Generative Design **Variations 2**

Geometry

Henk Lamers

Discovering programmed design

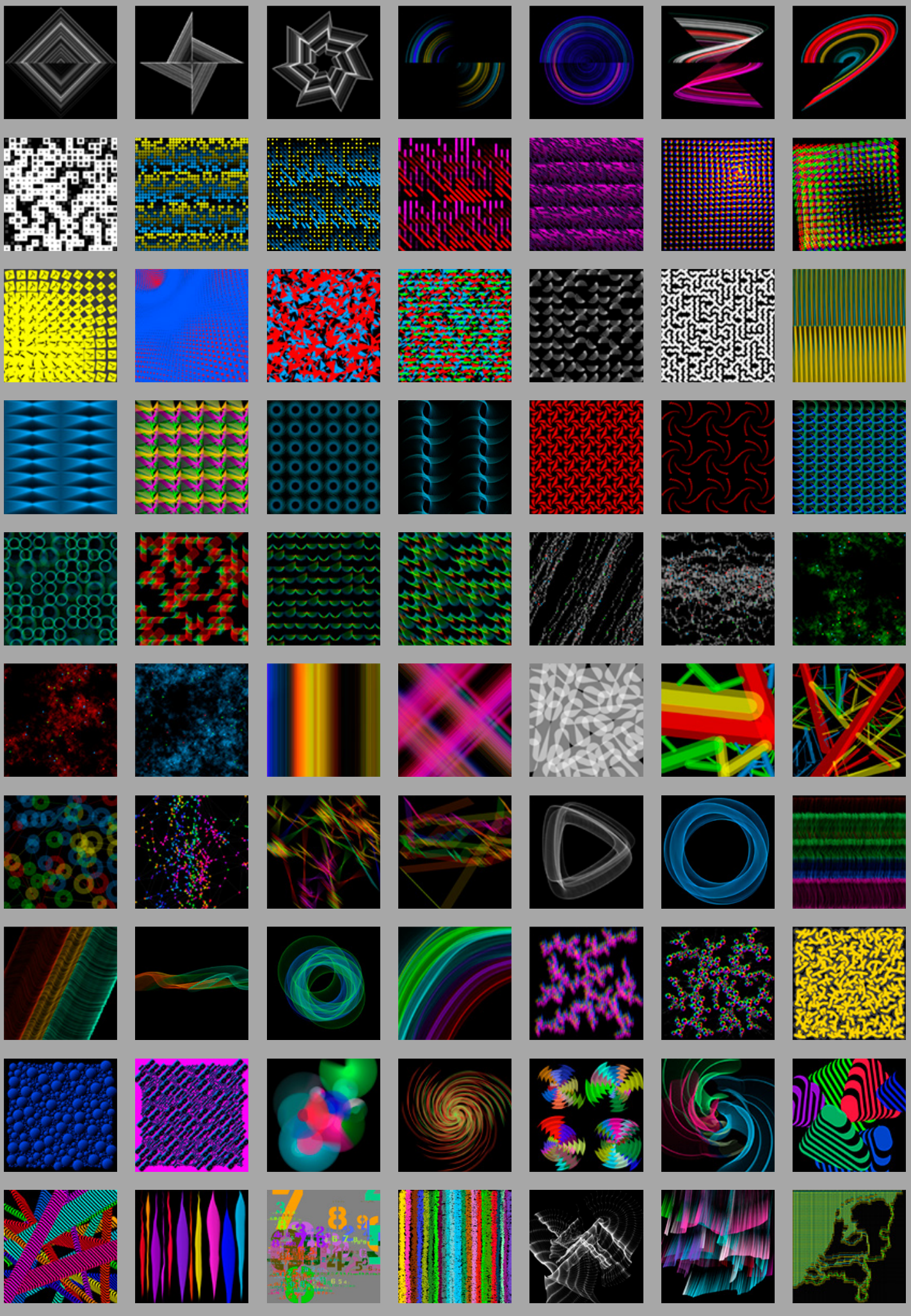
Geometry

Anyone know what a triquetra is? A nonagon? A tangential quadrilateral? A squircle maybe? Well I had to check Wikipedia because I never heard of them. But even when you don't know what I am talking about that doesn't mean that you did not see one of these geometrical shapes in your life. Every day and night we are surrounded by various basic 2D shapes: rectangles, triangles, corners, circles, lines, arcs and planes. These are just the well-known geometric shapes. And our world is loaded with them. I think one of the most interesting shapes is the point. A point is zero-dimensional. It has no height, no width and no depth. But you create a point when you draw one. A point is in fact an imaginary position somewhere in the world around you.

It is hard to imagine a world without geometry. It could consist only of colors. Or shades of color. But not too harsh shades of color because that results in geometry again. Colors that do not have any shape are also difficult to imagine. Thinking of color, I have to imagine at least a flat area. But that area itself is also a form. Even Yves Klein's, 'Proposition monochromes' at the Gallery Colette Allendy in February 1956 have some geometry. It displayed orange, yellow red, pink and blue monochromes. Yves Klein realized that from the reactions he got from the audience, viewers thought that he was exhibiting a new kind abstract interior decoration. The public linked all paintings together as a mosaic. From that time on Yves Klein would concentrate only on one primary color: blue. But when geometry is as minimal as possible, it is even more noticeable.

Operating a screw is deceptively simple. Just mate the grooves atop the screw's head to the appropriate tip of a screwdriver. What happens next is not as simple. My children remember this rule through a mnemonic taught by my spouse, 'righty tighty, lefty loosy.' Personally I use the analogy of a clock, and map the clockwise motion of the hands to the positive penetration curve of the screw. Both methods are subject to a second layer of knowledge: knowing right versus left. Thus operating a screw is not as simple as it appears. And it's such an apparently simple object.

The Laws of Simplicity, John Maeda, Designer and technologist, 1966—, MIT Press, 2006



Hello jaggies

MyCodeHistory: 19 January 2014

This program was a kind of déjà vu because I already started making variations somewhere in January 2013. In the beginning I thought it would not be so much work to redo everything I did before but that was a wrong assumption. This program is behaving almost the same as the first program of the Light chapter. A shape is reduced to one pixel when the cursor is in the middle of the display window. The horizontal value (x) sets the length of the lines and the vertical (y) value sets the number of lines.

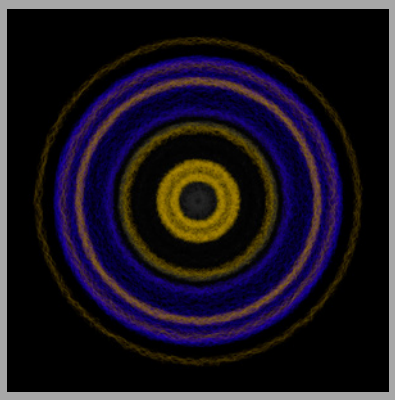
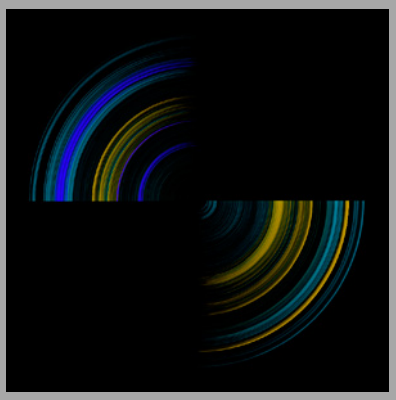
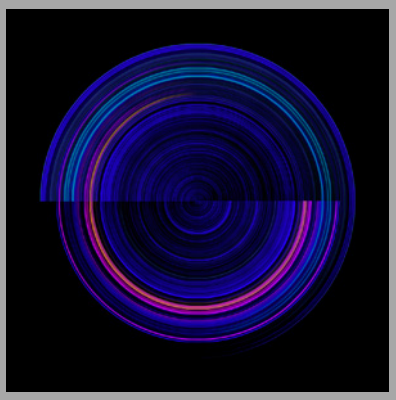
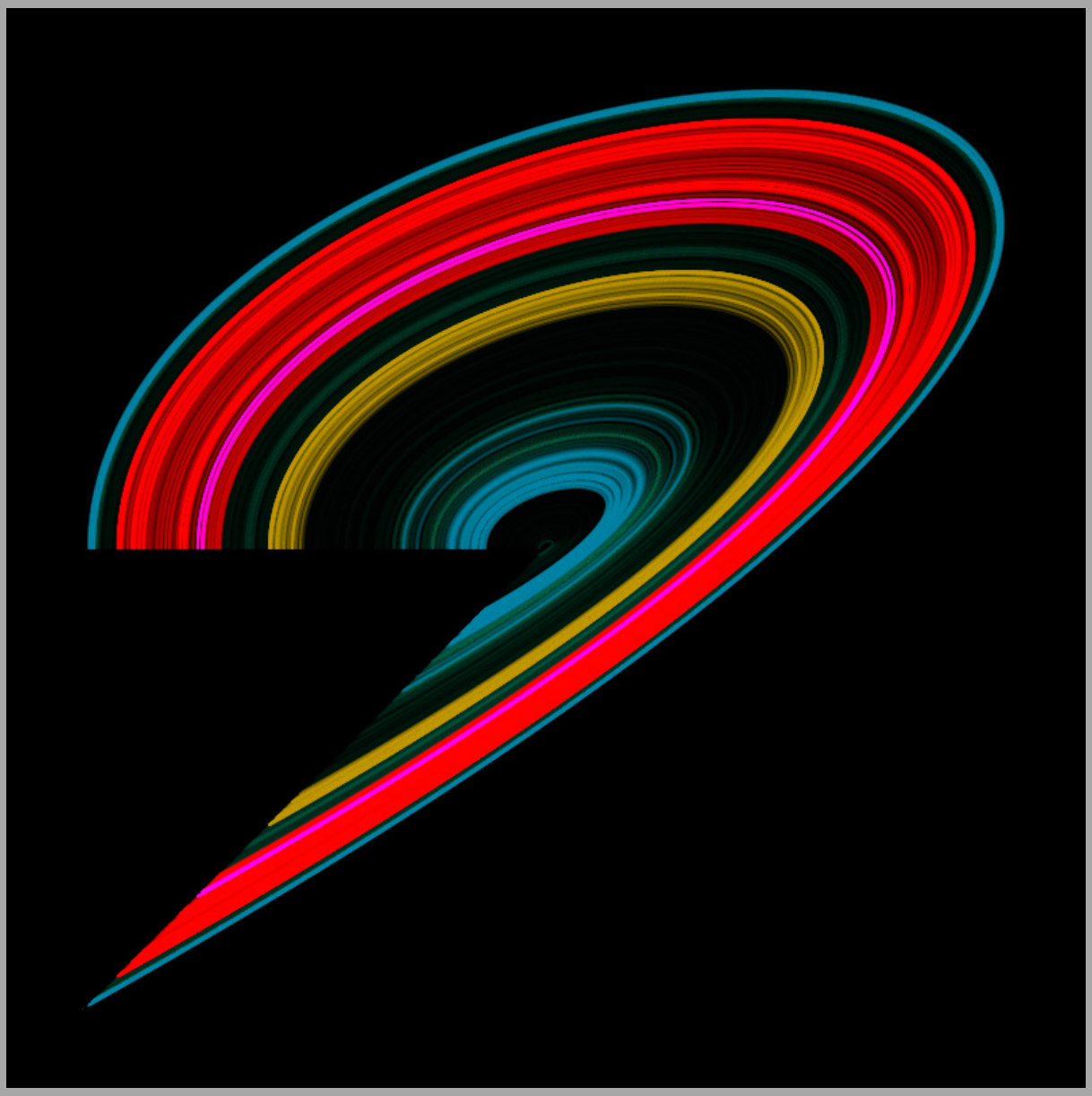
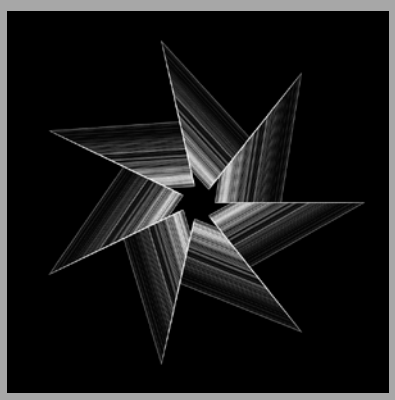
It was surprised that I could draw circles with rectangles. It is obvious that you can draw a rectangle with rectangles. But drawing a circle with rectangles was new to me. It seems also possible to draw a rectangle with ellipses. I made some variations that ended up as a star-like image. And than things were getting out of hand in Processing. The image quality was getting terribly bad. A lot of jaggies appeared. I really did not know what was going on. Switched smoothing on but that did not give a better result. The smooth() function draws all geometry with anti-aliased edges. But it seemed that smoothing is switched on automatically in Processing 2. So I checked the P2D, P3D and JAVA2D rendering modes. Using those did not improve the rendering quality. Still I had not found what the reason was for this bad rendering quality. So I thought maybe there is a way to work around it. Therefore I introduced some randomness in the vertices.

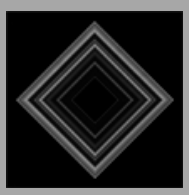
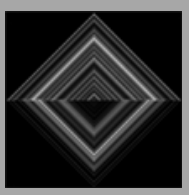
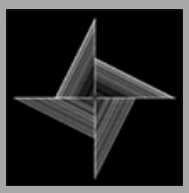
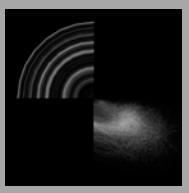
At that moment I had arrived at the same point where I was in January 2013. I compared the rendering quality with the images which I had made at that time. There was a big difference. The rendering in January 2013 was much smoother. What did I do wrong? Went to processing.org and I found a paragraph called 'Drawing Quality'. It states: 'Lines and strokes sometimes uglier – When using the default renderer, a stroke () on a shape may be uglier than in the past, due to recent changes that Oracle has made to Java. Usually this is a problem with single pixel strokes and rough edges, or thick strokes with many steps. To fix the problem, add hint (ENABLE_STROKE_PURE) to the setup() block of your sketch. We don't enable it by default because it makes all sketches considerably slower.' I found that really weird. But maybe I am the only Processing user who cares about rendering quality? Well, I don't mind if it makes my sketches slower because we are not dealing with animation. So I added the hint (ENABLE_STROKE_PURE) to the setup() block of my sketch. And to my surprise the problem was gone.

When using a programming language there is a lot more to pioneer than when you use commercial software such as Adobe Photoshop. A lot of things can go wrong but a lot of things can also be solved. For instance when I converted the Processing sketch to JavaScript I had to comment out the hint (ENABLE_STROKE_PURE). Otherwise nothing would be rendered. Beside of that Processing.js gives an: Unable to execute pjs sketch: ReferenceError: Can't find variable: ENABLE_STROKE_PURE. And one more last thing. If you comment that line out. The rendering quality of Processing.js with ENABLE_STROKE_PURE disabled is of the same rendering quality as Processing 2 with the ENABLE_STROKE_PURE enabled. Just trying to be clear on that issue.

Handed a salad at the table, Turner remarked to his neighbour, 'Nice cool green, that lettuce, isn't it? And the beetroot pretty red–not quite strong enough; and the mixture, delicate tint of yellow that. Add some mustard, and then you have one of my pictures.'

Joseph Mallord William Turner, 1775–1851, British landscape painter

Alignment in Tokyo

MyCodeHistory: 9 February 2014

This program makes use of a grid. And in every grid unit the program chooses between one of two possibilities to draw a line from the upper left corner to the lower right corner, or a to draw a line from the lower left corner to the upper right corner. The choice is made at random. One of the problems I faced is that it is difficult to change the code without getting stuck with a program that is not functioning at all. So to avoid that problem it is very easy to say: 'Hey, lets just change the .svg files and were done'. But I don't want to go that way at this moment. This project is divided into four different programs. I will try to make four times five variations on them.

