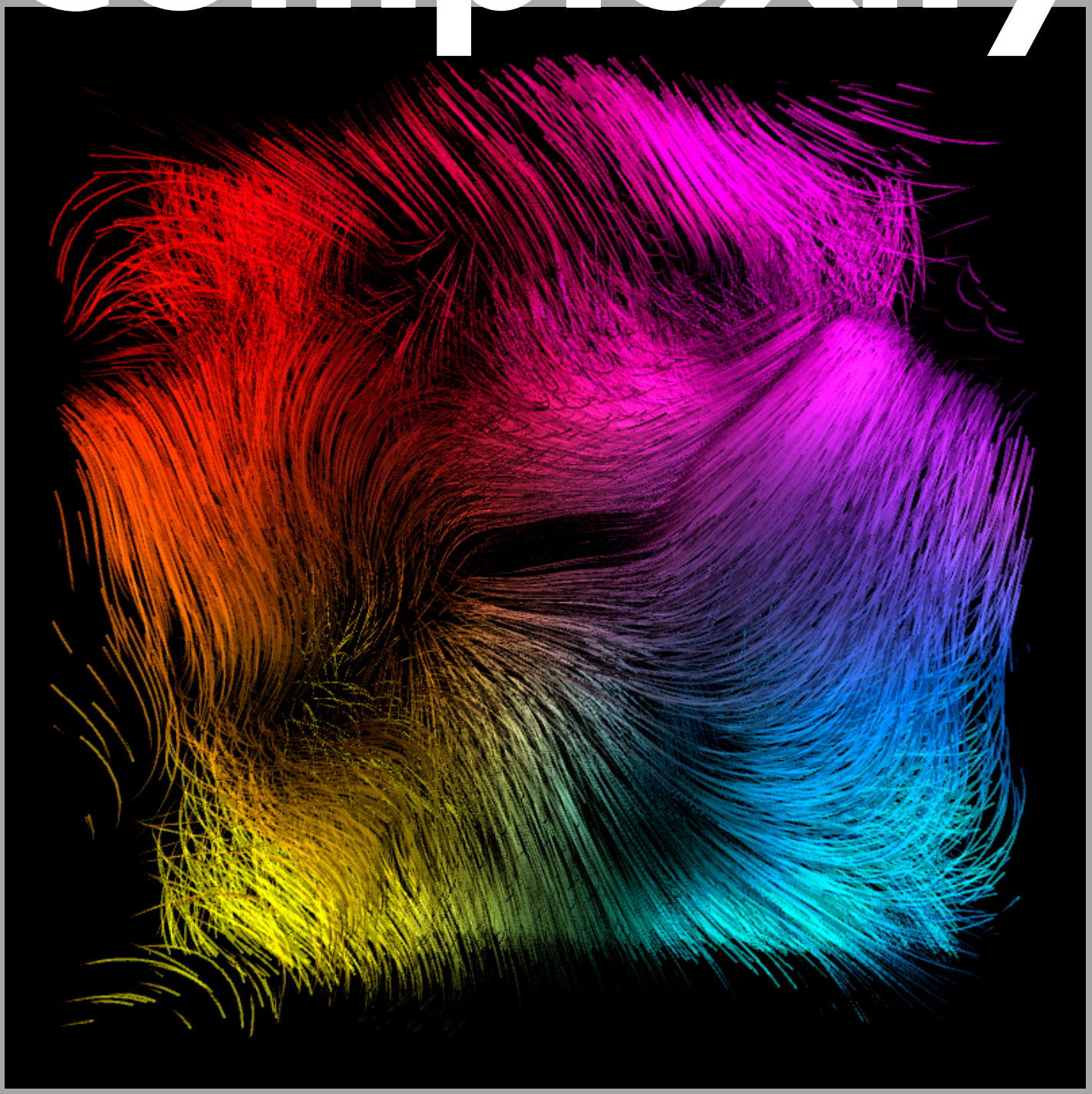
Generative Design **Variations 5**

Complexity



Henk Lamers

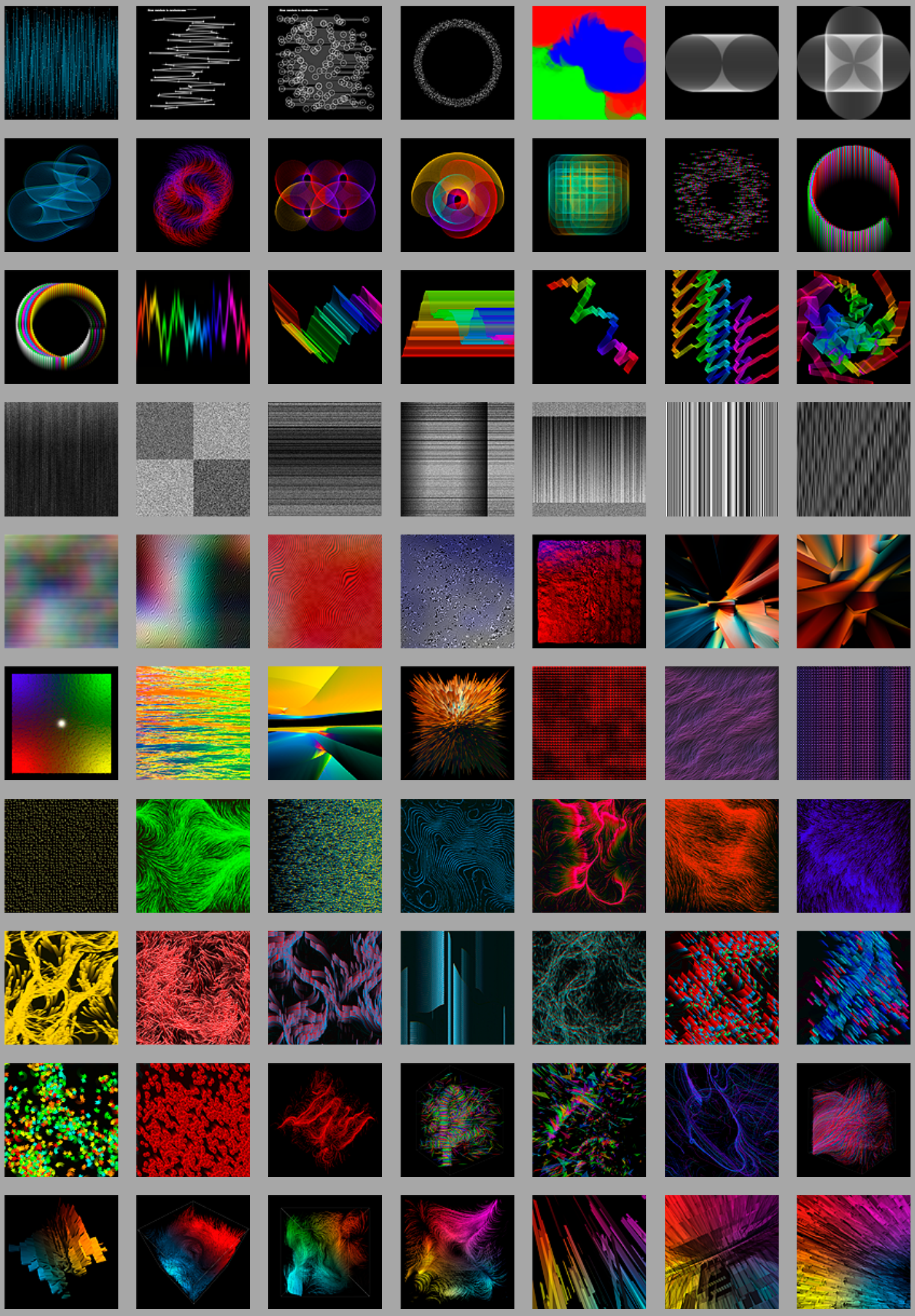
Discovering programmed design

Complexity

If there is nature involved there is complexity. Complexity has always been a part of our habitat. It is hard to explain what complexity really is. There is no general agreement let alone a definition for complexity. Therefore complexity always needs to be placed in context. For instance organized and disorganized complexity. Disorganized complexity refers to chaos, which is more like a state of disorder or confusion. Randomness also results in a kind of complexity. But it can be more organized than chaos. Huge amounts of data or information can cause immense complexity. In previous chapters I worked with simple programs but the next range of exercises are about working with complex programs. Programs which can be modified or used as a basis for new experiments.

