, since they are most visible on the road and in low light. Or the use of color in graphic design. You might want to read 'The Art of Color: The Subjective Experience and Objective Rationale of Color' by Johannes Itten. Or the power of color in films. I suggest to read Patti Bellantoni's 'If it's purple, someone's gonna die.'

'You see, I think every panting should be the same size and the same color so they're all interchangeable and nobody thinks they have a better painting or a worse painting.'

The Philosophy of Andy Warhol,
Andy Warhol, 1975



Hi, Katachi

MyCodeHistory: 11 November 2013

I must admit that when I started Generative Design Variations I was not very convinced of the fact that I could bring this project to a successful end. Of course you never know what the end result will be. Whether it's a design or a self-study project. Doing projects is like making documentaries. When you are making a documentary and you are interviewing a person who you don't know very well you get a totally different experience of that person's character after talking with him or her for an hour. That is also true with doing projects. When a project is finished, you experience the process totally different than when you started it. That experience is even stronger when you do something which is fairly new to you. Like programming for instance.

But when I looked at the 'Hello, color' program P_1_0_01 in the Generative Design book on page 182 it seemed that it was very easy to make variations. I think that the essential thing in this program is that the horizontal movement of your cursor would mean a change in size. And a vertical movement would mean a change in color. I thought it was a good idea to keep those essentials in the program when making variations. The original program uses one square. That's fine. But why not change that into 36 squares? You have to excuse me because I use the word 'why' a lot. And than I try to answer that question and bring it into practice by changing the program. Why not putting even more squares in it? The answer might be: because at a certain moment you get a pattern. And that would be not right. Because we only use large objects in this program. Why not use transparency? Because the colors are getting muddy. And before you know it you have a range of variations. That is one way I work.

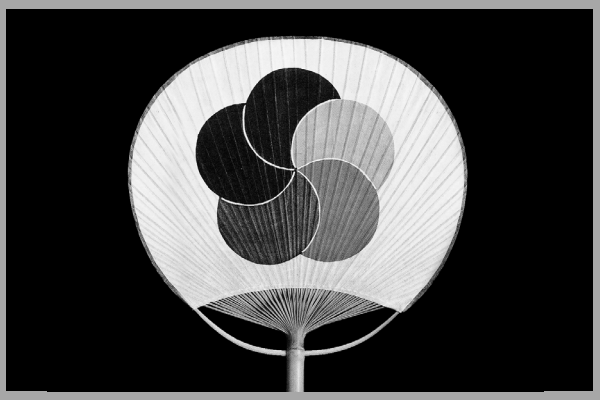
Another way of working. Why do I have to stick with circles and squares? Good question! I decided to use images which I 'designed' in Adobe Illustrator. These images are exported using the svg-format and imported into Processing. As a source of 'inspiration' (I hate to use that word because in most cases it represents the word 'copying') I used a book I received in 1999 from Jeanne de Bont when she came back from a business trip. The title is 'Katachi, Classic Japanese Design', by Takeji Iwamiya and Kazuya Takaoka. In that book I found decorations which were used on an 'uchiwa' or rigid fan. The 'uchiwa' was introduced in Japan from China during the Nara period (710 – 794).
I thought it was very simple to re-create it. Just five circles on top of each other. But that was a mistake because that won't work with circles. And that also says more about the intelligence of the Japanese people who designed the original symbol than about my ability to recreate it. A while later I found a stylized tortoise shell relief shape which you can find on the side walls of the main hall of Hagashi Honganji in Kyoto, Japan. It was established in 1602 by the Shogun Tokgawa Ieyasu. More then 400 years later I repeated the image twice with a different line thickness and fill settings. I also used flower-like designs which you can find on ancient Japanese lacquer trays. Stylized plum blossom shapes which are used as a decorative element on tea or sake cups. I made an interpretation of a symbol which I found on a Japanese Kimono called 'hanten' or 'happi'. Hanten were worn mainly by artisans and members of the fire brigade during the Edo period (1600 – 1868). I made an interpretation of a namako-, or sea cucumber-, style wall decoration. And finally I made an interpretation of the Namako wall or Namako-kabe. This is a Japanese wall design widely used for vernacular houses, particularly on fireproof storehouses of the latter half of the Edo period.