The first thing I changed is turning the background to black and the lines to white. I always do that because I think white images (displayed on a screen) are better in a black environment. I also changed the line function to point. And here something odd happens. In the Processing version you can use the 1 key for a rounded strokeCap. That is the ending of a line. The 2 key is reserved for a project strokeCap. Project is representing an extended line ending. This seems not to work in JavaScript. I placed a second set of points in the background. Here also the 2 key makes all points project in Processing. In JavaScript it does not work. And another mysterious thing. The 1 key (strokeCap ROUND), 2 key (strokeCap SQUARE) and the 3 key (strokeCap PROJECT) do work in JavaScript. But maybe that works because I switched everything from points back to lines again. By the way (strokeCap SQUARE) does not work in Processing on points either. StrokeCap (ROUND) and strokeCap (PROJECT) do.

In the second version of the program I tried to give the sketch a more randomised feeling. Also the contrast of the elements is different. Changed the colors a bit but my browser (Safari 5.1.10) does not like this sketch. I used arcs. It gives me a spinning beach ball when I hit one of the selected number keys. In fact it doesn't do anything. Or does it? Yes, it does something. It has a response time of a minute. In Processing it works just fine. Converting the sketch to JavaScript I have to restart my browser.

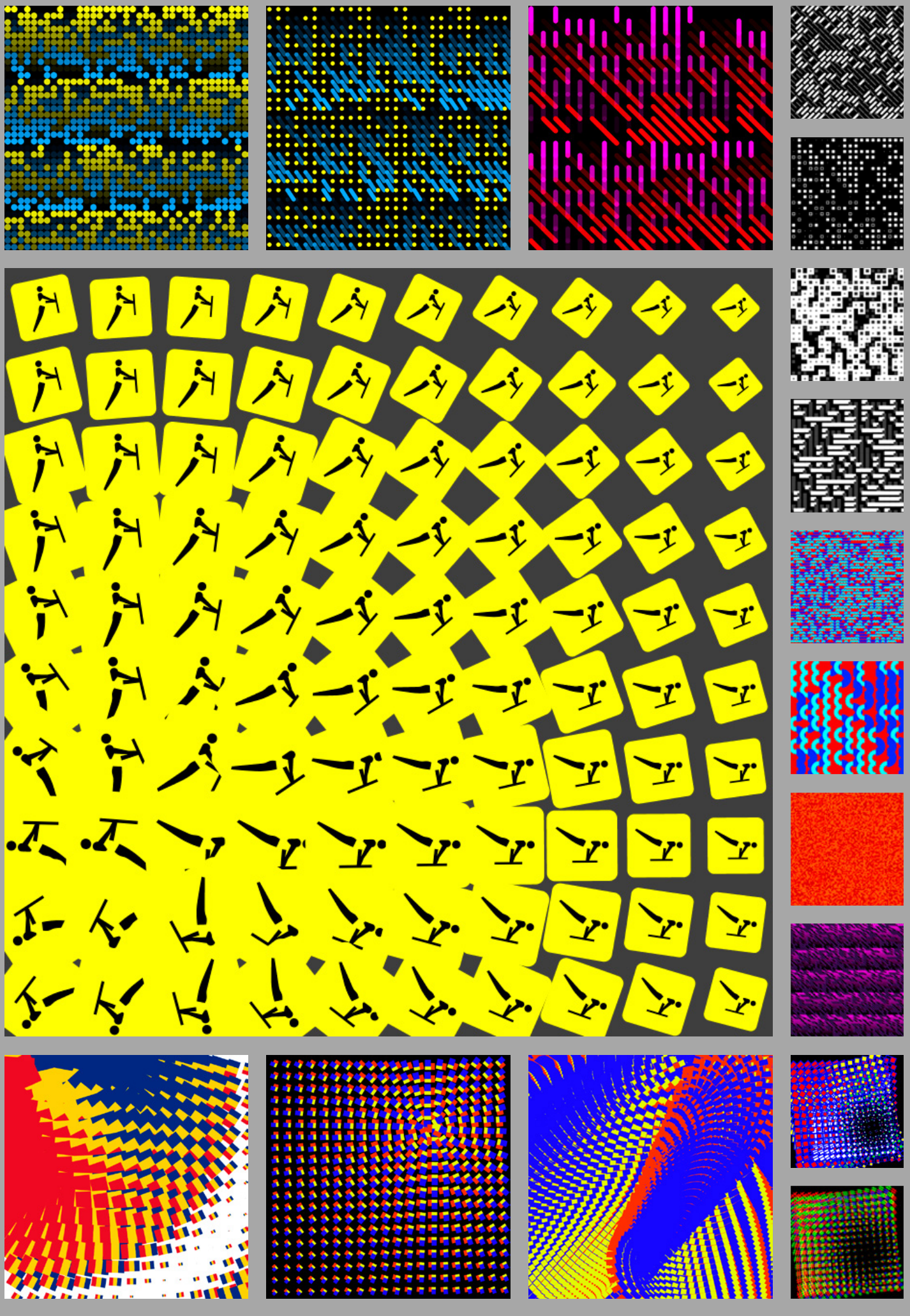
The third version of the program gave me visually totally different results in JavaScript compared to the output Processing gave me. Replaced the lines by points. Still a lot of trouble with JavaScript. I did not experience these problems when Processing worked with applets for the Web. But applets are depreciated from Processing 2 on.

In the fourth program I changed almost nothing. But this time I did changed the svg's. In this case I use the flags of some European countries. It all worked fine. At that moment the XXII Winter Olympics had begun from February 7 to February 23, 2014 in Sochi, Krasnodar Krai, Russia. I had a look at the icons that were made for that event. But for me there is no doubt that the best icon designs were created in 1972 for the Munich Olympics by Otl Aicher and his team. In 1964 the Olympics were in Tokyo. The icons of those Olympics were designed by Masaru Katzumie and Yoshiro Yamashita and they had an interesting style too. So I have redrawn five of them in Adobe Illustrator. Exported them as a .svg file and imported them into the program.

It is not always wise to choose the most practical and problem-avoiding approach. Sometimes you have to face the problems and try to solve them as good as you can. If that doesn't work you have always a fall back scenario available. Importing .svg files for instance.

'The assumption that the new is much superior to older methods is widespread.'

Note from: David Edgerton, 'The Shock of the Old.'



Do less, get more

MyCodeHistory: 14 February 2014

A pattern of objects is drawn to the display window. Random values are added to the position of the objects, forcing them to move horizontally and vertically. The further the mouse is moved to the right, the greater the movement. This is generally the essence of this program. Although this seems to be strait forward the images can become fairly complex. There were four different versions of the program so I made twenty variations.

In the first program I gave the circles some more thickness and repeated the program lines for that circle once more. Placed a second circle within the first circle and than it was a matter of replacing circles by rectangles. Replaced the rectangles with arcs and changed the weight of the lines.

In the second program I placed an extra layer on top of the existing one. Changed the ellipses into rectangles. Used the 1, 2, and 3 keys to change the colors or alpha-channels. I also used the variable tileCount() to influence the radii of the rectangles corners. Because the objects in the background are moving faster than the objects in the foreground you get the impression that there is some sort off 3D-ish thing going on. I decreased the size of every rectangle to get a more open structure.

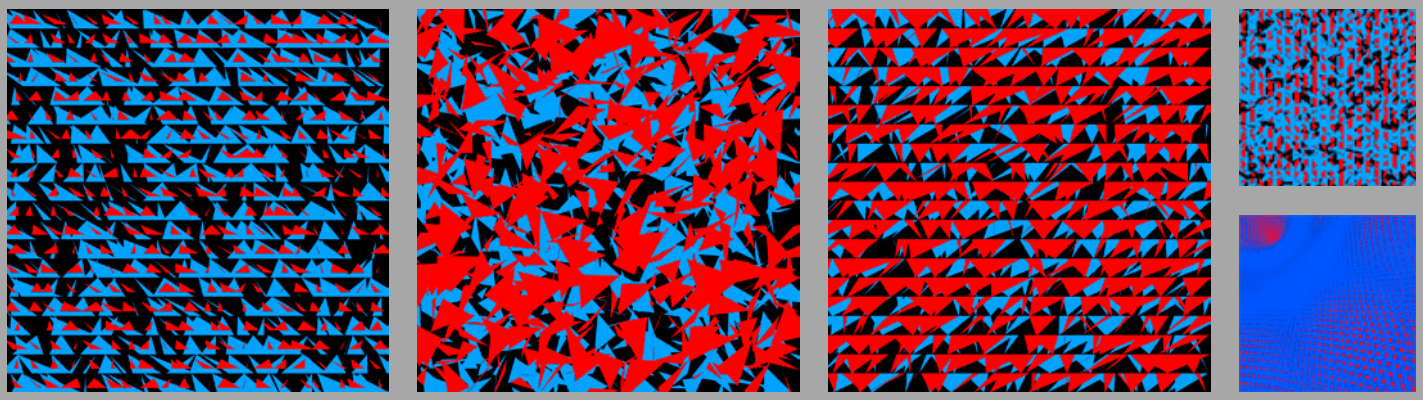
Working with the third variation of the program JavaScript completely ignored the richness of symbol-like elements. I think that JavaScript is using a kind of Photoshop flattening? Ah… I found that there is an alpha bug in Processing 2.0.2. It's been filed and fixed. I am working in Processing 2.0b9. So maybe I have to prepare for an update. But than I have to update OSX too. Luckily I know now that it's a bug. That might also explain why all colors look so bland.

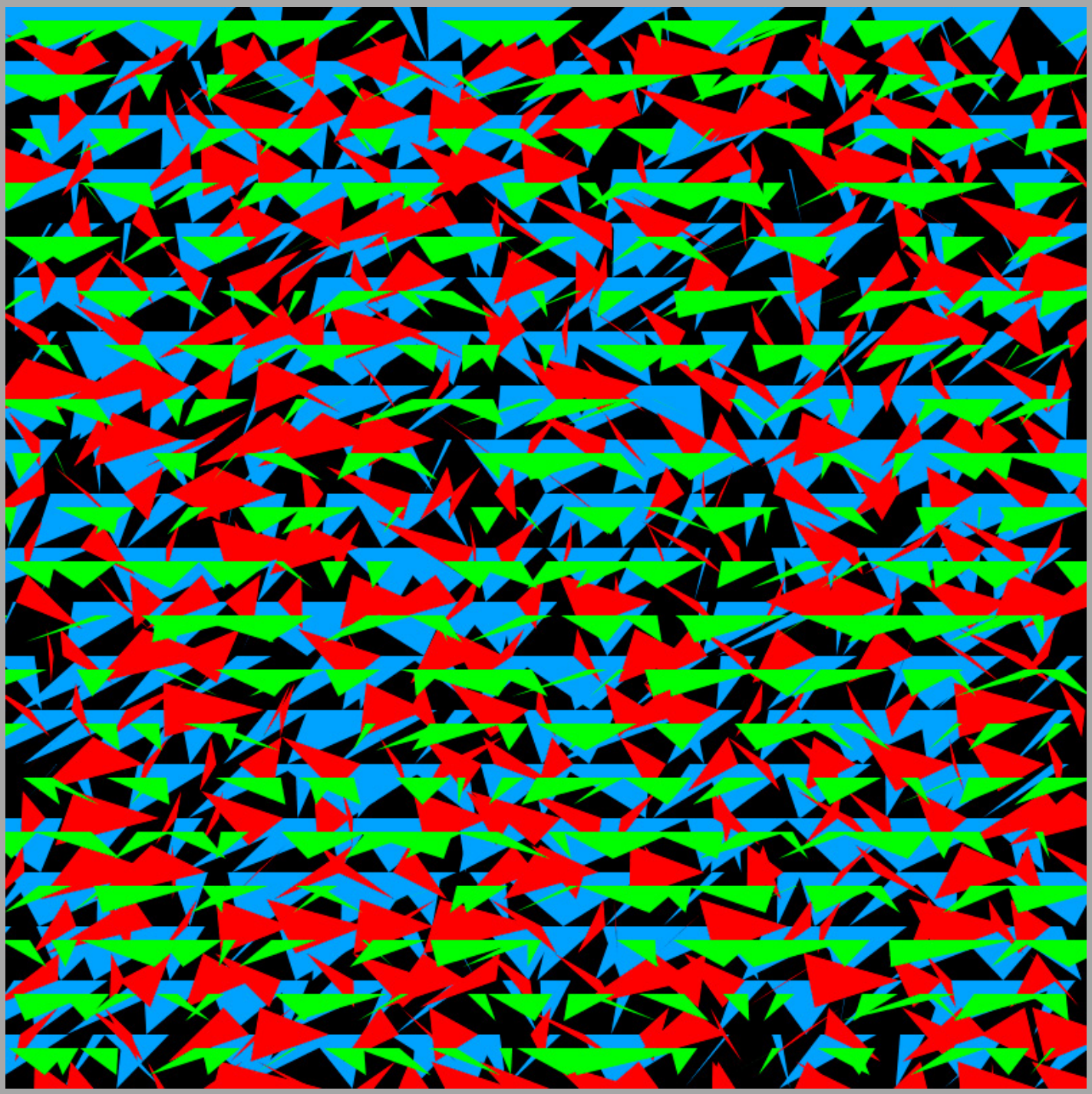
When I started with the fourth program I was thinking about drawing complex shapes into the beginShape / endShape block of the code. But because I was curious what would happen when I placed a copy on top of the existing layer of code I decided to wait with that idea for a while. I changed the colors so I could see the objects better. I commented out two of my lines of code which were basically the same. And I'm getting triangles! What happens if I comment out two different lines of code? I'm getting different triangles! And what happens when I comment out two of the same lines of code before the vertex is closed? I got even more different triangles! I decided to add another layer of green triangles and divide the size of the rectangles by four. Even more interesting!

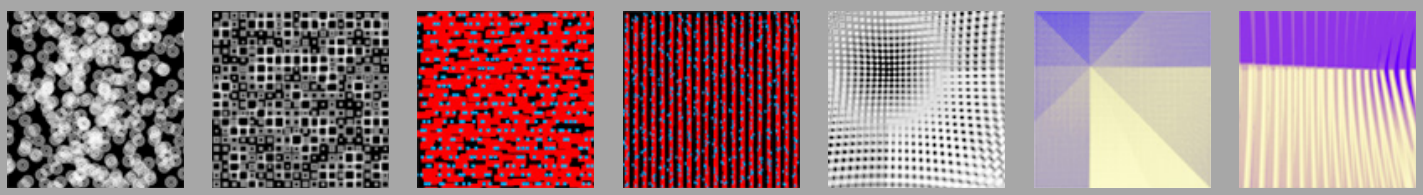
Sometimes it's nice that you don't have to do a lot of work to get another image. In this case I only had to comment out a few lines. Sometimes it's even better to do less to get more.

I went to lunch once in the Tin Angel. Along the way a guy came at me, he was wearing a sheepskin suit, in July, Yeah! He had a black badly threaded beard, and he said to me: 'I want to play in your band.' I asked: 'What do you play?' 'Nothing!' He replied. 'Okay', I said, 'You're hired!'

The Real Frank Zappa Book, Frank Zappa, Peter Occhiogrosso, 1994







Complex random patterns

MyCodeHistory: 27 February 2014

The program generates interactive growing and shrinking modules with random complexity in a grid. The amount of modules is connected to horizontal movements of the mouse. The movements of the modules are connected to vertical mouse positions. In fact there are five variations of the same program in the Generative Design book. I made twenty five variations of them. So it took me some more time then the previous programs.

I started with the first variation to replace the circles with arcs. The arcs are rotating randomly at —45, 0, 45 and 180 degrees. I have changed these values a few times. The arcs are shifted from their centre every time the program is started. I noticed that the file needs six seconds to load. I got a rotating beach-ball when I use another browser than Safari. But finally it worked. Every mouse click gives a different composition. If you click on the left side it works faster. To the right more arcs are displayed so it takes more time to show them.

The second program starts with a problem. The program is divided into three separate programs. All slightly different. They are defined under the 1, 2, and 3 key. But only the 1 and 2 key are functioning in JavaScript. In Processing they all work fine. I think it has something to do with the frameRate() in the 3rd program. Not sure though. This program also uses the 1, 2, and 3 key to switch drawing modes. This time that works perfectly. The horizontal position defines the thickness of the lines while the vertical position increases or decreases the amount of the modules. The images are getting more complex than the earlier ones. When repeated many times it still gives an impression of order. The program now displays three versions of colored lines on top of each other. All with slightly different settings.