A visitor who walked through the noble forest of Ferney made Voltaire a compliment that his trees had grown so beautifully. 'Oh,' Voltaire replied, 'they have nothing else to do,' and he walked on without a word to say.

François-Marie Arouet (Voltaire), 1694—1778, French writer, philosopher, playwright and historian.



How random is randomness

The next programs are all short introductions for the full program which will follow later. Each small program goes into a detail of the larger final program. This one will dive a bit deeper into the random function which generates random values. It returns an unexpected value within a specified range of numbers. If only one parameter is passed to the function, it will return a float between zero and the value of the high parameter. If two parameters are specified, the function will return a float with a value between the two values. Which is fine. But what about randomSeed. That sets the seed value for random. By default, random produces different results each time the program is run. But when you set the seed parameter in randomSeed to a constant it returns the same pseudo-random numbers each time the program is run. But why would you want to produce the same random numbers in the same order every time a program is run? To be honest I really don't know. To find out why that would be use-full I commented out random-Seed related program lines. But than the program keeps on running. So is randomSeed used to stop and run the program? It is used to get the same pseudo-random number over and over when the program is run. I just exaggerated the amount of lines by putting a line and a dot at every pixel in the width. The position of the begin and endpoint of that line is depending on the random factor. So you use the randomSeed one time for drawing the lines. And a bit further you use randomSeed again to generate the same random numbers to place the points at the begin and end-points of the lines.

I stopped working on this project for a week and I think that what I've done until now is not the way to go. I am going to try to make fake graphics with fake graphic representations using the random and randomSeed function. It is relatively easy to put some numbers close to the dots. I have rounded up the numbers because otherwise you get floats and those are way too long. At this moment the graphic is on its side. But maybe that is good. Let's see if I can control the data from the dots to make lines. But I noticed that the numbers are too high because they are calculated from zero, which is the left side of the display window. I added a margin so that has to be subtracted from the width. In this case its: xCeiling minus DisplayMargin that will give the correct numbers. We also need vertical axis labels. Years maybe? And every line should stand for one month. So one year should contain 12 lines. Maybe its good to make a separate function for that. I call it VerticalAxisLabels. There are now 27 years displayed. 2014 + 26 = 2040. It should start at the bottom with 2014 and count up to a certain amount of years. Fixed that by introducing the variable yearCounter. That will count down from 2040 to 2014. I also have to re-map some numbers from one range to another. At least I thought it would work like that. But I had to adjust it a bit by fiddling with the numbers. But in the end I left out the year-labels because they didn't make sense anymore.

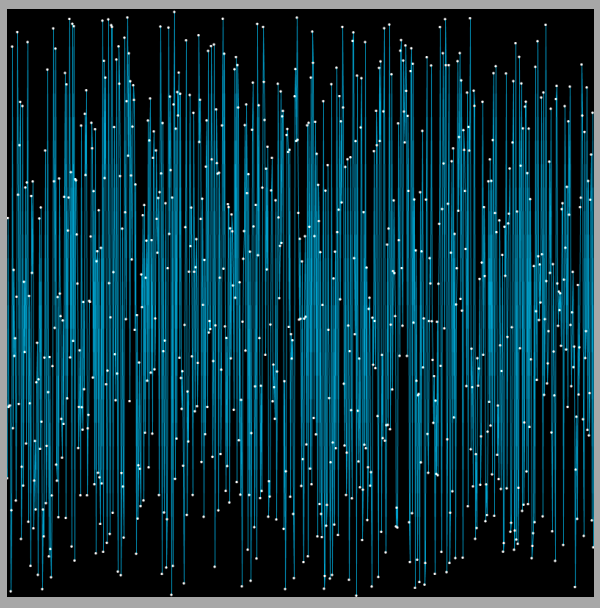
Based on this graphic I made different variations. So how does this look like when you leave out all the lines? Not boring! But you don't have an idea about the orientation. Would it be helpful if I double the amount of random numbers? That does help a little bit. I wonder if it is possible to see a pattern in the random numbers that are generated. Hence the title: 'How random is randomness'. What if I introduce the dots again? Well they do help but it does not give you more information. I have introduced circles which are on the locations where the percentage lines end. It would be interesting to see when these circles show the percentage. Zero percent is no circle. Hundred percent a full circle.