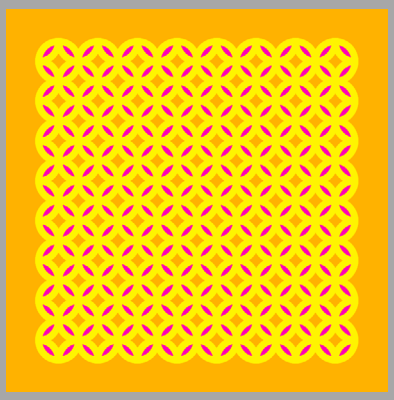
When I was finished with making the variations, using this program, I got mixed feelings about the work I did. Because in the end the only thing you do is replace and change the svg-file in Processing. The basic program (as it was in the original Generative Design book file) was still the same. I only made some adjustments which anybody could have made. Even without knowing anything about programming. I don't know if that is a good or a bad thing. Anyway I have now learned what I can do with x- and y-values when you connect them to size and color. And I guess that's the essence.

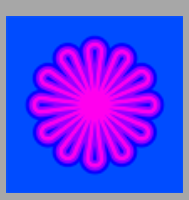
Things which look simple are often very complicated when you study them. Here is a quote from Steve Jobs about his father: 'It was important, my father said, to put also the back of cabinets and fences properly together, even if they are not visible. He liked to do things well. He even cared about the parts that you could not see.'

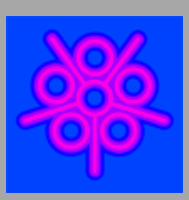
Notes From: Isaacson.
'Steve Jobs, The Biography.'

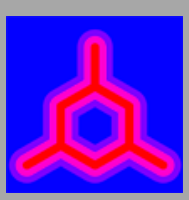

Katachi
Classic Japanese Design

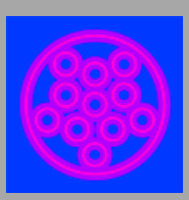


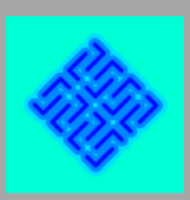


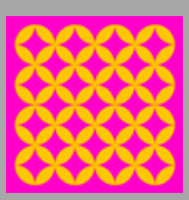


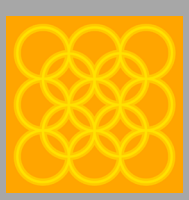












Weird things in a grid

MyCodeHistory: 26 November 2013

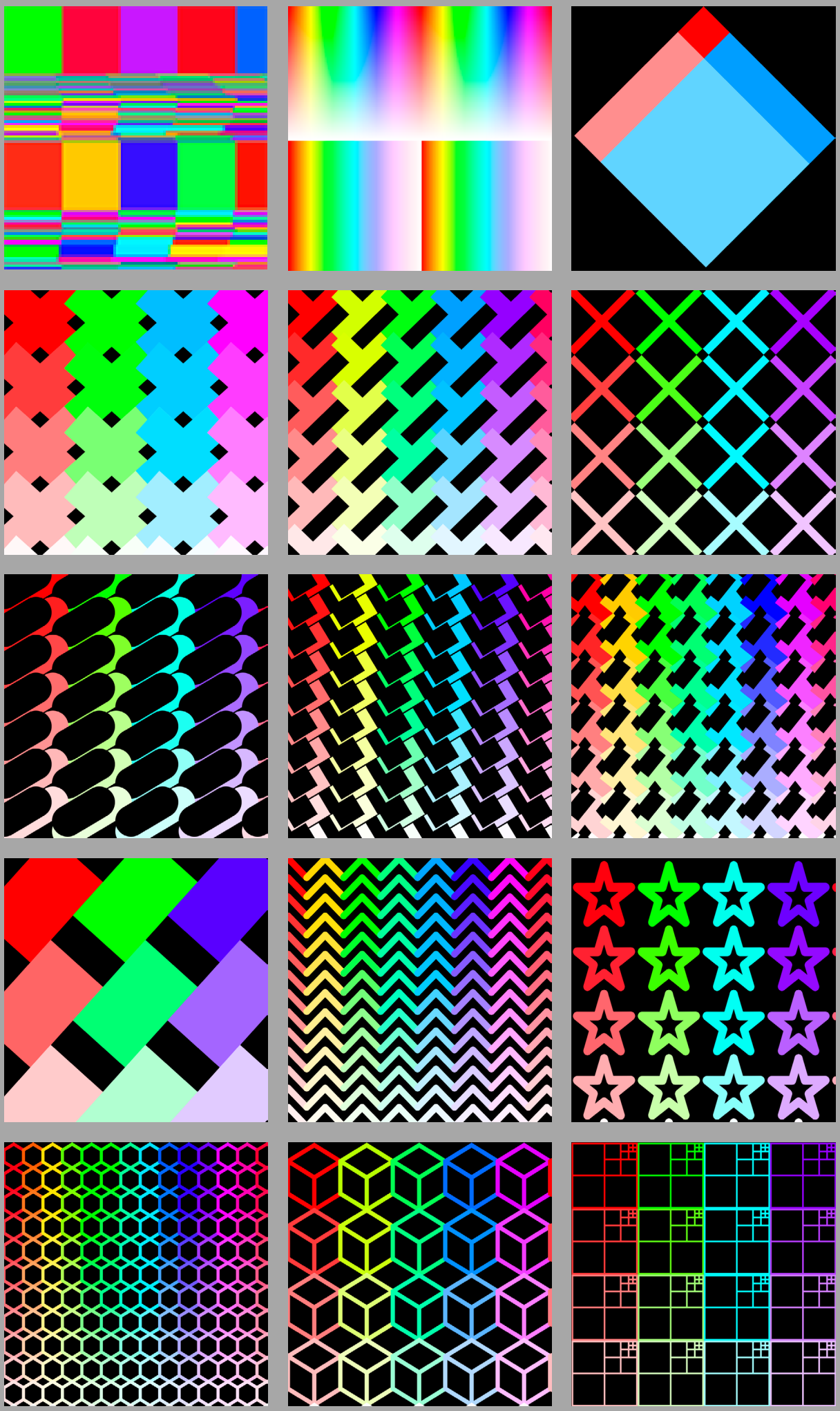
I can mention all the adjustments I did in the 35 program variations but that is not so interesting. But if you really would like to know you can find all of them on the mycodehistory blog. What is interesting though is the fact that at a certain point, during programming, serious problems can occur. For instance: to be able to publish interactive programs on the web you can convert Processing's code automatically to JavaScript. And that is a good thing. The bad thing is that weird things can happen. For instance I was working on the P_1_1_1_01_ GDV_28 sketch. I let the program run. But nothing happened. At least, that is what I thought. There were some minor changes see-able in the display window and the program did not give any errors. It just kept on running. I had some other things to do so I let the program run for a while. When I came back, an hour later, something was displayed on the display window. But it took more than an hour to display the image on my MacPro. During that hour you see a colored spectrum on the screen accompanied by a cursor which sometimes turns into a rotating colored beach ball. So I tried it on a MacBookPro and there it took 5 seconds to display. At that time another MacBook Pro was still very busy with it. And the iPad Mini is displaying the file after two minutes. What is going on? It is time to post an alert on the Processing forum. Explained the problem and asked if anyone could tell me why this file needs so much time to display? I got an answer from GoToLoop. I have no idea who GoToLoop is. I have never met him or her. And he or she has never met me as far as I know.