In the third program I just uncommented some lines and increased the strokeWeight() to twenty pixels. I also used some transparency. Again uncommented several lines. Changed some lines a bit. Uncommented even more lines and played with the settings. How easy can it be?

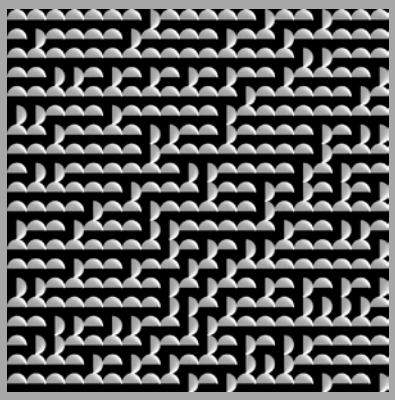
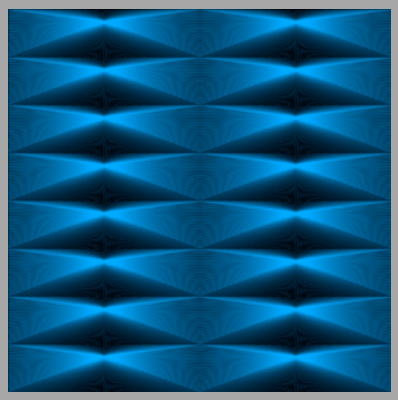
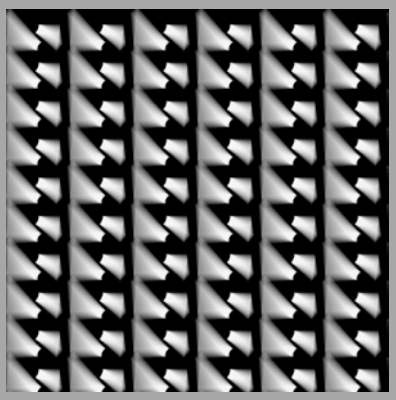
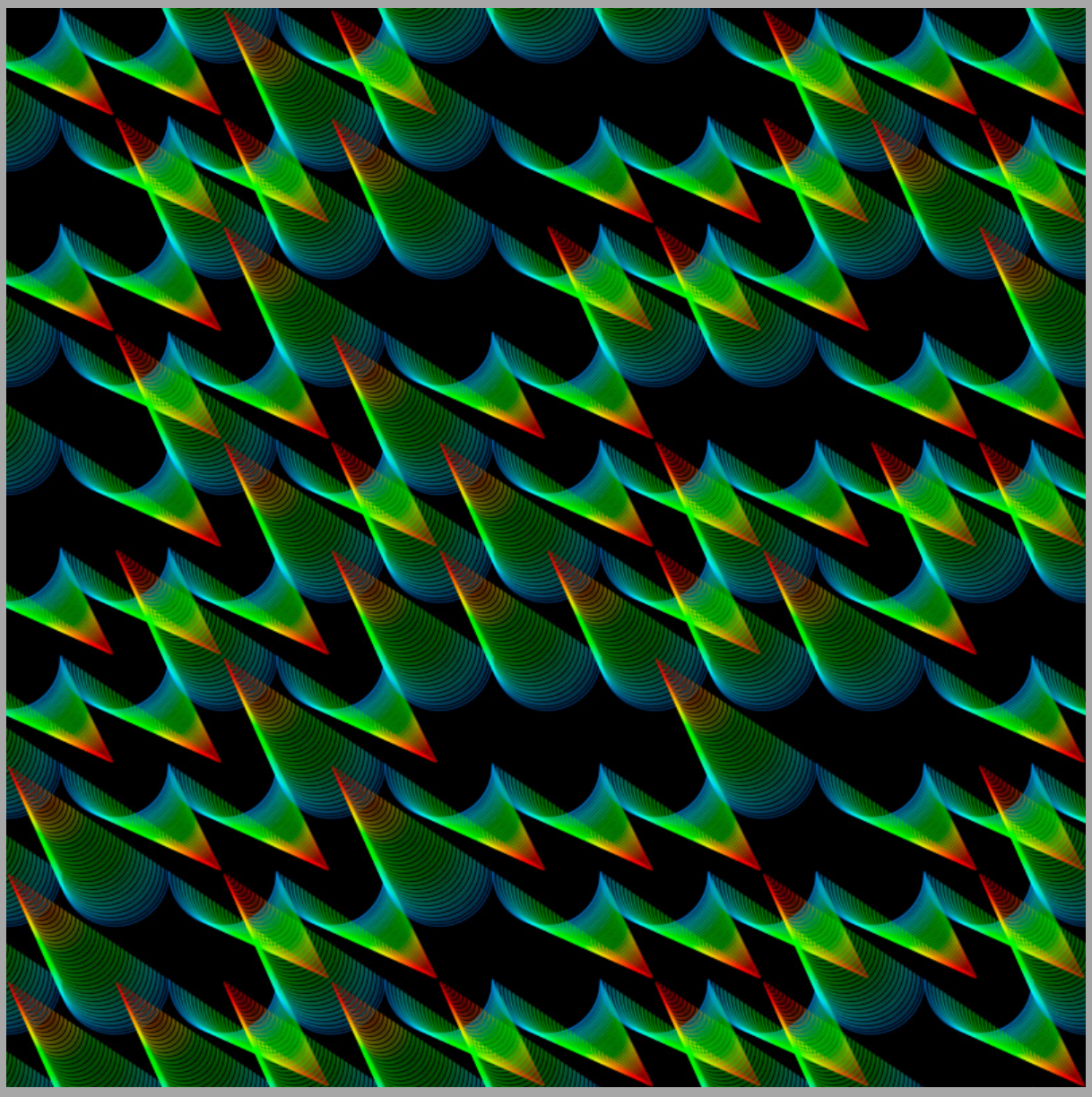
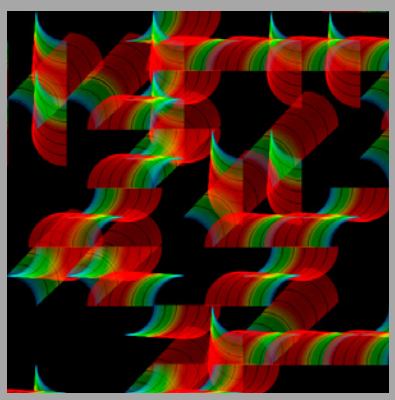
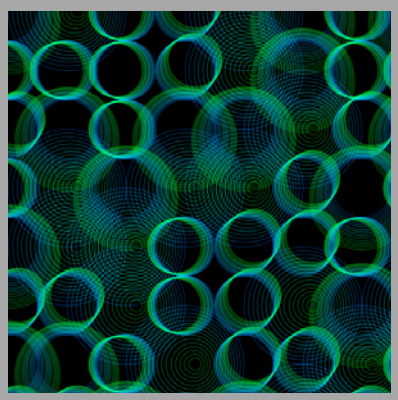
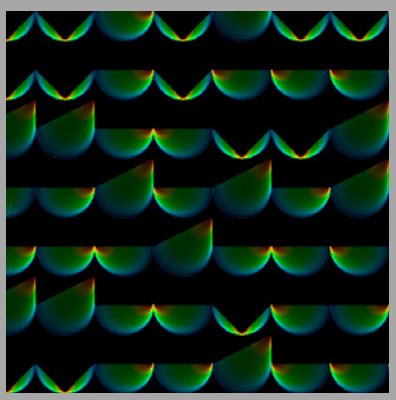
In the fourth program I changed the colors and tweaked the tileWidth and tileHeight variables. I defined some grey arcs under the 1 key. Sometimes I just change things because I want to see what will happen. I have no straight plan. On a certain moment I defined some grey arcs under the 1 key. They made interesting patterns. Especially when the cursor is moved in the horizontal or vertical direction. The 2 key introduces ellipses. And key 3 creates 3D-ish images when you point your cursor near the top right corner.

In the fifth version of the program I removed all the fill functions of the original program. I decided to work only with strokes. Why? I really don't know. Maybe it's because I thought that strokes would give more interesting patterns. Which it did. Replacing the circles by arcs shows even more the idea of a 3D environment. Not that I was looking for such an effect.

In the fifth program it is getting totally unpredictable if a Processing sketch is working in JavaScript or not. So I guess there might come a moment in time that I give up to convert sketches to JavaScript.

A student at the MIT (Massachusetts Institute of Technology) once asked Fuller whether he took aesthetic factors into account when tackling a technical problem 'No,' replied Fuller. 'When I am working on a problem, I never think about beauty. I think only of how to solve the problem. But when I have finished, if the solution is not beautiful, I know it is wrong.'

Richard Buckminster Fuller, 1895–1983, Writer, architect and engineer.

Stupid agents

MyCodeHistory: 6 March 2014

The original title of this chapter in the Generative Design book is: Dumb agents. But I have changed that for this description of the program because the next program is about intelligent agents. A stupid agent can only make one step during each drawing on the screen. Processing is running at sixty frames per second. During that drawing session the agent can choose random from eight directions: north, northeast, east, southeast, south, southwest, west and northwest. A tiny ellipse is drawn. This program has two variations in the Generative Design book. I was able to make fifteen modifications of the program.

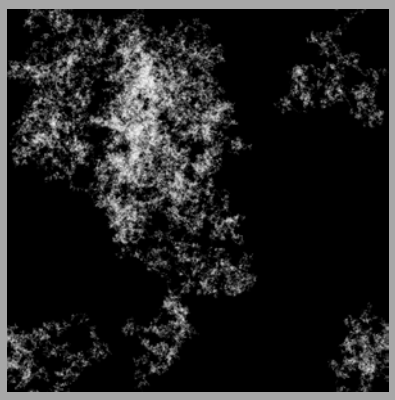
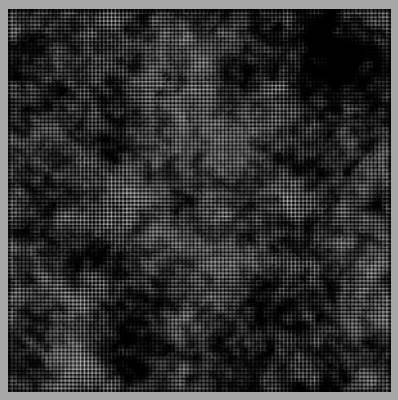
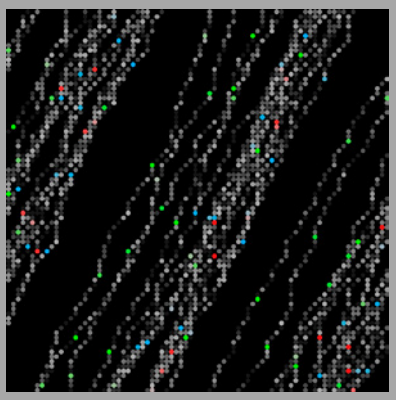
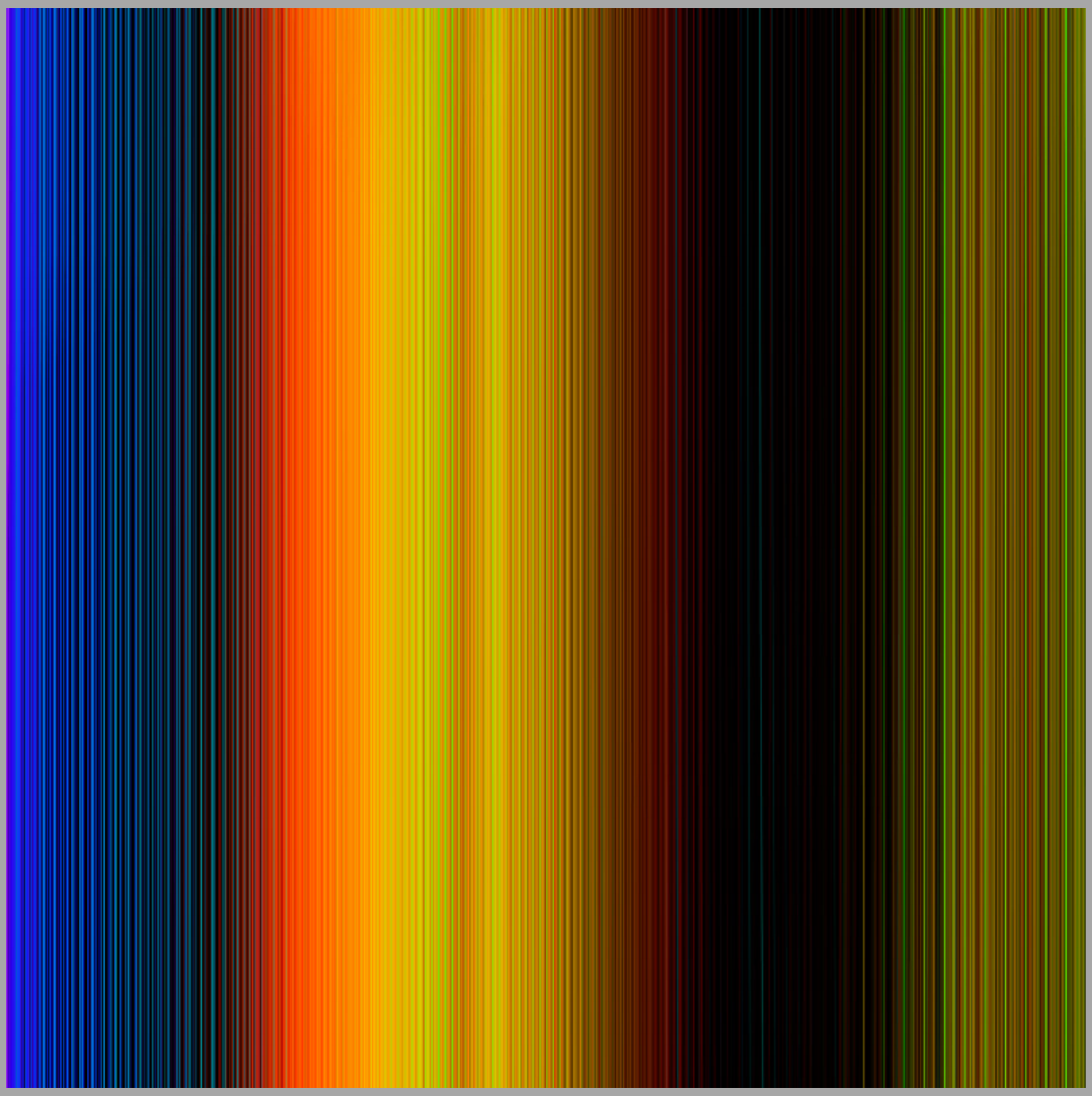
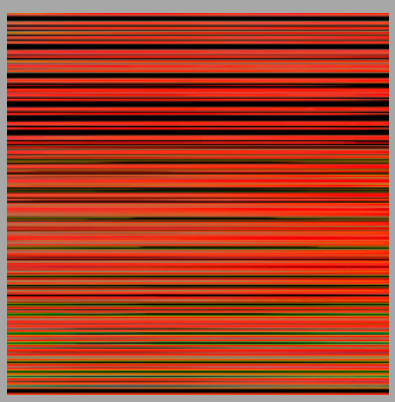
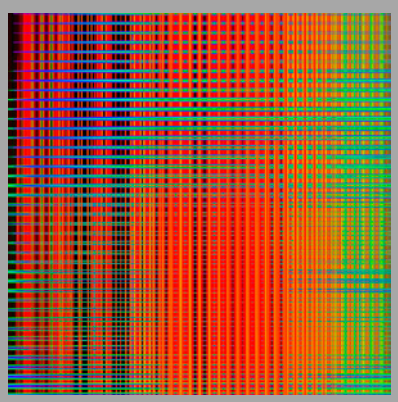
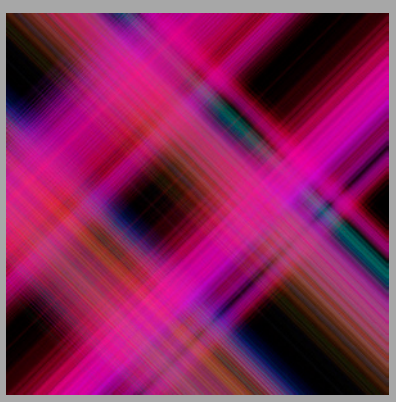
As a challenge I inverted everything in the first program. The program continues rather slow because it uses a very low transparency level. In the end I have used ellipses, arcs and rectangles to display the stupid agents.