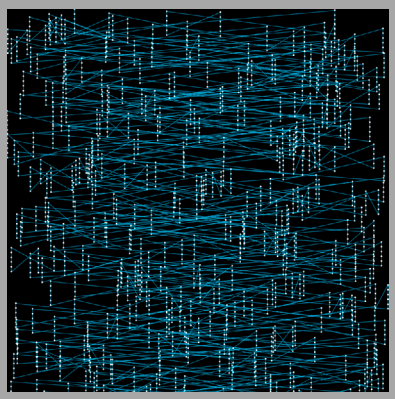
I made only ten variations. That is not much if you compare it to the amounts of images created with previous programs of the Generative Design book. But I think I now have a better idea about what the randomSeed function can do for me.

Long is the path through lessons, short and effective by examples.

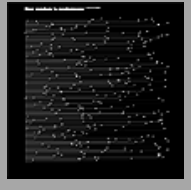
Lucius Annaeus Seneca, 4BC—65, Roman Stoic philosopher, statesman, dramatist, and humorist.

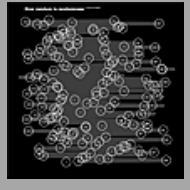

How random is randomness
In percentages

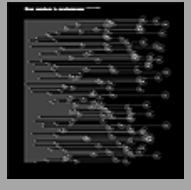


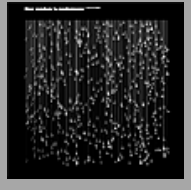



How random is randomness
In percentages









Concentration and essence

MyCodeHistory: 22 December 2014

The program reads the mouse position which is used to define some degree of randomness for a certain amount of dots. When you move to the left with your mouse the dots are completely random positioned. Moving to the right the dots form a perfect circle.
I removed all the dots and replaced them by points. There is no special reason why I did that. I have also increased the amount of points from 150 to 10.000. That made my computer behave very slow so I divided that amount in half to 5000 points. I also removed all color. But would it be possible to introduce a second circle? That is certainly possible! A third circle is also a possibility. I've now three colored circles and I have exaggerated the stroke-weight to 200 pixels. That gives interesting combinations. But it makes the program very slow and I think the images are not very subtle. I thought it might be better to remove the two other circles and color. Used rectangles of 400 x 1 pixels. A rectangle with a height of 1 pixel seems not to make any sense. Could have used lines. But for a change I kept the rectangle. It does create two circles though. Well, in fact it is still one circle but due to the overlapping of the lines its looks like there are two circles. Tried to make two vertical circles too. Had to use translate, pop- and pushMatrix to center everything in place. Seeing this image it is still remarkable that you can make circles with rectangles.