GotoLoop: 'Hey! Made some tweaks! I believe it's starting faster now!' I added the tweaks and the program worked fine. I asked GoToLoop if he had any idea what went wrong. GoToLoop: 'Problem was that sketch started w/ both mouseX & mouseY = 0. And since stepX & stepY are derived from them, 1st double loops had an immense # of iterations. And consequently lotsa shape() draws which demanded a very powerful computer to finish! Once passed that, and user hovers the mouse upon the canvas, we get bigger stepX & stepY. And program finally soars w/ a smaller # of iterations!' I said thanks to GoToLoop for helping me out.

Sometimes I have totally no idea what is happening within a program. When something is wrong it is a good idea to post the problem on the Processing forum. It is a good thing that there are always people who would like to help you out. Another thing is that I did not understand how GoToLoop solved the problem at that time. I'm writing this on Wednesday December 9, 2015 but now I do understand what he did and what the problem was.

'We should feel free to research, develop, innovate, even in areas which are considered out of date by those stuck in passé futuristic ways of thinking.'

Notes From: David Edgerton. 'The Shock of the Old.'



Variations in a circle

MyCodeHistory: 3 December 2013

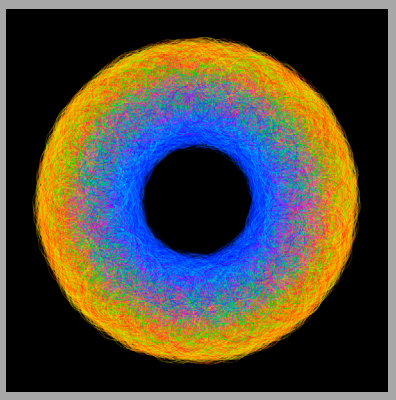
When I started with this program I thought… this is a very difficult one. I don't think I can make more than ten variations. But that was a mistake when I got rid of the fills and only used lines and points it was much easier to make variations with this program. Sometimes you have to break things down to the basics. Which on its turn might lead to new goals. Anyway, the shapes became more interesting and I lost control of the intention of the original program. But I don't know if that's wrong or right. I also learned a new thing! I had no idea that you could give a line the thickness of 1.5. The strokeWeight function is Processing's manner to say: 'Hey this line should be of that thickness'. It makes a difference though if you change this to strokeWeight 1 or 2. 1 is one pixel. 2 Is two pixels. But how thick is 1.5 pixel? I think it has something to do with the anti-aliasing of the line. When you switch off anti-aliasing, a 1.5 pixels is a 1 or 2 pixels line.