In the second variation of the program I enlarged all ellipses. And when you hit the 1, 2, or 3 key you get different colors each time when hundred ellipses are displayed. I also commented out some lines which means that the agents will now only walk in the north and northeast directions. I played a while with switching the several directions on and off. But I ...thought that I could get more out of the program. So I continued a while until I saw that the program developed itself a little different. I changed the background color of the agents into shades of green. And around every four-hundred it displayed an agent without transparency. In the end I made the background agents shades of blue. And introduced some randomness in the diameter of the agents. But I felt that I still could go a step further. Just forget about the individual dots. But leave the positioning intact. Exaggerated the ellipses by making them very wide. They almost look like lines. In this case horizontal lines. And here JavaScript does not a fine job. It does not work at all. In fact I will not pay attention to the Processing to JavaScript conversion anymore. Because it is my intention to continue my study of Processing. I do not want to spend time on looking for mistakes (I made?) by converting Processing code to JavaScript.
I decreased the step-size of the agents. The longer you let it run the better the image gets. I ended up with horizontal and vertical ellipses to get a cross-wise image. Rotated that for 45 degrees.
I think I could continue for a while but I would like to stop here with the stupid agents.

It's of no use to me to go on and on and on. That doesn't work for me. There has to be a beginning and an end at a certain moment. Otherwise you loose your focus. Which is to bring my knowledge of programming on a higher level. I think you can only do that by doing small but continues steps. So studying a lot of different programs might be better than studying one large program.

Mahatma Gandhi was once asked what he thought about Western civilisation: 'I think,' he replied, 'that would be a very good idea.'

Mohandas Karamchand [Mahatma] Gandhi, 1869–1948, Indian statesman and spiritual leader.

Following a recipe

MyCodeHistory: 10 March 2014

In this program the (intelligent) agent moves in four directions: north, east, south, and west. But what makes this agent intelligent is that it can choose from several angles. Only right angles are not possible. When this agent reaches a border of the display window it turns around and randomly selects an angle. When it reaches a line (a previous path) it selects a new angle.

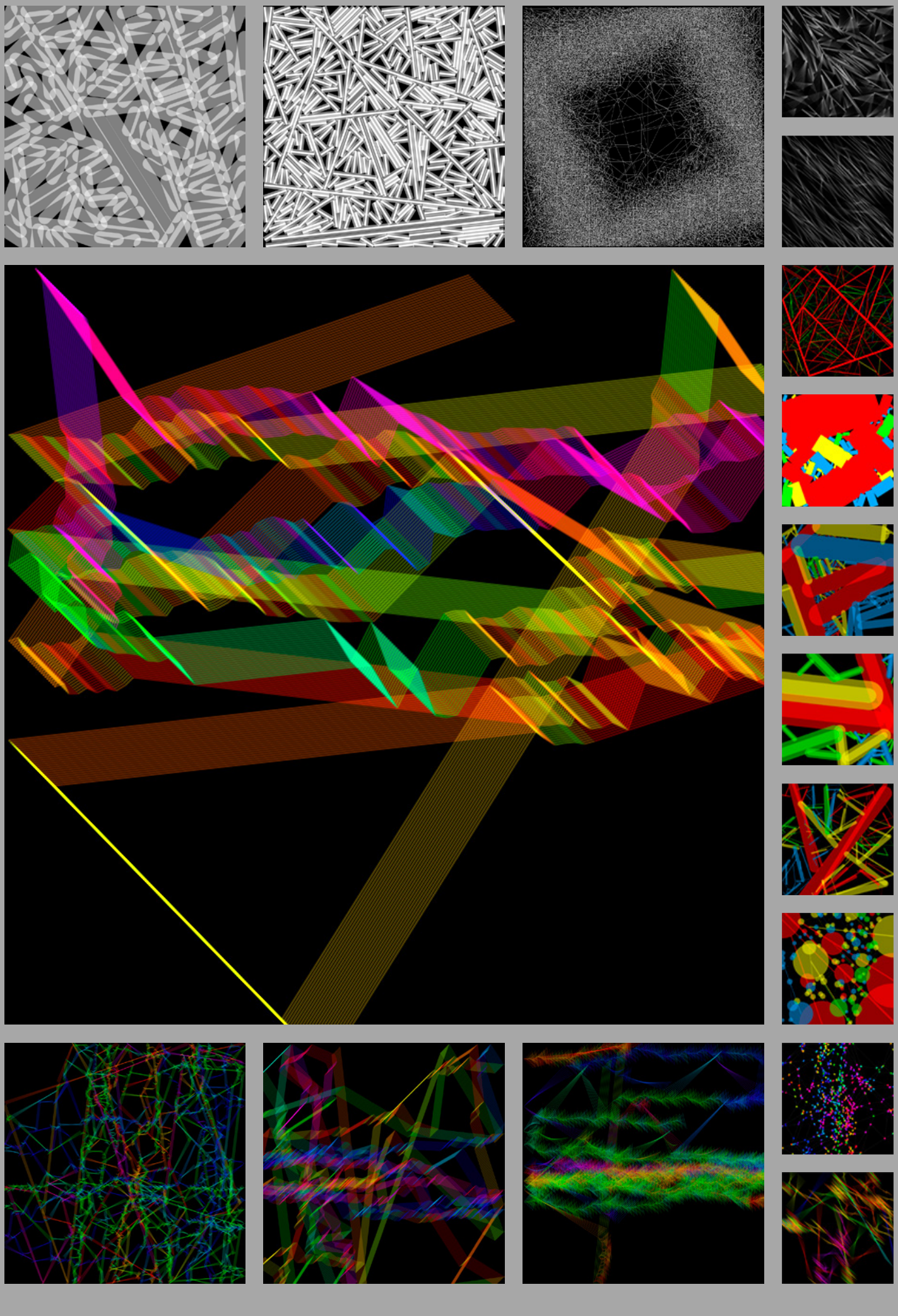
I divided the horizontal mouse position by a hundred to slow down the intelligent agent's movements. Otherwise it is too difficult for me to follow what it is doing. Increased the thickness of the lines from 0.5 to 40 pixels. Added some 50% of transparency. You know what? For a change I give you an impression of the things I did during modifying this program. Uncommented the strokeCap(SQUARE). This gives (as a default) lines with round end-caps. Increased the line thickness from 40 to 100 pixels. Made the strokeCap(SQUARE) again. The width of the line 20 pixels. I placed an extra line on top of the previous line. Changed the line thickness of 6 pixels with no transparency. Played around with float a. Increased it from 0.5 to 6.0. Increased the smallest length from 10 to 100 pixels. This means fewer lines because the lines which are shorter than 100 pixels are not drawn. Decreased the minimum length of a line back to 10 pixels. Increased the width of a line to 20 pixels. Increased the minimum length of a line back to 100 pixels. Repeated the drawing of a line 6 times. Replaced those lines by a for loop. Added 10 pixels to the horizontal and vertical position for every time the program goes through the loop. Lowered the minimum length of a line to 50 pixels. Used the rather strange number 135 which leads to these strange patterns. No idea why 135. But it seems ideal in this case. And than you have 10 folders with each 1 variation.

As you might have noticed its a lot about changing numbers an replacing functions. Or sometimes even adding new functions. For the second range of variations I used color. But that doesn't change much in the way of working. You only have more possibilities. Here is what I did. I added a 4th key function which adds an extra color to the previous 1, 2, and 3 keys. Changed the minimum length to 20 pixels. Changed the minimum length to 10 pixels. Commented out strokeCap. Which makes that lines get the default round strokeCap. Used 50% transparency. Reduced the stroke thickness by half. When the second line gets displayed it uses an angle of 5 radians (degrees). At the starting point of the horizontal and vertical position a point is drawn with a radius that depends on the distance divided by drawWeight divided by 40. This sounds complicated because I did not use the ellipse function but the point function. I did not know that you could draw ellipses by increasing the point thickness. Replaced the points by a for loop with ellipses. All lines are displayed with a transparent ellipse on top of the beginning of that line. At the beginning of each line ellipses are placed with a diameter of 10 pixels. Instead of one line two lines are placed with different start and end positions. At that moment I really did not know what do next. So I stopped and continued a day later.

I continued with the same code where I was left the day before. I noticed that there was a lot of repetition in the image. After I corrected that in the program I made a version in which I could make variations easier. But this also gave repetition in some horizontal and vertical areas. Then a graph-like version emerged which I might use later when we enter some data visualizations. It was interesting to see how these lines develop without any interactive intervention. I modified the program a little in a way that it gave interesting shapes at the cross-points. There is less repetition and more divers behaviour than in the previous sketches.

The strange thing with programming is that on a certain moment you can work on one sketch for hours and there seems to be no progress at all. But when you stop and come back some time later you might make several variations in just fifteen minutes. I see that as a positive issue.

Increased the thickness of the lines from 0.5 to 40 pixels. Added some 50% of transparency. Sprinkled some lemon juice, pepper and salt. Sometimes programming has a lot similarity with following a food recipe.



Mousing around

MyCodeHistory: 18 March 2014

This program is about a couple of points that are calculated in a circle. Those points are the starting points for the agents. The agents slowly leave their origin position when moving the mouse. In turn creating interesting traces. The story behind the variations I have made with this program is a bit different then the previous stories I have told about programs. Because this program gives you the possibility to draw your own shapes you have to do more than just wait for the images generated by the program. Basically it comes down to this: it does not matter who is using the program. He or she will always get different versions of images than I have made. I even don't know if I could re-create my own images because I did not save all the different modifications of the program. That was a stupid thing to do but that is the case. I was too enthusiastic at work and sometimes you would not like to be disturbed in that process.

To continue with the sketches the variable formResolution determines the amount of points in the basic shape. By reducing formResolution to 3 you get a triangle (with rounded corners). When you increase the formResolution to a 100 that transforms the triangle to a circle. Oh, I got puzzled by a variable with the name distorsionFactor. I could not find where or what-fore it is used. Another thing is that I did not use the filled version of this sketch. But it is nice that you have it as an option.

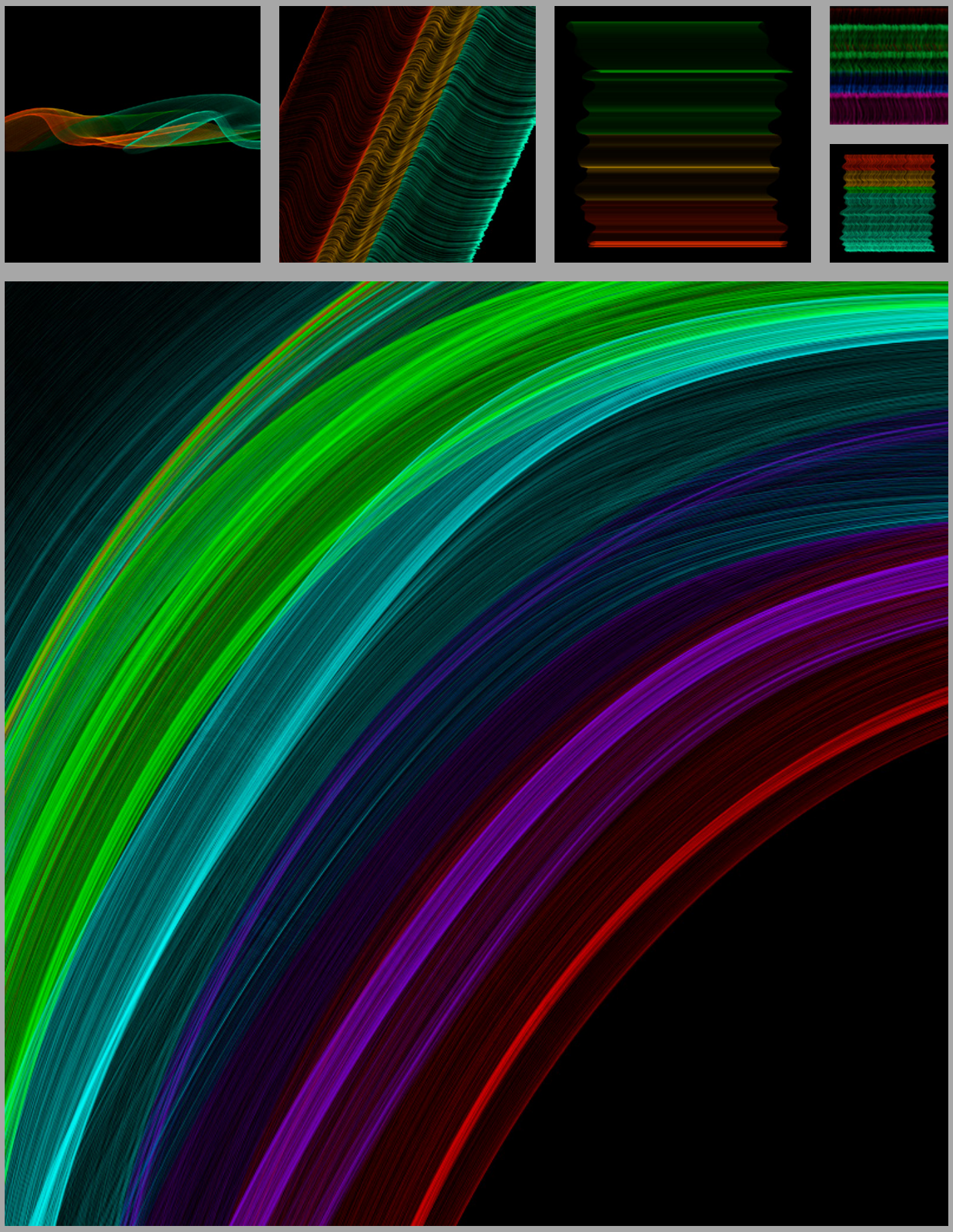
I changed curveVertex into vertex. Now only straight connections are made. As that was mentioned in the Generative Design book and as a comment in the original code itself. I also switched over to HSB colorMode. And I added four more drawModes. 1 For each color. Hitting the 5, 6, 7, or 8 key changes color. Although the 8 key does not make much of a difference (yet). I increased the color range by 90. Although I see I have made a mistake when drawMode is 1. The color range indicates 0 to – 30. But all four keys are now giving different colors. OMG, it seems that the JavaScript version is running a bit slower if you compare it with the Processing sketch. Point noted!

I had a short conversation with Erik van Blokland of Letterror. The reason is that I think that roughly 20% or less of my Processing files will not convert properly into JavaScript. Sometimes its due to me (making mistakes) but another part is much deeper hidden in the Javascript language (I think). So I asked Erik why the conversion between Processing and Javascript isn't 100%. Erik's reply: 'Because Javascript is a messy collection of fixes, and Processing is a well-designed language'. Well… I don't know if this is of any help but it still gives you an impression of what is going on between those two programming languages.

To create something totally different, you sometimes have to do strange things. At the end of the 40s (previous century) Jackson Pollock began to spread paint over a canvas lying on the ground. 'On the floor' Pollock said, 'I feel more at ease. I feel more involved, more part of my work because I now can walk around my painting. I can approach it from four sides and literally be 'in' the painting itself.'

Paul Jackson Pollock, 1912–1956, The Shock of the New, Robert Hughes, 1991.

Film-tip: 'Pollock', Ed Harris, USA, 2001.



The option of choice

MyCodeHistory: 25 March 2014

This program generates a new ellipse 60 times per second. It is created at a random position with a random radius. The program checks which ellipse is closest to the new to be created ellipse. The new ellipse is than drawn at the closest older ellipse. In the Generative Design book there is a reference to diffusion-limited aggregation. Which sounds pretty scientific. So I checked Wikipedia. Diffusion-limited aggregation (DLA) is the process whereby particles undergo a random walk together (due to a Brownian motion cluster) to form aggregates of such particles. And that sounds even more challenging. So I thought for now it's better to stick to the program instead of delving deeper into the theories of TA Witten Jr and LM Sander.