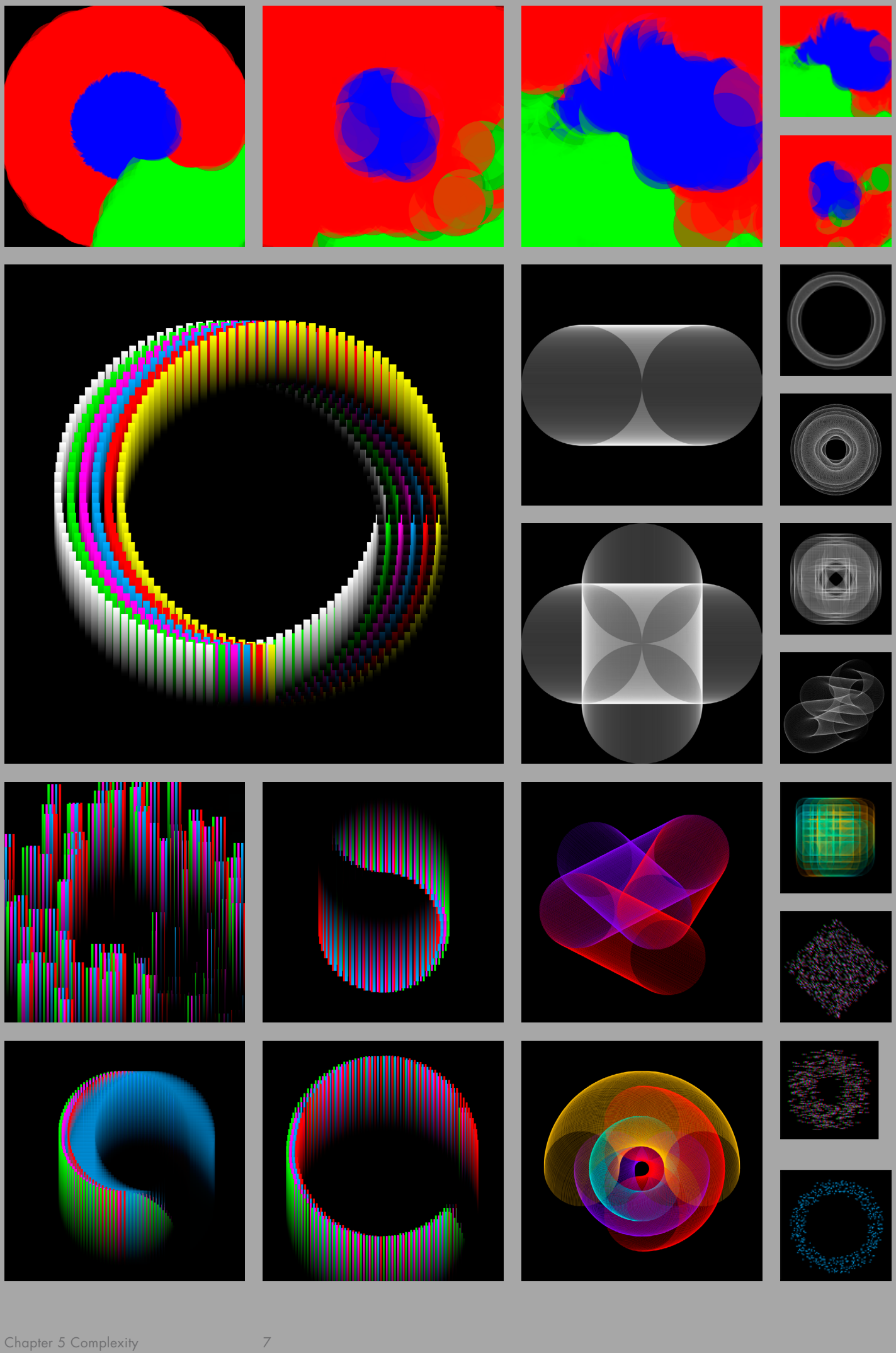
Made a random shape with bezier curves. Copied that shape three times and gave it different colors. Copied the three shapes ten pixels to the left and ten pixels up. Copied another shape ten pixels down and to the right. And than something went wrong.
I think I have overwritten the file. So I had to redo this one. But it's not a big deal because the new one is better than the overwritten one. The four bezier curves make an interesting object though. In fact the basic objects are two half circles or four quarter circles. It seems that working with circles or half circles creates very nice objects. The basic shapes of these forms are four half circles which are cut through the middle. Copied the same object in a different order. The basic object is still half a circle. You can check this by decreasing the AmountOfObjects variable to one. Still working on the same object. But repeated it under a different angle. So in total the object exists out of eight half circles. But the object's shape is getting to complex for my taste now. Four half circles shifted 45 degrees anti-clockwise. Not that clockwise or anti-clockwise would make any difference.