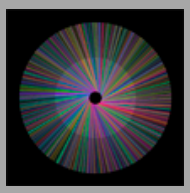
Another strange thing I noticed, there is a superfluous semicolon in the code. I do not think that it does any harm. Hmm… I now added eleven more semicolons in my sketch. Nothing seems to happen. Ok! Might be good to know at some point.

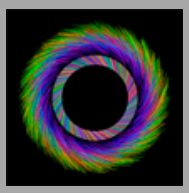
What's also nice is that sometimes you look at a program and you think… hmm… that might be tricky. You have the feeling that there are not so much variations possible. But when you delve deeper in the program everything is tending to change. Suddenly you see possibilities which you did not see before, when you started the process. And in the end I managed to make thirty reasonably different variations. By the way: P_1_1_2_01_GDV_20 Reminds me of Méret Oppenheimer's 'Object: Breakfast in Fur', 1936, Museum of Modern Art, New York.

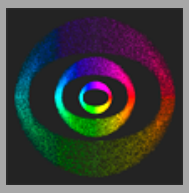
During World War II, when John von Neumann constructed his famous electronic brain for the government. he identified it on delivery as a Mathematical Analyzer, Numerical Integrator, and Computer. Scientists worked with it for several days before they realized that the first letters of the name its inventor had given it spelled MANIAC.

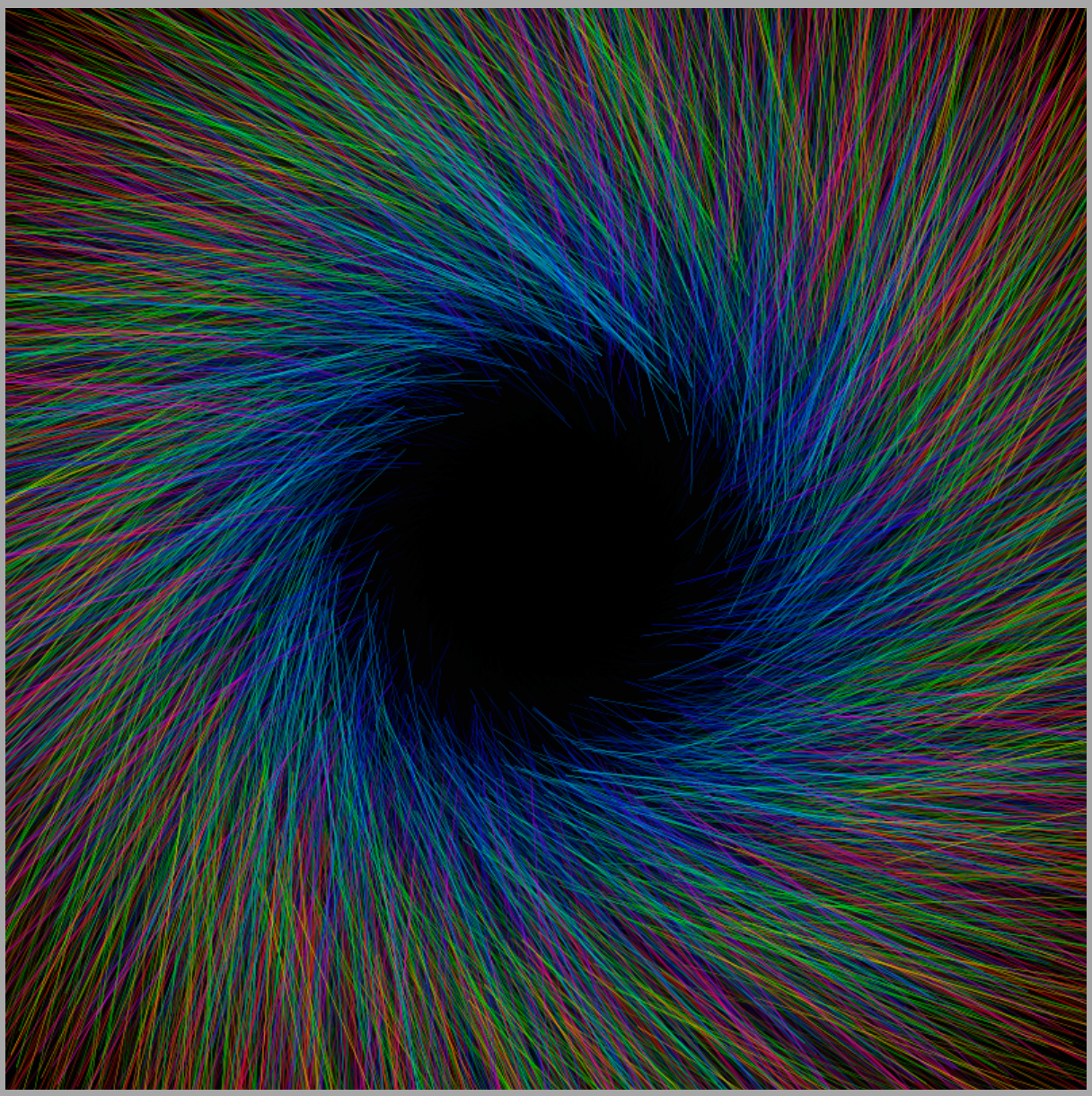
John von Neumann, 1903–1957, Mathematician