I changed the calculation of the original display size. I have no idea why the display size is calculated in such a complex way. The size is 167 x 3 in width and 241 x 3 in height. Changed that to 800 in width and 800 in height because I have done that with all previous programs. I also changed the static fill function with a random fill. This gives an interesting effect to the animation. Which was, by the way, a happy accident. I used that happy accident in a colored version. Which on it's turn than looked more like a living structure.

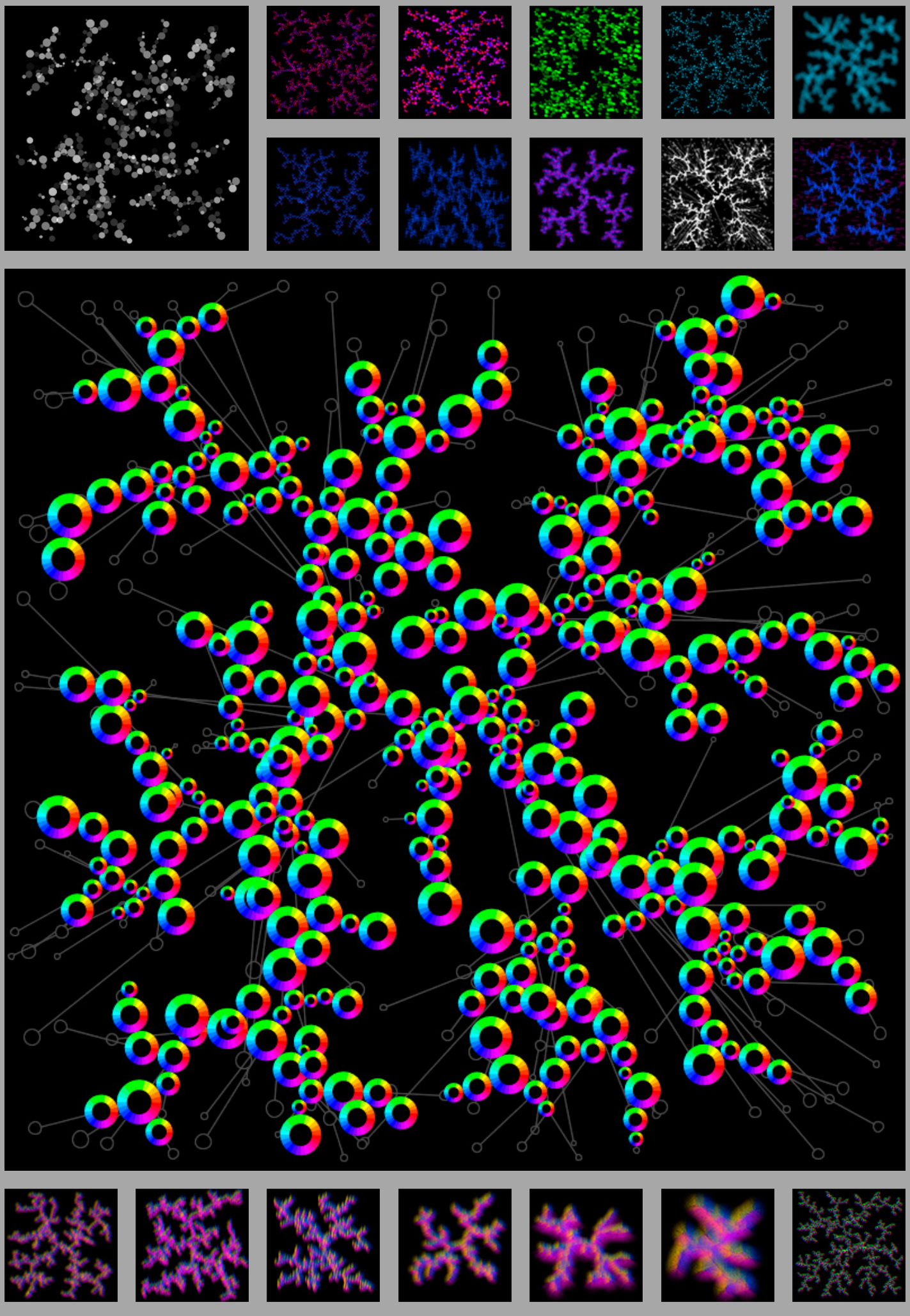
Replaced the ellipses by rectangles with rounded corners. To draw a rounded rectangle, you have to add a fifth parameter, which is used as the radius value for all four corners. You'll find this information in the Processing reference. That gave me the idea to draw simple leaf-shaped modules instead of rectangles. I have used different radius value's for each corner. Therefore you have to include eight parameters. A parameter is a reference or value that is passed to a function, procedure, subroutine, command, or a program. Which sounds very abstract. And, I have to admit, it is! Programming is always abstract. All rectangles are colored by a gradient. And I replaced the rectangle by ellipses. I liked that gradient idea. So I reduced the alpha value with 20% at every larger ellipse that is drawn. And if it works with ellipses than it works also with arcs. Added bottom arcs (radians 45, radians 135) and top arcs (radians 225, radians 315).

When the program is running and you hit the 1 key you will see that circles and lines are drawn at random positions in the background. All lines and ellipses have an alpha value of 20%. The 1 key has the on-off functionality for the background. But from here on I did not pay much attention to the background image anymore. I tried to concentrate on the foreground image. I could have spend more time on optimizing the code but I didn't. Generally I take as a rule of thumb that when it takes me more than one hour to optimize code than I do not optimize it at all. I think that an image is more important than how smart the code is written. While the earlier programs used circles positioned around each other. The last two examples make use of a color spectrum in a circle. The circle is divided into colored segments. On top of that a black ellipse is displayed. The end result shows some similarity with the colors you see appear in detergent.

In the last proposal I made the circles bigger and the black background circles smaller. In fact this is the best image of the range. It came to me when I had worked for two hours on an image. I could not find a new variation because I had a kind of designers-block. So I talked with Jeanne. She advised me to take a step back. Just make the circles a bit bigger and make the black background circles smaller. And there it was! Most of the time it helps to talk to others when you are stuck. They always see different opportunities in your work.

Giving up the option of choice, and letting a machine choose for you, is a radical approach to shrinking the time we might spend otherwise fumbling with the iPod's scroll-wheel. The Shuffle's approach is to generate random choices, but we can foresee a future in which the iPod knows your preferences, habits, and even your moods and will play music accordingly.

John Maeda, The Laws of Simplicity, 2006 MIT



Exaggeration and Keith Haring

MyCodeHistory: 1 April 2014

Every frame an ellipse is generated by the program. When it intersects with another ellipse the program starts again. Otherwise it calculates a possible new ellipse. The distance and radius are defined by the position and size of its nearest neighbour. When it bumps into it the ellipse is being drawn. But first something else. From now on I will define global variables with uppercase and local variables with lowercase. This is a tip I got from Andrew Glassner. I find that it makes the distinction between global and local variables better. Another thing which is a bit more complicating. Sometimes names in the code are not meaningful enough for me. For instance the global variable 'r' gives me not a slightest idea of what it stands for. Looking at the code it says that it has everything to do with the radius of an object. So instead of 'r' I call that global variable ObjectRadius. And because it's a global variable it starts with an uppercase. The same thing I do for newR. I will change that into newRadius. And it is starting with a lowercase because it's a local variable. This is going to make things much more readable and understandable for me.

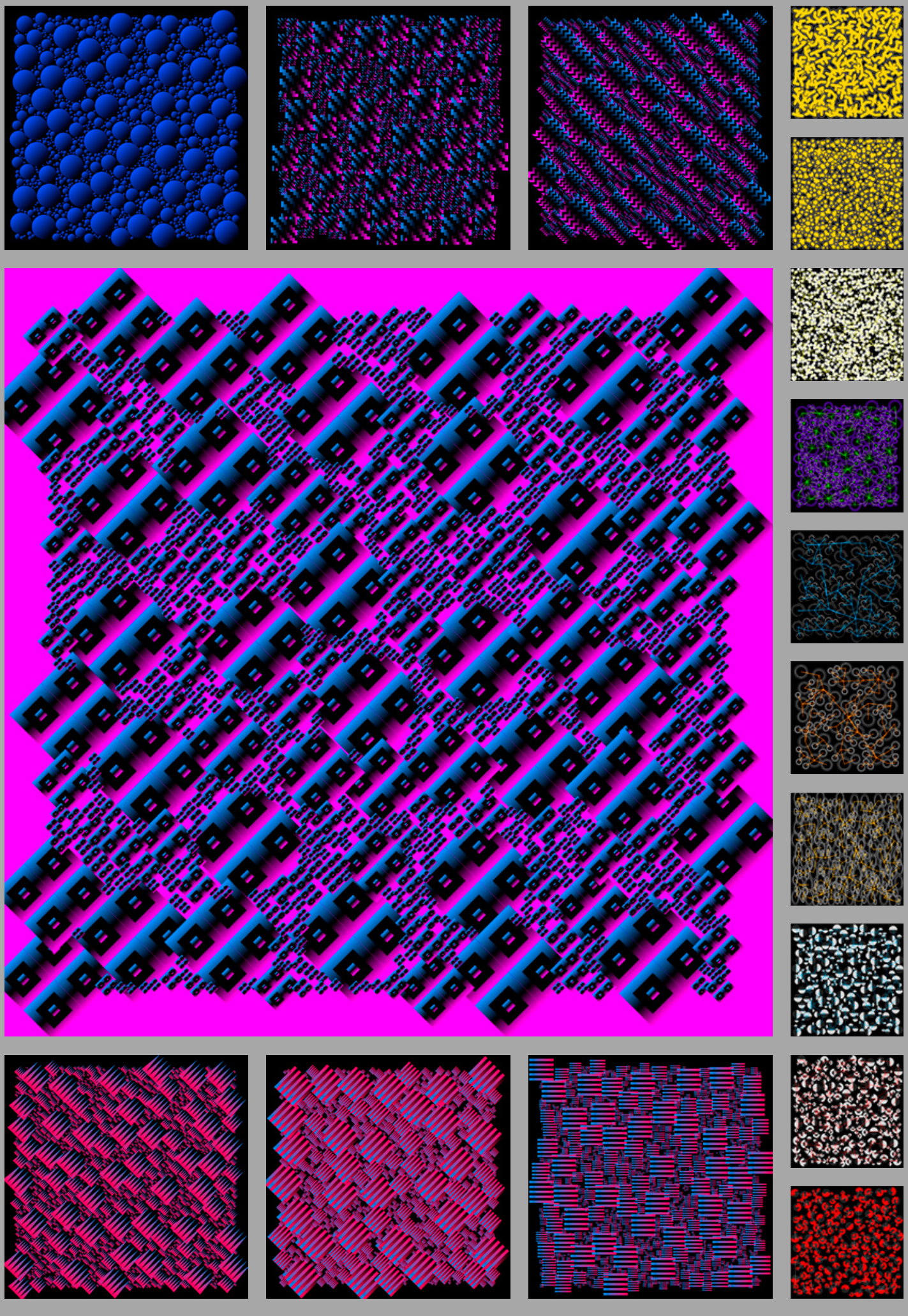
In the first sketch I exaggerated the line thickness. That generated Keith Haring-like symbols. Which is fine. On a certain moment I took a more generative approach. Instead of 1 ellipse 3 ellipses are displayed on the same horizontal and vertical positions. Moving the mouse up or down is going to increase and decrease the thickness of the lines and ellipses. Moving the mouse left or right is the same as moving through the HSB color range. A few sketches later I decided to make the size of the objects half the width of the height. If the height is 10 than the width is half of it. If the height is 0.7 than the width is 0.35. At a later stage I have cut every ellipse horizontal and vertical into arcs. Filled them with grey and white by reducing the brightness. And a version which is even more difficult to explain in text. I have cut the ellipse in half arcs. Vertical and horizontal. Filled the left half with white and the right half with grey. Then I made a smaller version of the same ellipse. It's a copy of the larger version. Used the same code as the previous program except for the fact that I introduced four steps in between. Instead of dividing the ellipses in half they are now displayed in quarter parts. The arcs may start now on 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°, 360° (or as you might prefer radians).

Now the last ten sketches did not work in JavaScript. All these sketches make use of color gradients. And because color gradients including radial and linear gradients are not yet supported in JavaScript it doesn't work. I didn't know that when I started with them a few days before. If I had known that I would have restricted myself to use only plain color fills.

I really hope that the next versions of Processing will give better JavaScript support. But the near future of Processing is looking very promising because on March 14, 2014, I got an announcement from Processing.org: 'Work on Processing 3.0 is starting. What is it? The priority is the Processing Development Environment, the PDE. We plan to integrate the work on PDE X to bring code completion, debugging, and more to writing Processing sketches. We're also working with a small, talented group of people on two other major projects to extend Processing in other directions to better support JavaScript and Python.'

Beware of people with only 1 idea.

Jean de Boisson (Cees Buddingh), 1918–1985



Drawing with time

MyCodeHistory: 6 April 2014

This is the first program that you can use as a drawing tool. It's based on a line that rotates around the mouse position. The joining of lines is based on the speed and direction of the mouse. In my case I was using a Wacom (pronounced 'Wok-um') Intuos 3 tablet and pen. With each pen-click the line length and color change. The rotation speed of the lines can be set with the left and right arrow keys on the keyboard.

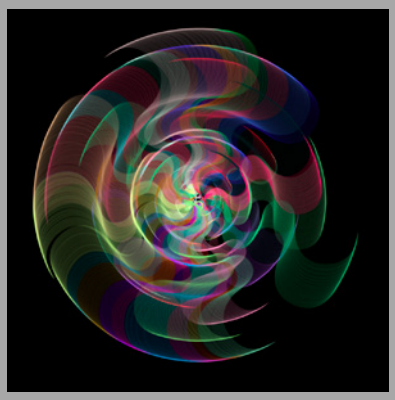
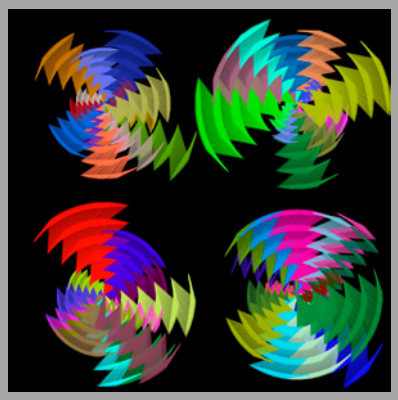
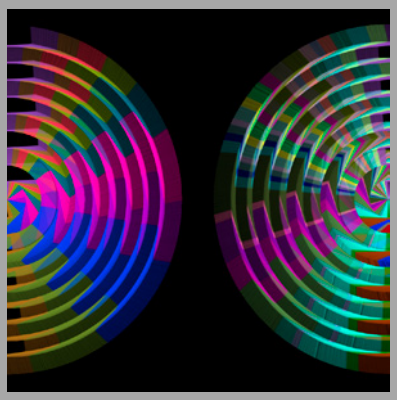
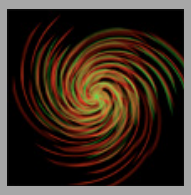
I must admit that there are enormous visual possibilities using this program. I was, for instance, able to create 91 different images with this tool. I tried to make the first image with the pen on the tablet. But that did not work. The pen is not stable enough to keep it on the same place during the drawing. So I replaced the pen by the Wacom mouse. And that worked very well. Position the mouse somewhere on the tablet and let the drawing tool do it's work. For the second range of images I kept the mouse positioned on the same place and let it finish a complete circle. I increased the strokeWeight to 50 and added a random transparency between 5 and 10. If you hit the space bar you get a random color for each drawn object. I increased the strokeWeight again but now to 100 pixels. LineLength is now a random number between 10 and 350. I noticed that I did not use the keys to control the default colors. I mostly used the space bar to get a random color. This has a drawback because sometimes you get very dark and muddy colors. So I defined a color range in the program. But to get interesting images from this program you have to invest time. And that is going to be the case for all sketches that follow after this one.