Almost back to the beginning of the first sketch. I replaced the circles with six horizontal lines. All ten pixels long and one pixel thick. The brightness is lowered every time a line is displayed. It's very stupid programmed because I should have used a loop. Replaced the code with a loop and I added a second loop for a similar object in a red color. Added two more objects. But to keep everything centered in the display I subtracted the positions of these two objects. And now it would be a fine moment to make a function from those loops. Put everything under a 45 degrees angle. Increased the amount of objects and scaled the total object 0.6 times. I have no idea why this object changes into a rectangle on its corner when you slide the mouse-position to the left. It's a nice one though. Expanded all the lines in the vertical direction. Removed the angle because it delivers bad image quality. In the last modification of the program I got back to the original settings and replaced all dots by gradients. Added yellow and white gradients. I think this last image is one of the best images I have made with this program.

I have gone too far away from the essence of the original program. Sometimes that happens when you are very concentrated on your work. I don't know if that is a bad thing. But I see concentration as a quality.

'The power to concentrate was the most important thing. Living without this power would be like opening one's eyes without seeing anything.'

The Elephant Vanishes, Haruki Murakami, 1949—, contemporary Japanese writer



Perlin versus Wagner

MyCodeHistory: 31 December 2014

Until now I used the random function to get random values. But to create natural effects like clouds, water, hair or smoke we can use Perlin noise. Perlin noise is a gradient noise developed by Ken Perlin in 1983 as a result of his frustration with the 'machine-like' look of computer graphics at that time. In Processing it is possible to create one, two, or three-dimensional noise with the noise function. This program comes in three variations to get used with the differences between the random and noise functions. I started with the first part where I left off in the previous program. I removed the dots and replaced them by a gradient. Switched to HSB color mode. Mirrored the gradient vertically so its smooths out to both the bottom and the top of the display window. Increased the stroke-weight to 16 pixels. And than there are interesting things going on where the gradients intersect. I also used lines to show the noise. At the right side of the display window the image is not so interesting but close to the left side it's creating nice flag-like images. Worked on that for a while. I changed the x-variable from an int into a float because that gives me better gradients. I liked the images best when you click on the left side of the display image. This has nothing to do with the contrast between random and noise anymore. It only gives interesting images. Put everything on a 45 degrees angle. Rectangles of 100 pixels long following the noise pattern. Now I would like to make more of those ribbon-like objects. The simplest thing to do is copy paste the lines of code that create those rectangles. I brought noise into the image to make it less predictable and symmetrical.