GDV 1.2.1 Bars and numbers

MyCodeHistory: 10 December 2013

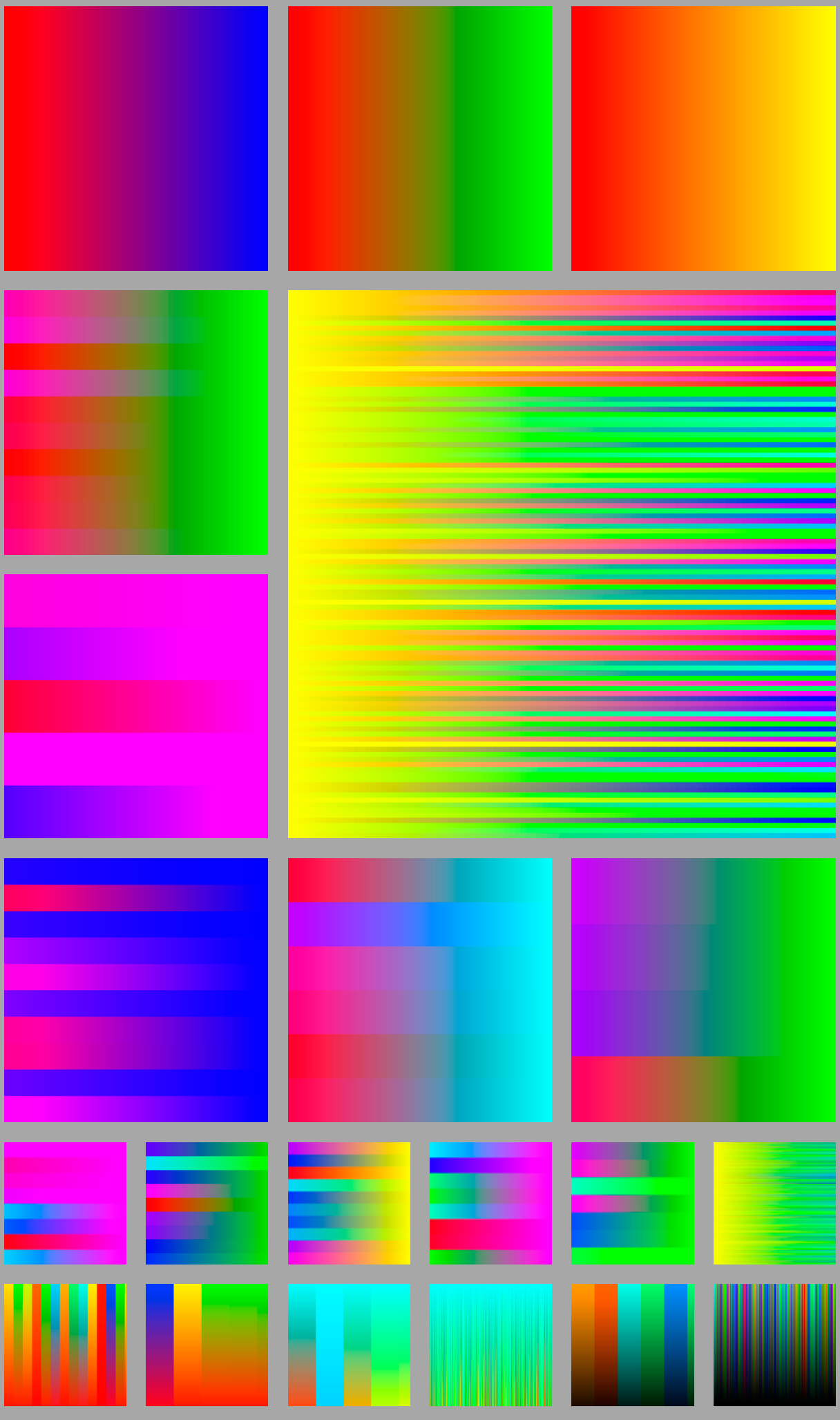
I treated this program a bit different than the earlier ones. After making some changes in a few sketches (Processing programs are called sketches) I was trying to change the rectangles into ellipses and triangles. But I thought it would be a better idea to find out what the essence of the program was. When looking through the code I learned that a HSB color (HSB stands for: Hue, Saturation and Brightness) is mainly determined by 360 numbers of hue, 100 numbers of saturation and 100 numbers of brightness. You could eventually add another 100 to that range and use it for transparency levels. But when you subtract 60 from 360 you get 300. 300 Minus 60 = 240. 240 Minus 60 is 180. And continuing subtracting leads to 120, 60 and 0. The numbers 300, 240, 180, 120 and 60 will show you a cross-section through the HSB color range. So I decided to use those numbers for changing the code a bit.