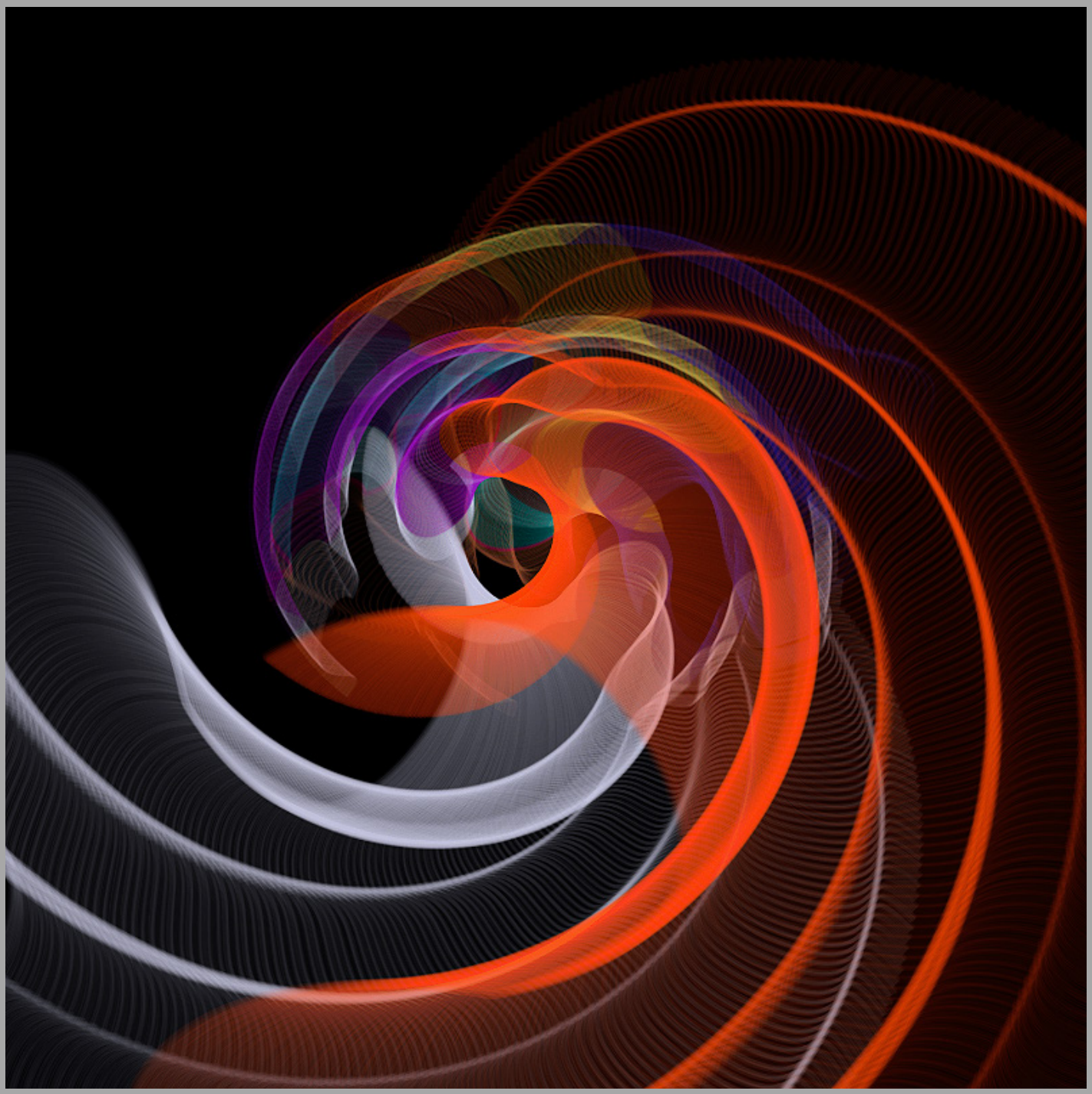
In a few sketches I used the default .svg files that were in the data folder of the original program. I lowered the transparency level to 20. The size of the .svg file is randomly chosen between 50 and 200 pixels. The circles in the images are random positioned by hand. I decreased the AngleSpeed to 0.5. This makes the movements very slow but in return it gives a better image quality. And image quality is more important for me than time. Some sketches work with svg-files which I made in Adobe Illustrator. I have put them under the 6-key. Commented out the other keys (7, 8 and 9). And I replaced those with my own .svg files. I used Code Line's Art Directors Toolkit 5i to precisely position the cursor in the images. When you are able to hold your mouse-cursor very still you can get undistorted glass-like images. In fact it is very difficult to keep the mouse straight at one point. That would be a good improvement for the program. This was a very fine session. I made a lot of images and I now have a better idea about how to make different images with this program.

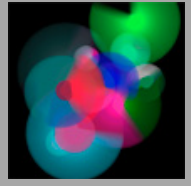
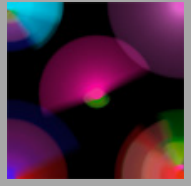
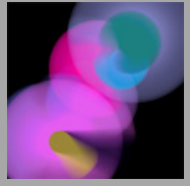
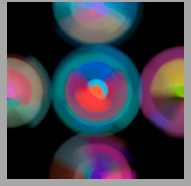
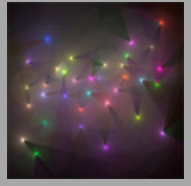
In 2010 I bought the book Generative Gestaltung. It was in German so I translated all accompanied text of the programs into English. After that I tried to work with some programs but I did not have a clue what they were doing. Let alone how they worked. So I waited for the English version which I received in 2012. And in April 2014 I arrived at a point in the Generative Design book were I left it about two years ago. The difference is now that I understood 75% of the code. Sometimes I know what the program is doing and why things are working (or not when I have modified it). That is an improvement. Time is an important learning factor.

Matisse's painting 'Le Bateau' hung upside down in the Museum of Modern Art, New York, for forty-seven days before anyone noticed it. In that period 116,000 people had visited the gallery.

Henri Matisse, 1869–1954,
French painter



Fast is thick, slow is thin

MyCodeHistory: 11 April 2014

In one of the earlier programs a new shape was drawn each frame when the mouse button was held down. In this program a new shape is only drawn when it stays a minimal distance from the previously drawn shape. I removed the very thin lines. And replaced them with very thick lines. They have a StrokeWeight of 120 pixels and a StepSize of 20. I also used the HSB colorMode because I think it is easier to work with. There are two DrawModes in this program. And I must admit that I only used DrawMode 1. I Increased the strokeWeight to 400 to get even thicker lines. Left all other settings as they were. As you might have noticed these images look like I was inspired by Daniel Buren.

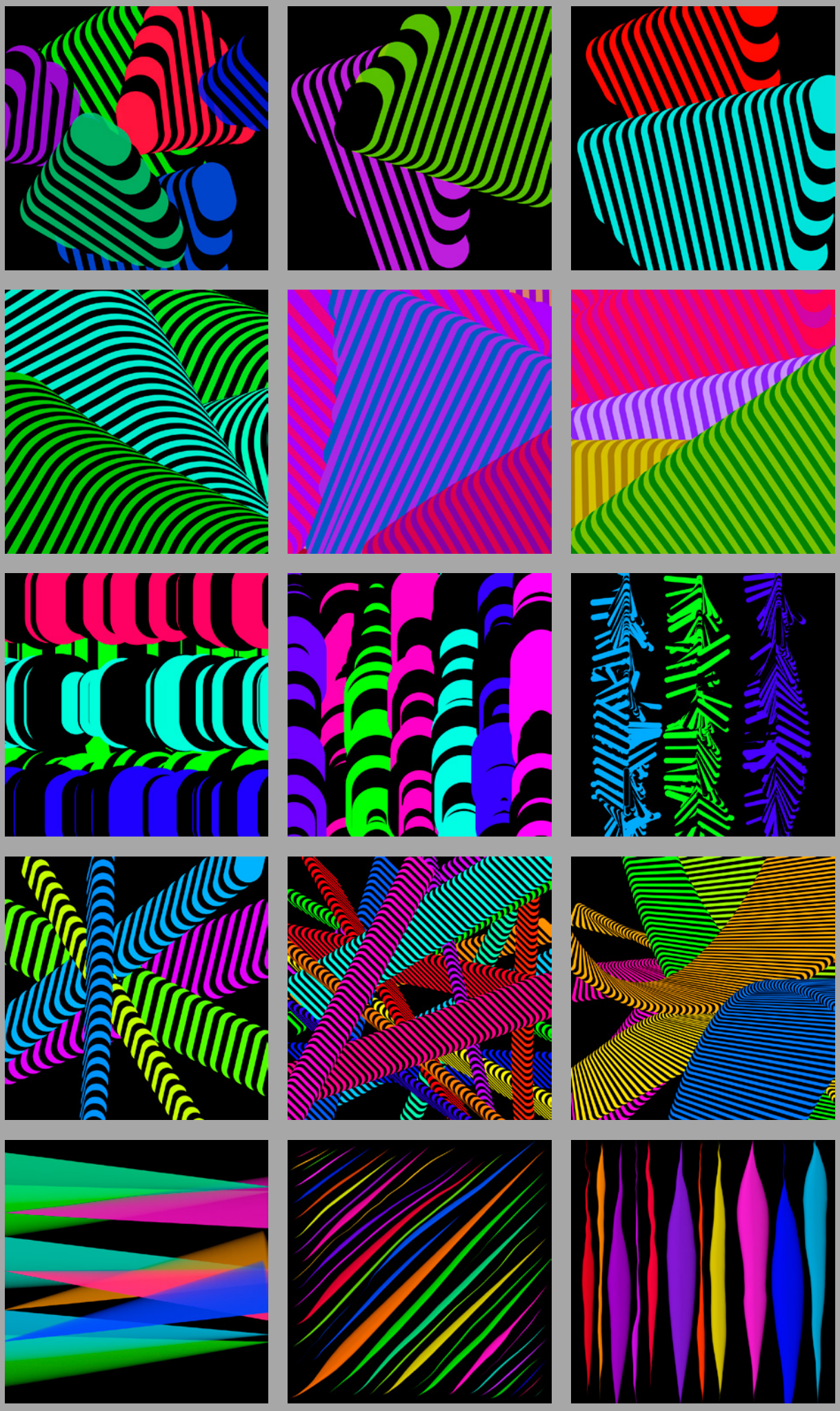
I would like to get away from these inspirational images which are looking very Bridget Riley-ish. So I used some randomness. I introduced a random strokeWeight between 0 and 20. And I also commented out the Y_Position locations. This means that you can only use the horizontal interaction. And that is what I wanted. It is also possible to get very straight lines. Therefore you have to position the cursor at a point close to the side of the display window. And point the cursor again near another side of the display window. A straight line is drawn. It is also possible to add a more circular behaviour in the objects. To calculate the X_Position of the object I divided StepSize by 2. To draw the line I also divided the Y_Position by 2.

At that moment I decided to start from the beginning. Changed the StepSize to 2 and the LineLength into 3. Commented out the line function and replaced it with the arc function. This gave me thicker arcs when I move the mouse very fast from left to right. Or from right to left. In the end I replaced the stroke function with a fill. That gave me triangles with very subtle gradients. Something I did not now that it was possible when I started this sketch. In the last sketches I made 3d-ish shapes. When you move your cursor fast the size of the shape will get thicker. Slow movement will create thinner shapes until the shape ends in a line.

With a toolbox like Processing you never know what kind of images you will be creating. The possibilities, opportunities, problems and solutions are so immense that it is even not possible to go right at your target. I think that it was never easier to make different images. Although you sometimes end up with look-a-likes. Which, in turn, could send you in a different direction.

One of the things I've said many times in interviews is: 'Progress is not possible without deviating from the norm.'

The Real Frank Zappa Book. 1992, Frank Zappa, Peter Occhiogrosso



Painting with characters

MyCodeHistory: 18 April 2014

In this program the position and size are connected to the speed and position of the cursor. In that way you can paint random series of characters. The text that appears is defined in the program. For this session I used one of our own fonts (FIOTEXTRegular). This font was specially designed by Jeanne de Bont for our animation film 'Fantasy in Orbit'. Now, the intention of this program was to paint images while switching brushes. But I did not want to make a portrait of someone. Or a still life scene with food, flowers and some dead animals. I tried to keep it as abstract as possible.

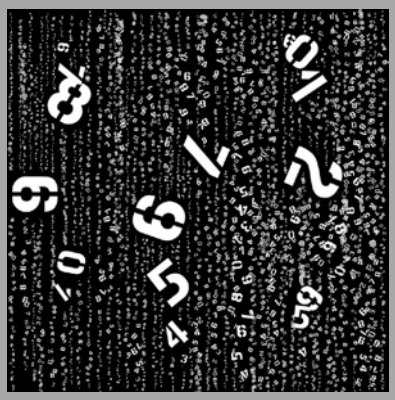
So I commented out the PaintingPos_X line. And as a result I could only draw text as vertical lines. And that is what I was looking for. I also used the rotate function. As it was written in the original code. But I found it much too random. So I replaced it with a less random function between 0 and 45 degrees. But in the end I thought only 45 degrees was good enough to work with. In the beginning I used numbers instead of text. I was thinking about writing some poetry. But this could easily become a project on itself. So I decided to stick with the numbers for a while. I used the space bar to get a random color for the type.

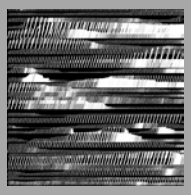
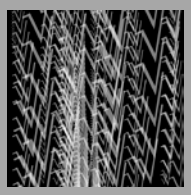
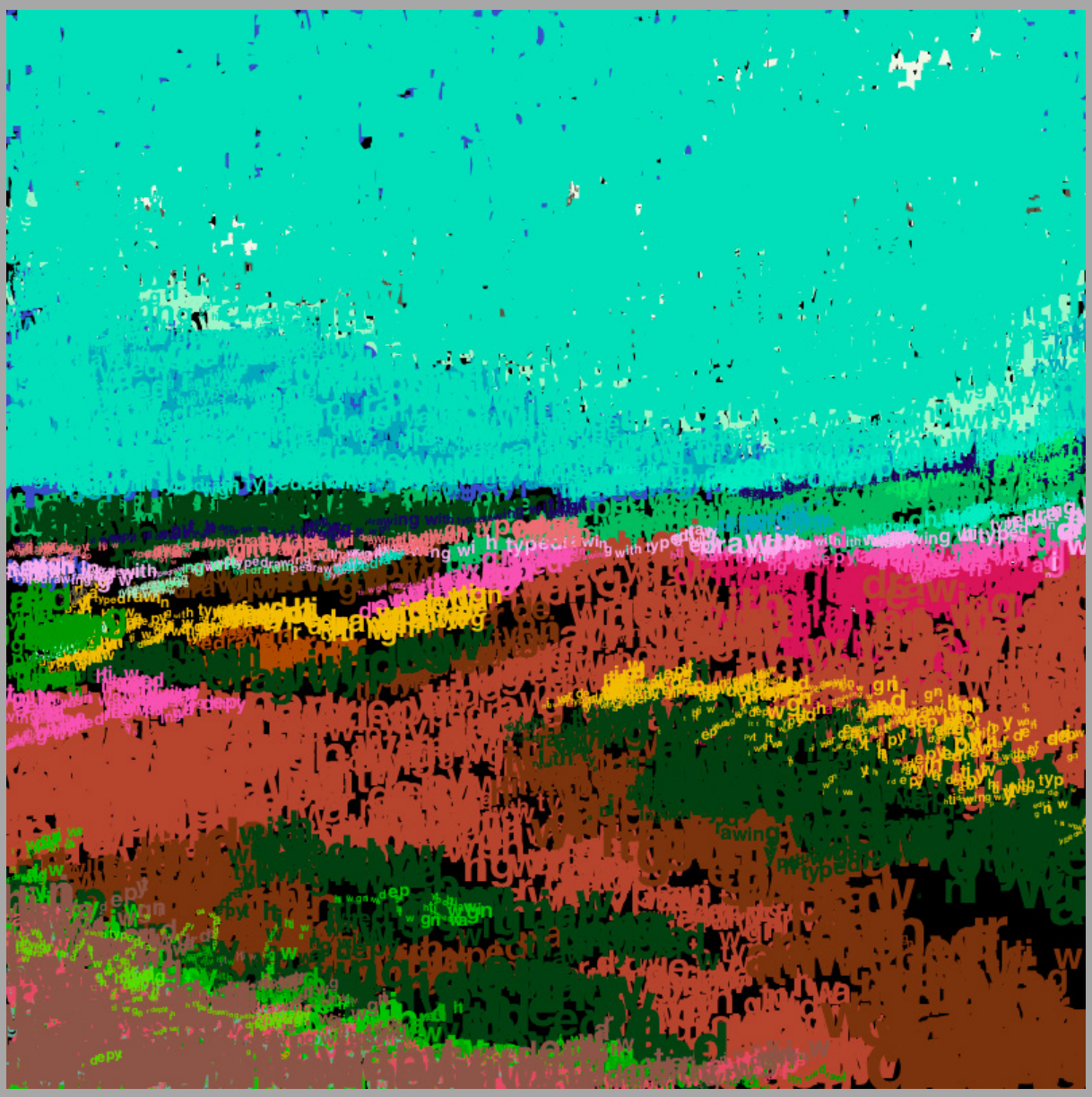
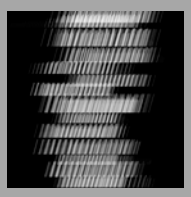
On a certain moment I decided to use only punctuation characters for a while. Did some experiments with several point-sizes of the solidus (or slash), parentheses, points, comma's, vertical bars, left single quotation marks, division signs, the almost equal to characters, square root and tildes. Especially the tilde seemed to work very fine. And I ended with some fictional random type landscapes. In the beginning I was thinking of making a lot of those but finally I thought it might be a little cheesy. I interpreted the name of this program in the Generative Design book very broad. Instead of 'Drawing with type' I was more occupied with 'Painting with type'.