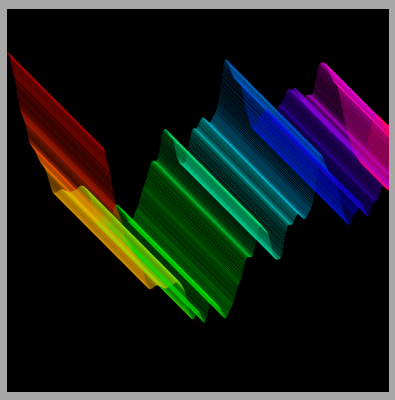
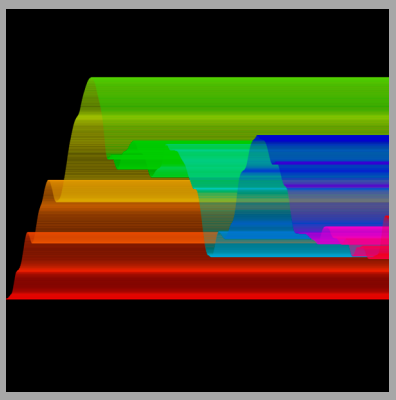
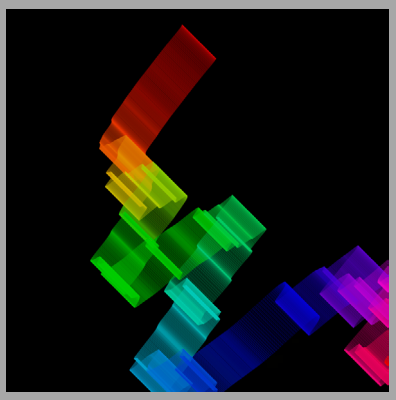
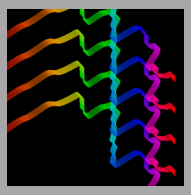
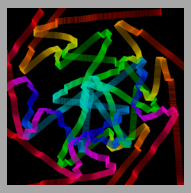
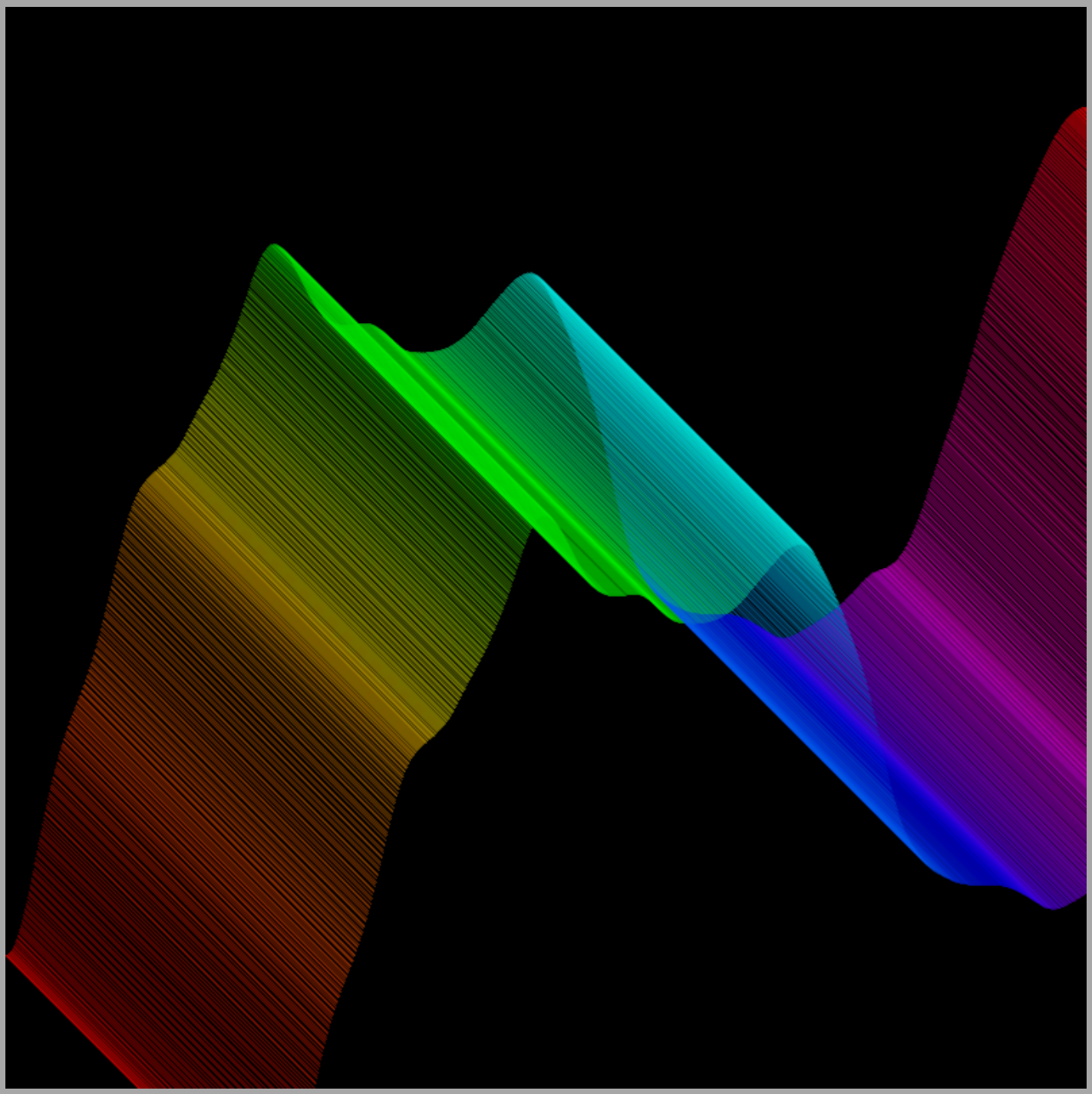
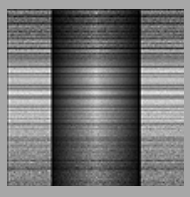
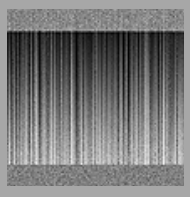
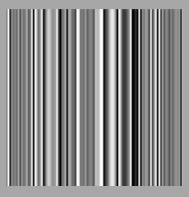
I found the second part of program ('Texture from randomness') particularly difficult to make interesting images with. It delivers a random grey value between black and white. Which is not very attractive. What can I do to make this structure more interesting? The idea is to enhance the random structure by adding an extra layer on top of the original random gray value. In fact I want the structure to be enhanced. To do that I have to add a kind of new visual value to it. I divided the display window into four squares. Each with a different interpretation of the original noise pattern.