Most of the images use gradients which are the outcome of interpolating one color to another color. On a certain moment I found it interesting to break the smoothness by a small area which interpolates within another range of numbers or an area which does not interpolate at all. I tried using only thick horizontal bars. In a later stage I made the bars smaller and the same size. The colors are picked random in a certain range. The interesting thing is that the program sometimes chooses two colors which are the same. That makes the bar double its thickness. Which in return makes the image much more interesting.

Sometimes people really don't understand what you are doing and why you are doing it. Lately someone said to me: 'Hey… why didn't you do that in Photoshop.' Of course that would also be a possibility. Photoshop was developed in 1987 by Thomas and John Knoll. They sold the distribution license to Adobe Systems Incorporated in 1988. But I have worked with Photoshop since 1990. At that time Photoshop 1.0 was released. It had only one layer. Digital retouching on high end systems, such as SciTex, cost around 250 Euro's an hour. Just for basic photo retouching. I don't know what the extra value would be if I used Adobe Photoshop. Working with programs gives much more possibilities. It can also lead to mistakes. And these mistakes can bring you to other solutions which you might never meet when using Adobe Photoshop. Another point is that I am trying to learn to program. I don't feel the need to study Adobe Photoshop again. There are excellent books about Photoshop by Katrin Eismann I studied seven years ago.

A rich Dutch grocer was introduced to Picasso. He examined the works in the studio and then said, 'Master, I understand every one of your productions except one.' 'And that is?' asked Picasso. 'Your dove. It seems to me so simple, so primitive that I cannot understand it' said the grocer. 'Sir,' Picasso said, 'do you understand Chinese?' 'No!' The grocer replied. 'Six hundred million people do' said Picasso and he politely showed him out.

Pablo Picasso, 1881–1973,
Spanish artist, sculptor and ceramist.



Libraries and JavaScript

MyCodeHistory: 26 December 2013

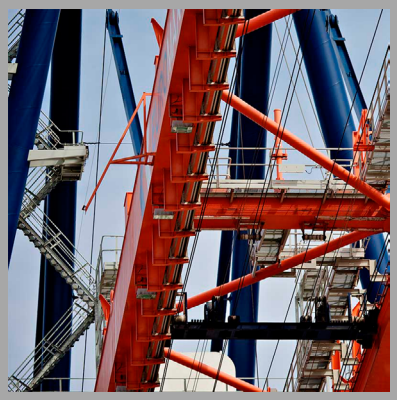
Some programs take more time to get into then you had planned. It took me a few weeks to go through the P.1.2.2 Color palettes program. Basically the program scans the pixels of a photo. Those values are then stored by hue, saturation, brightness, or greyscale. I began with selecting three of my own photographs. One from the harbour in Rotterdam. A road sign from Iceland and a dam in the town of Driel. I could have selected more than 3 images but I thought it would be more interesting to select just these three because than you can compare the different variations better.

The basic things I did for all variations of this program is to put an extra layer on top of the photo. The program variations display simple shapes like rectangles or ellipses. But during the creation of the variations I found a few new things. This, and the previous program, use the GenerativeDesign library, A library made by Benedikt Gross and Hartmut Bohnacker for the Processing programming environment. The package contains classes for 3d-surfaces, simple physics (nodes, springs, attractors), graphic tablets, ASE-export and asynchronous loading of xml, html and images. This programs works mainly with the ASE-format and the sortColors function. It's getting a bit technical now but the ASE-format let's you save a color palette which you can re-use in Adobe applications. The sortColors function returns a sorted color array. To be honest I never worked with libraries before so this was a good opportunity to get to learn about them. I went to the GenerativeDesign library site and noticed that there were a few more functions which I could use because they were not in the program. I checked the reference page for this. Besides of hue, saturation, brightness and gray scale there is also a possibility to sort on red, green and blue. Another option is alpha which allows you to sort on transparency.