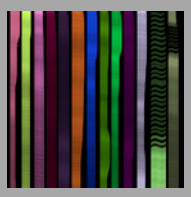
It almost doesn't matter what kind of brush-shape you use. If it can be used on a large and small scale it will work for anything you would like to paint. Abstract, landscapes or for decoration purposes.

I found I could say things with color and shapes that I couldn't say any other way — things I had no words for.'

Georgia Totto O'Keeffe, 1887–1986, American artist.

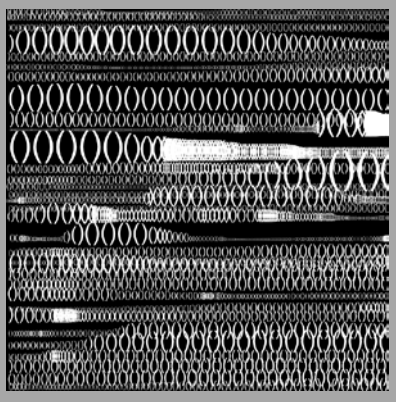
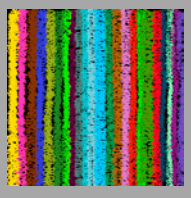
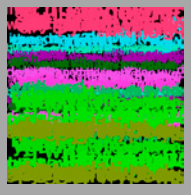








Connected brushes

MyCodeHistory: 12 May 2014

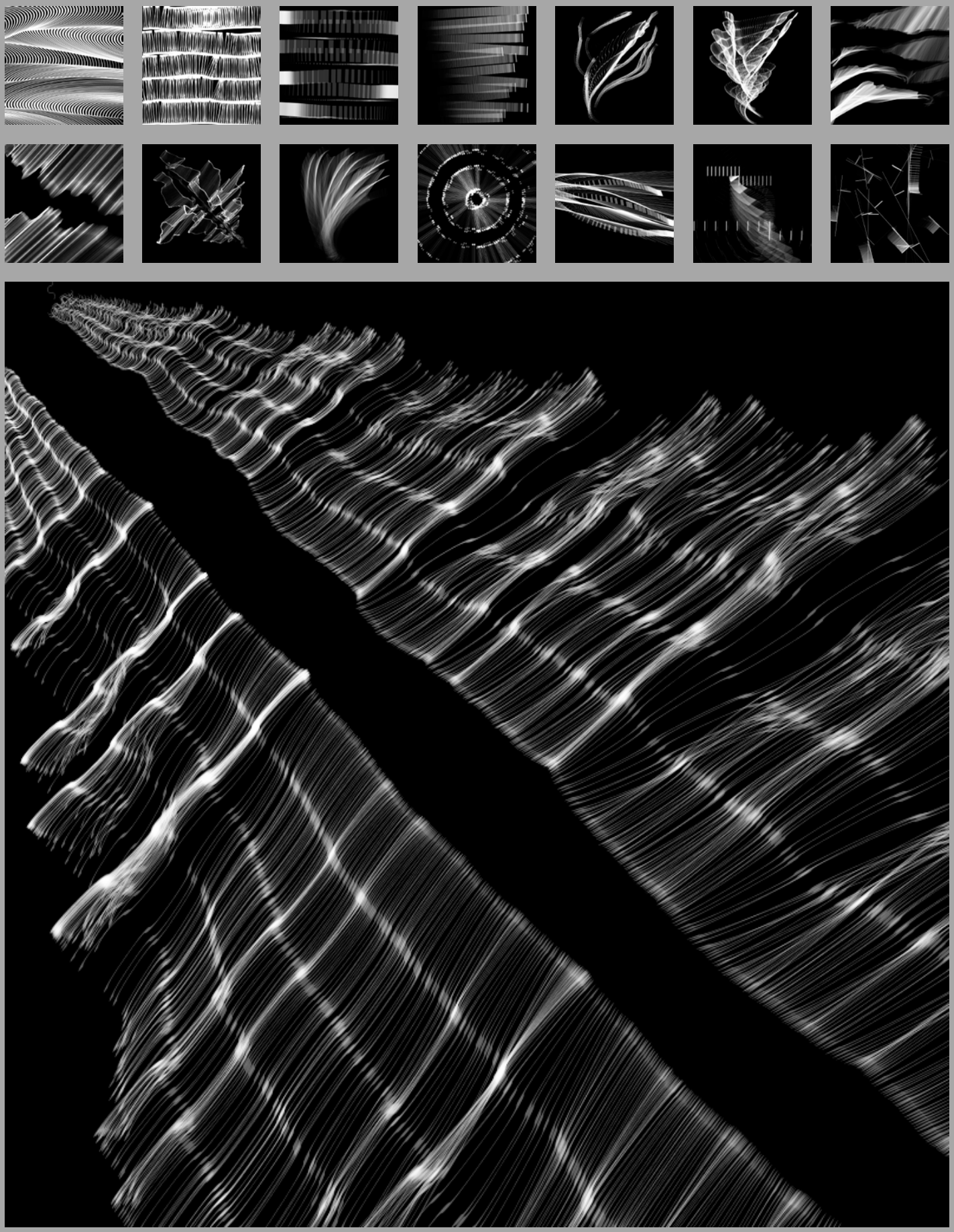
This program lets your cursor behave like its connected with a rubber band to a lazy agent. Fast movements give large stretched repeated objects. While slow movements give small repeated objects. Again this time the work progressed very different compared to working with the previous programs. Maybe it was because there was a week of vacation in between. When I started with the original program I forgot to think about it. I just started to make variations. It gives you so much possibilities that it is difficult to stop. Finally I ended up with 103 images. I reversed the original .svg files which were delivered with the program. Using them I could draw wavy, repeating, architectural, horizontal and vertically banded patterns, organic plant-like and circular structures.

After I used the 'standard' .svg files I started using my own svg-files. In the beginning I used very simple shapes but when those worked I was sure that it would also work with more complex ones. I modified almost nothing in the original program except for multiplying mouseX with 2. That's all. Some of the later images I made do remind me at the 'Proun' series El Lissitzky made around the 1920's. Although I think that it needs much more refinement to get any closer to his work. This was a very short session but it was a useful one. Especially because this program is simple. You can make a lot of variations with little changes in the code. I have no idea why I continued working in black and white only.

To make interesting images you do not need a complex and large program. You only need a program which does things intelligent and simple.

'The simplest way to achieve simplicity is through thoughtful reduction.

The Laws of Simplicity, John Maeda, © 2006 Massachusetts Institute of Technology



Turning the tablet

MyCodeHistory: 15 May 2014

My Wacom tablet is going to be the main player in this program. Compared to a mouse a tablet has a few more parameters to work with. Pressure and the pen's horizontal and vertical angle define rotation, saturation, and the length of a shape. The shape is not loaded from an .svg file but drawn as an easy to manipulate curve. As a start I tried to find out why the function frameRate is used in the original code. So I commented it out (which is the same as switch it off). But even when switching off the program just kept on painting the objects. When I decreased the frameRate to 10 frames per second, the drawing lags behind. FrameRate (30) seems to be right but I'll switch it off anyway. While checking what the other keys were doing I thought what would happen if I change the orientatation of the Wacom tablet itself? Well, it seems that it influences the angle as-well. And you can draw curtain-like images with it. Again I did not feel the need to make life-like images. I like to work more abstract.

I did not like the white stroke around the objects so I switched it off. Did some experimentation with the grey color. Replaced it by almost a full red. If it can be red it can be every color. Eh… case 3: the line of code says, color julia; I assume that must be a joke referring to Julia Laub of the Generative Design team. Another thing I didn't like is the shape of the drawing-object and changed the elementLength (or penLength) to 2000. This gave the possibility to draw shapes that remind you at cloth or silk-like structures. In the initialization part at the top of the program I changed the variables FromColor and ToColor to yellow. Gave it an alpha of 20. And I think some color adaptation is needed in the keyReleased block so I changed that and gave them all an alpha of 20. The h1 and h2 local variable names are a bit too cryptic for me. So I changed those to radiusTop and radiusBottom. Might not the best names but it is less mysterious for me. Removed all random things and gave radiusTop and radiusBottom a value of 2.

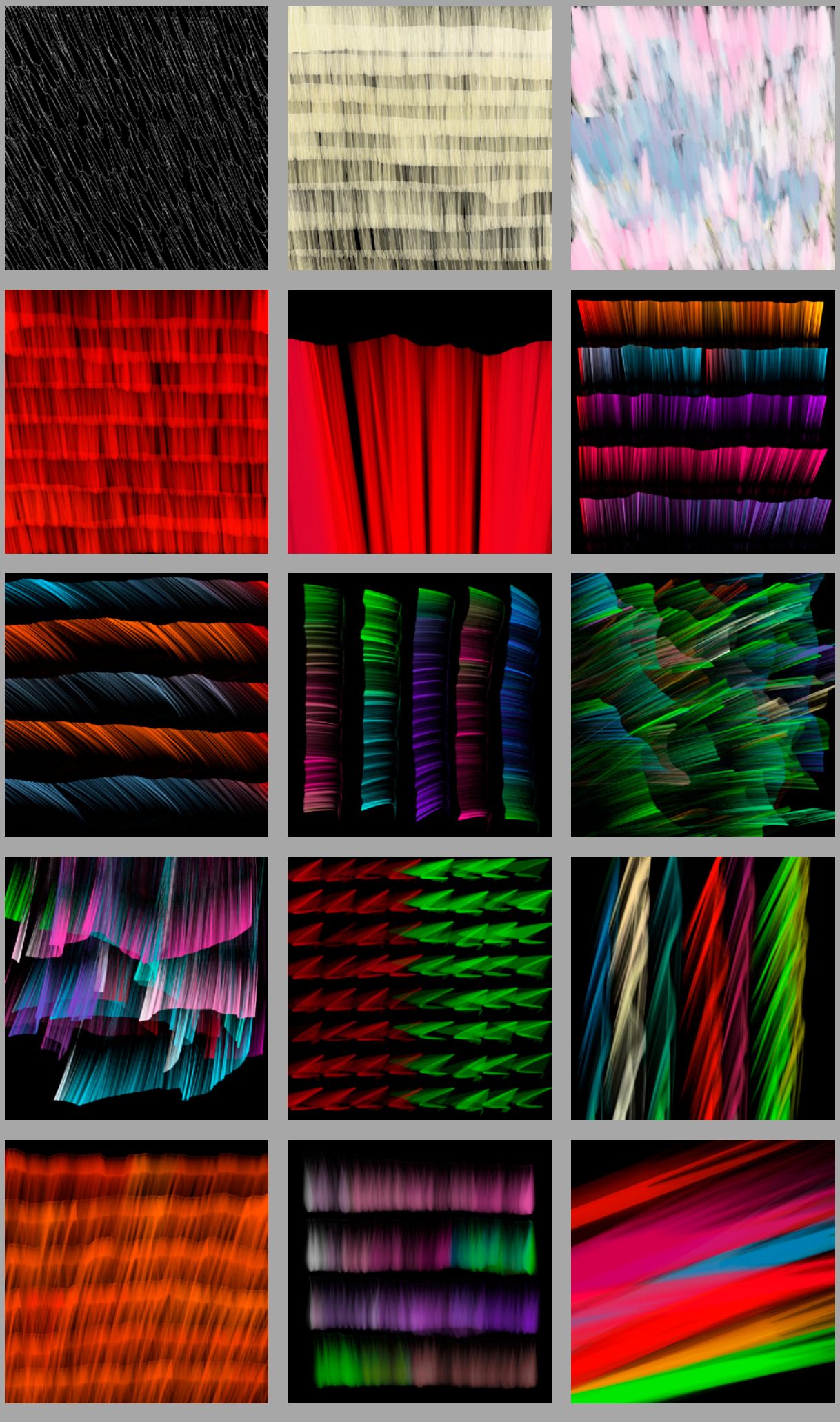
What would happen if I give the two radii negative values? For instance —20? Well… not much actually! I changed it now to —100 and there seems to appear a kind of curve. Lets exaggerate that and change it to —1000 for both radiusBottom and radiusTop. Now I got straight lines. Lets divide them by —500. Still doesn't make sense. —250? Back to where we started? —100 Than? Changed radiusBottom and radiusTop both to —50. Changed julia to green (0, 255, 0, julia). Changed elementLength equals penLength times 100. Changed the randomness of radiusBottom and radiusTop to a 100. But I do not like that the element stays green. The 6, 7, 8, 9, and 0 keys have no influence on the green color. So I commented out julia and replaced it with the code of case 2 (with a little change). I also changed all alpha's from 20 to 40 because I thought that the image would better stand out.

After drinking an espresso I changed (just for fun of it) the tablet's physical position 90 degrees anti-clockwise. That gave a totally different image when drawing with the same settings. Did the same with a 90 degrees clockwise change of the tablet. Tried one with 180 degrees. That was a difficult task. But it gave me an interesting image. Switched penLength back to its original setting and started to draw a kind of Northern Lights vertical swinging lines. After a lot of tweaking and adjusting I doubled the elementLength and made horizontal movements with the pen on the tablet. I multiplied penLenght to 1000 and made vertical movements to get transparent plastic-foil-like images.

There was a lot of adjusting and modification going on in this program. Even changed the physical position of the Wacom tablet to get interesting images. And in the end I stopped with a program that was almost the same program I had started with. And I think that is fine. We are not dealing with the patients of House MD. They arrive in the hospital in a different condition than when they leave.

During his years of poverty Balzac lived in an unheated and almost unfurnished garret. On one of the bare walls the writer inscribed the words: 'Rosewood paneling with commode'; on another: 'Gobelin tapestry with Venetian mirror'; and in the place of honour over the empty fireplace: 'Picture by Raphael'.

Honoré de Balzac, 1799–1850, French novelist



Modules and structures

MyCodeHistory: 24 May 2014

This program let's you draw with small modules in order to create large structures. With the mouse (in my case a pen) you draw various svg-modules in a grid. A grid position can be filled or left empty. If it's filled an svg-module is selected. Depending on the grid positions around the filled grid position other related svg-modules will be selected. Which in the end make unforeseen structures or patterns. But first some household affairs. There are a few functions in the program which I have used before but forgotten about. What does textAlign(CENTER, CENTER) do? Checked Processing's Reference. It centres text horizontally and vertically. What does round() do? Ok. It rounds a number. Up or down? To the integer closest to the n parameter. And nf()? That is used to add zeros to the left and/or right of a number. Eh… floor()? Calculates the closest int value that is less than or equal to the value of the parameter. And constrain()? Constrains a value to not exceed a maximum and minimum value. I never have used str(). It returns the string representation of primitive data types and arrays. Unbinary() converts a string representation of a binary number into its equal value.

A thing I would like to try is to make the program mirror a copy or reflect horizontally. That would give me symmetric images. I thought that swapping the x and y positions would do the trick. That did not lead to any success. I searched for a flip image function at the Processing forum. Found a mail of February 21, 2008, from Daniel Shiffman where he mentions that scale (—1.0, 1.0) will flip an image. Together with translate and a repetition of the shape line that works. Some minor flaw makes debug mode behave not as expected. Made a few versions. But it didn't work like I would like it to work. Some connections between the modules seem not be as good as I thought. And also the line quality isn't always as good as in the original code.

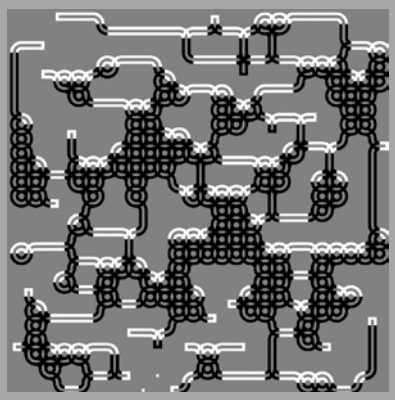
I uncommented the symmetric part and would like to go more into detail. So I changed the TileSize into 10. This gives even more interesting patterns. This could be (with a little more functionality) an ideal tool for designing maps. I could replace the grid on/off function for displaying a map. In that way I could use any image to trace a map. Or even better. Let the DrawGrid function stay as it is and make a new function called DrawMap. I'll import an .svg file with a map of The Netherlands. It seemed a little boring to show only the borders of the country. So I added a free interpretation of rivers to make it a bit more interesting. With the debug mode switched on it is also very nice. Just for fun I'll draw a few more maps for the neighbours: Belgium, Germany and England.

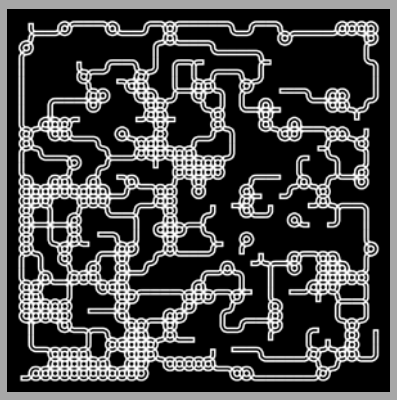
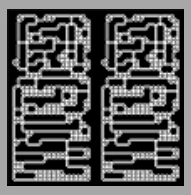
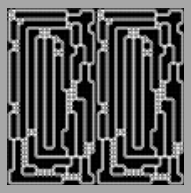
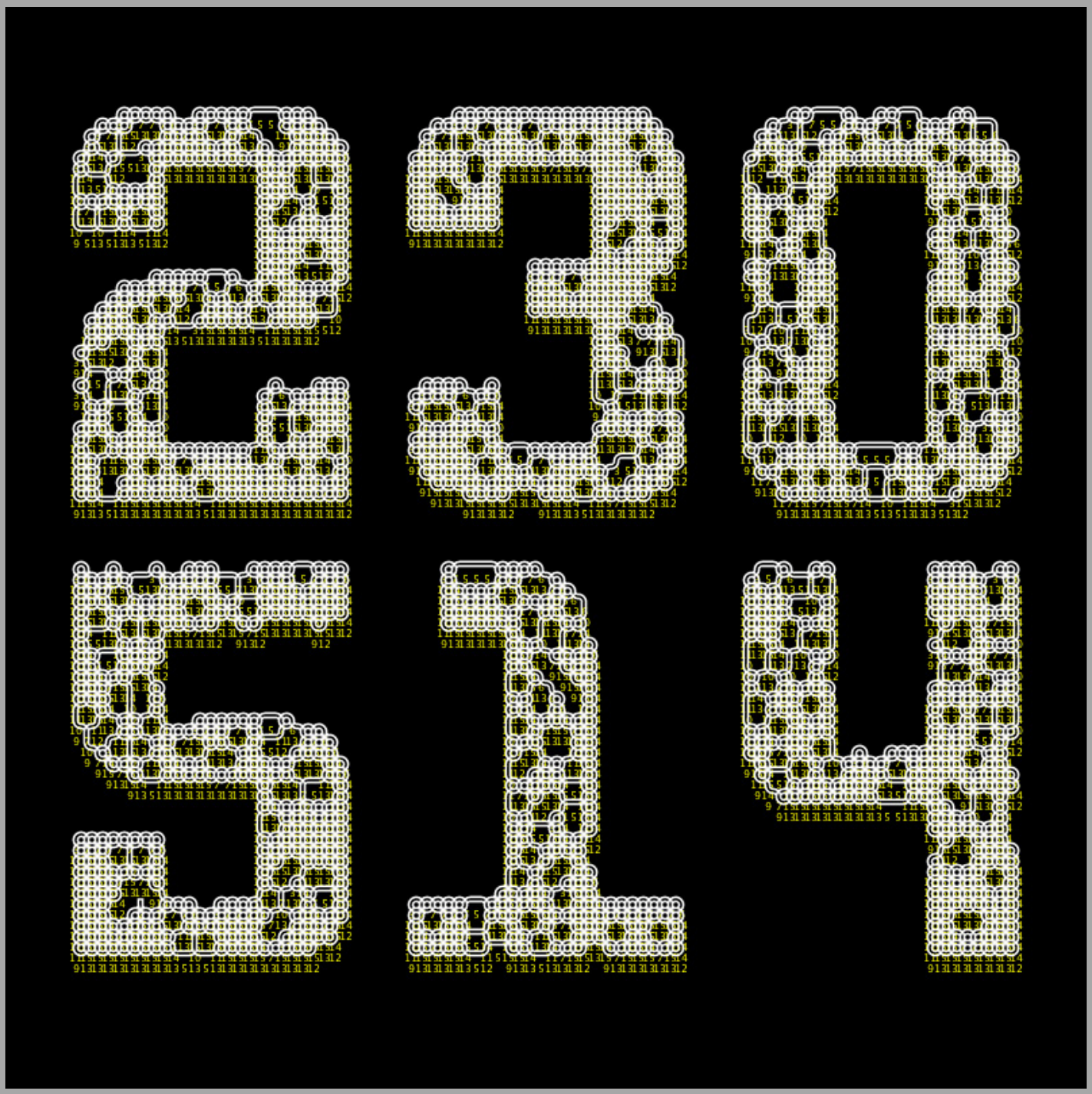
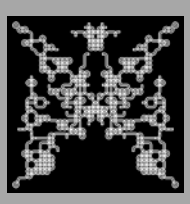
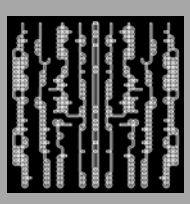
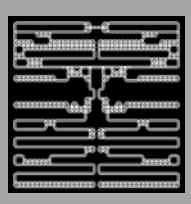
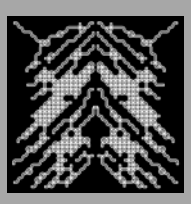
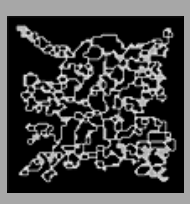
When it is possible to draw maps it's also possible to draw type. Started with the coca-cola word-mark. But it did not fit in my display. So I have only traced the word coca. And its scaled horizontally. Which is fine. Tried another one with the word DADA. Increased the module size. Used our Loftmatic typeface to trace the word. But I think that I am reaching the limits of the details because the program is not drawing smooth. You have to repeat drawing several times to get a module painted.

You don't have to make your own svg-modules to realize your own ideas. Just see how far you can go with the existing ones. In a later stage you can always replace them with your own creations.

A weaver set up his eight-loom workshop next to Botticelli's house. The noise drove the painter mad, but his process remained unheeded. Botticelli hoisted a vast rock onto the roof of his own house, balancing it in such a way that it overhung the neighbours' roof and threatened to come crashing down upon it at the slightest disturbance. The weaver, eyeing this novel sword of Damocles, came to terms.

Allessandro di Mariano Filipepi
Botticello, 1445–1510, Italian painter



Coca

BOBO

My modules and structures

MyCodeHistory: 30 May 2014

In this second part I would like to introduce my own modules in the program. I have changed everything in the program back to its original settings. I am not sure why smooth and colorMode is repeated in the draw block. Uncomment it and everything seems to work fine. Changed the global variable names and started with changing the modules set. Opened the original .svg files in Adobe Illustrator and replaced them with my own images. I started with simple dots. I also incorporated the DrawMap function of the previous program. I renamed it to DrawImage. That's a more generic name. But during that change I got the idea to make ten different images with different modules. In the previous session I used a map of the Netherlands. So it seems a good idea to make 10 different images with that map of The Netherlands.

Anyway... I have started to change the second set of modules. While I was working on the images I thought... hmm... these maps are all getting the same. It does not make much difference when you change the modules for the total image. So I thought about changing that maps-idea.

In 2009 we were invited to make a contribution for an exhibition in London's text gallery. It was an exhibition of design, typography and illustration inspired by forgotten English words. 47 Participants had been chosen from among the dictionary's lost but lovely words, with an open brief to create an original work inspired by their choice. We were asked to make a contribution with the word: 'Mansuetude'. Which means: 'gentleness' or 'mildness'. It had nothing to do with Processing. We used Apple's Motion, Final Cut Pro, Logic and NewTek's LightWave. There was no code involved.

So in addition to that I changed my original idea of maps into lost words. I first had to check how many characters could fit in the display window. Beside of that I have to make sure that these small modules can be seen. From the previous program I knew that I could display at least eight characters. I also had to find more lost words. In the end I had chosen the following words: aquabib (water-drinker), boreism (behaviour of a boring person). I have chosen these words because of their length and how interesting they looked to me. Citharize (to play the harp). Defedate (to pollute). I introduced cyan and red in the image. Ejurate (to renounce). Made an inverse blue version with it. Foppotee (simpleton). Added a double line. Did not have to be very accurate because I am going to use it on a very small-scale. Gelicide (a frost). I did not like the character spacing. I decreased it because it gave me more space to make the type bigger. But it did not work out. So I increased it again. Hiulcity (an opening or cleft). I finished with the last lost word: 'ictuate'. Which means to emphasize; to put metrical stress on. As in 'She preferred free verse over carefully-ictuated classical poetic styles.'

And this was the last program in the 'P.2 Shape' chapter of the Generative Design book. I learned a lot new things and was able to make different images. It has expanded my programming toolbox. Let's see what 'P.3 Type' brings us.

'After all, work is still the best way of escaping from life!'

Gustave Flaubert, 1821–1880, French novelist

---

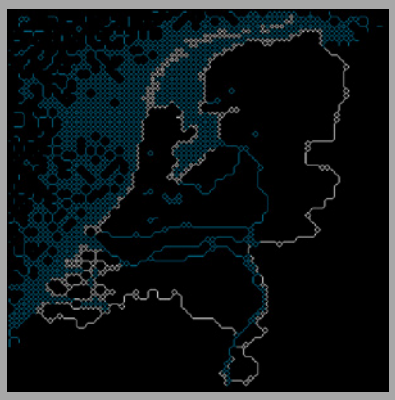
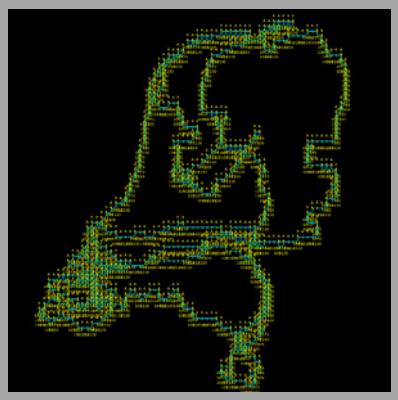
text/gallery.

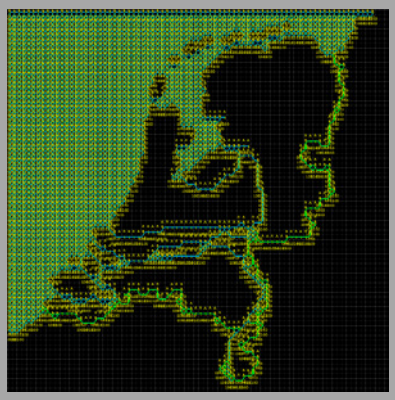
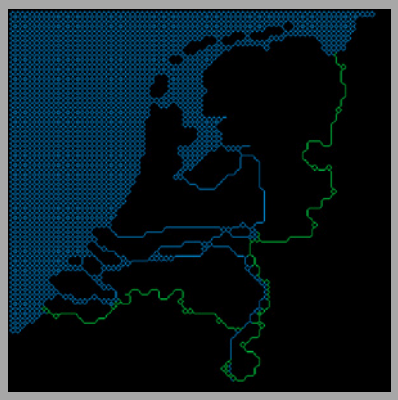
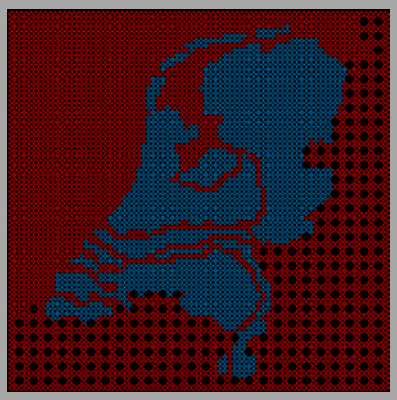
The art of lost words

Mansuetude by Loftmatic

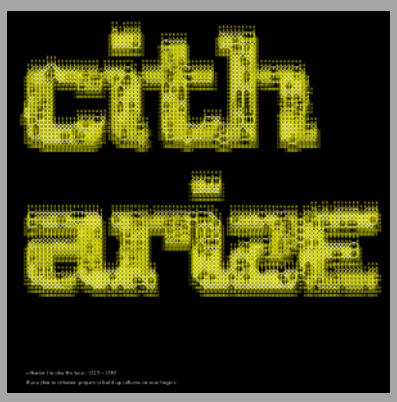
Henk Lamers and Jeanne de Bont have been at the forefront of digital type design in the Netherlands for over 20 years, for Philips Design and Philips Research. They create films, sounds, interfaces and more.

Voice over by Gavin Proctor

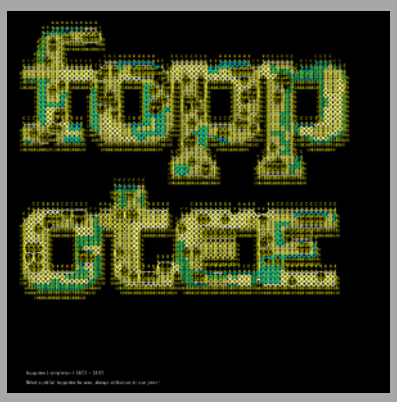



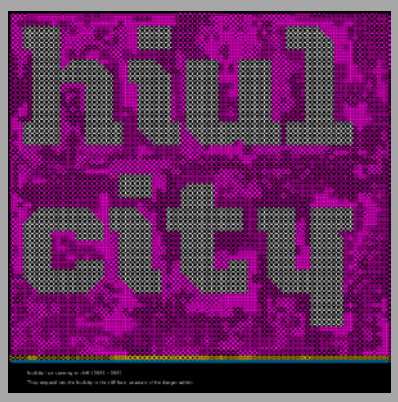
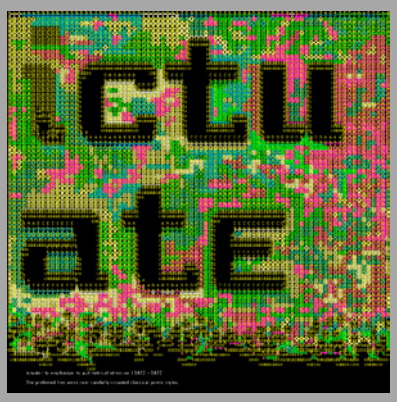




aqua
bib

bor
eism

cith
arix

dere
date

eju
rate

fopp
ctes

gek
cide

hun
oicu

ctui
ate