Not all Processing Libraries can be imported in JavaScript. A library is a kind of helper and provides extra functionality. You import it in your program and don't worry about it until you need classes or functions which are supplied by the library. At a certain moment I converted the Processing files to JavaScript to show them on my Loftmatic website. When displaying those in the browser I noticed that the pixel-sorting was not working. Most Processing libraries are not working in JavaScript mode. But it is good that I know that now.

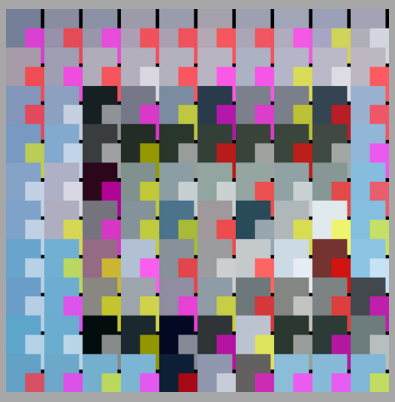
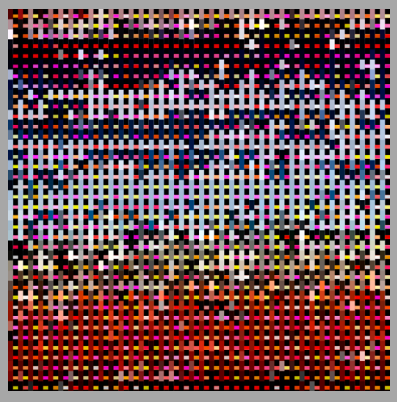
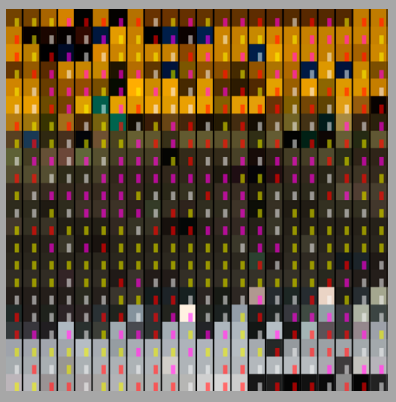
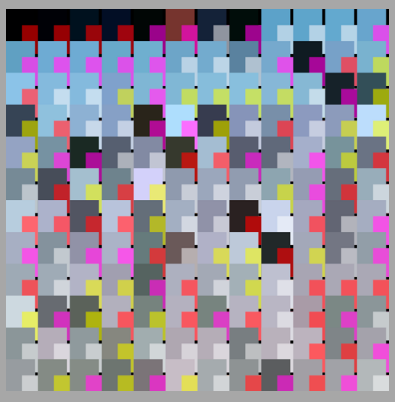
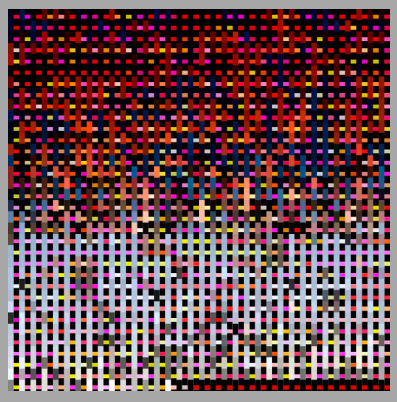
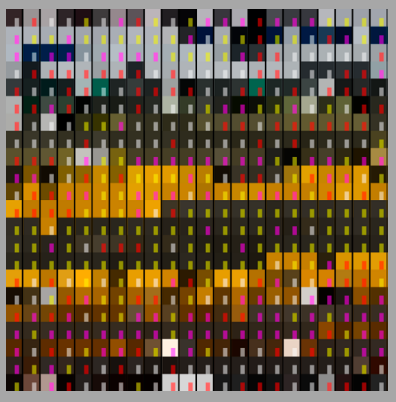
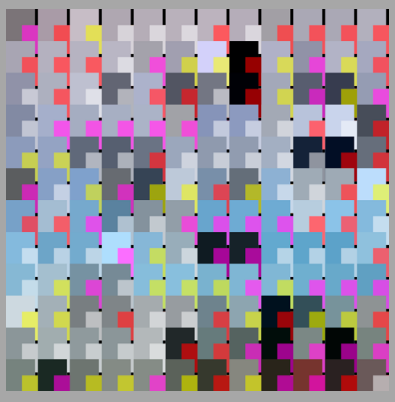
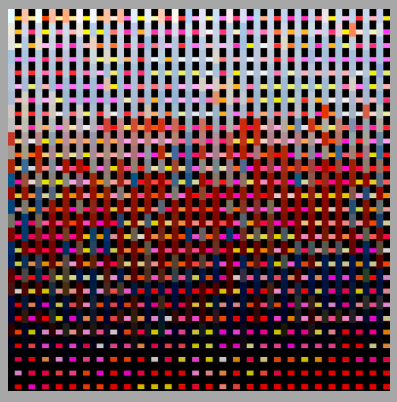
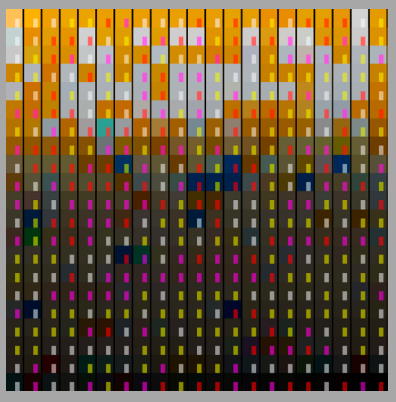
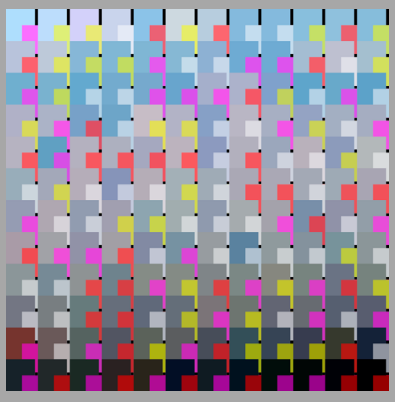
During an interview in 2005 with the composer Dick Raaijmakers he mentioned the following: 'This is a kind of hobby-horse of mine but technology delivers something extra to you that you actually can't do yourself.'

DVD Room 306, Dutch Electronic Music from the Philips Research Laboratory, 1956–1960,
©2006, Loftmatic, Henk Lamers and Jeanne de Bont



Fljótshlíð 42 F261
F 210



F210











Color palettes rule

MyCodeHistory: 12 January 2014

In the Generative Design book this program was divided into four variations. The program used random values which are selected between two points in the hue, saturation and brightness ranges. I made five variations of all four programs. The first variation was changed a little. I only placed circles positioned at the cross-points of the rectangles. I think I am slightly ahead of the book because this is a shape modification instead of a color issue. I also switched off the fill function and replaced it with a stroke. And I used the new rounded rectangles function in Processing 2. I think I did not use rounded rectangles since 1984 when we had to draw them by hand. The program behaved rather clunky. because I slowed it down by decreasing the frame rate. But it is nice that you get shapes with fine matched color palettes.

In the second group of variations I used some horizontal randomization to generate more vertical lines. I removed almost all color information except for two colours. This gave me interesting vertical shapes but in certain cases some artefacts showed up. The program displayed horizontal lines of one pixel thick. I didn't mind, the effect looked nice. I reduced the amount of horizontal rows to 2. By clicking the mouse you generate interesting patterns, but you have to click a few times before they show up. Increased the randomized amount of rows between 1 and 10 and increased the horizontal randomization effect which results in even more vertical lines. More randomization in each horizontal bar. With every mouse click it also gave me uncommon, but surprising, color combinations. Than I reduced the amount of colours to three. Sometimes I completely abuse it to see what will happen and in many cases the effect is really great.

With the third group of variations I introduced several amounts of transparency and overlapping. And here I got into big trouble. When I converted the program to JavaScript the sketch didn't show up. What had happened? I posted a question on the Processing forum, thought the problem was raised by using OpenGL in this sketch. I got an answer back from calsign: 'It appears that you use the java.util.Calendar class, which is not available in JavaScript mode because it is (obviously) part of a Java package. Instead (depending on your usage), you must use either Processing's date / time functions or JavaScript's native date / time functions. This is one example of deviation in JavaScript mode from Java mode. Also, I'm not entirely sure what the generativedesign.* package is, but it may have been created for Java mode and, as such, won't work in JavaScript mode. You may have to port it over.' So the Processing files work but the JavaScript version doesn't.

Within the last group of variations everything seems to be fine again. The display problem was solved. OpenGL also worked and apart from the jagged edges it was fine. If I run the program in Processing there are less jaggies. To make these images I mashed up the program by randomizing the vertex points of the vertices between 0 and 100. I tried to find out if JAVA2D option is working in JavaScript. It seemed to work. The drawback is that you lose all the shadow and transparency effects which is not what I was looking for. But you win in display quality, all jaggies were disappeared. I flipped the vertex-points of the vertices. But also this variation looks much nicer if you run the program in Processing.

I have no experience with writing libraries for Processing. But I assume that writing a library is a lot of work. And I think that re-writing the same library for JavaScript is the double amount of work. Maybe that is the reason why most Processing libraries do not work in JavaScript.

Everything you need in this life is ignorance and confidence, success guaranteed.

Mark Twain, 1835– 1910, American author, lecturer and humorist.