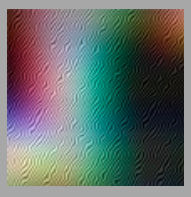
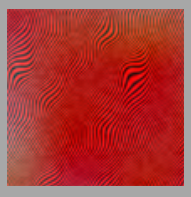
In the third version of this program ('Texture from noise') the gray value of each pixel is determined by its position in relation to the two-dimensional version of the noise-function. This sounds very complex but the result is a cloud-like texture. I tried to get away from that idea by increasing the amount of octaves from 4 to 8. As far as I could notice not much changed. So I switched off noiseY. And that creates nice curtain-like images. You could do that also in a horizontal way. Just set noiseX to zero. I have put noiseX back but increased the starting FallOff to 0.005. I also removed the mapping to noiseXRange and noiseYRange. That creates even better curtains. Changed noiseX to map to 10, 100. And map noiseY to 1, 10. That creates a rhythmic pattern. Which is the opposite of Ken Perlin's idea for generating noise. I have commented out almost all code. And I adapted some code from Andrew Glassner's book 'Processing for Visual Artists'. And that creates very subtle cloud-like images. Experimented with the color settings to get more convincing images. The rendering performance is terrible but I think that the images are very refined and subtle. Experiments often lead to images which are successful or not.

In these programs I used almost no code from the Generative Design book. Finally I ended with a program that looked like a mistake. It is a combination of some code from Andrew Glassner's book: 'Processing for Visual Artists' and some code from the first edition 'Processing' book of Casey Reas and Ben Fry.

Error is the mother of understanding.

Wilhelm Richard Wagner, 1813—1883, German composer, theatre director, polemicist, and conductor.

Eight lights

MyCodeHistory: 9 January 2015

In the previous program I was able to make cloud like textures. With this program it is possible to create landscape-like terrains. There are two new things introduced in this program: Processing 3D and Marius Watz's TileSaver class. The TileSaver class is used for rendering high-resolution images by splitting them into tiles using the viewport. I did not take the time to get more acquinted with the Tilesaver class. I have concentrated on the Processing 3D part. The first thing I did was to get all color out of this object. After that was done I needed to check what I could do with the lighting settings. I positioned the 'landscape' as a top view. Removed the ambient light and replaced it by two red and blue spotlights. I also removed all wild hills and valleys until I was left with a kind of creasy paper. I increased the tile count to 6000. And than it gives me a lot of Perlin noise. I also increased the octaves amount. And added a green light to it. I ended up with four lights. A magenta, cyan, red and orange light. Which is in fact yellow but because the lights are mixed it is rendering orange. Increased the z-scale to 2000. And tilecount to a 100. And than it renders flower or architecture-like images.

Once in a while its good to start completely from scratch. I opened the original program and I checked whether I could find a new approach to this program. A TileCount of 12000 is of course ridicules. It takes at least more than 90 seconds to render one frame. And the result is not very good. So I lowered that to 1000. It seems that the distance of the light to the surface (the landscape) does make a difference. When the lights are close to the surface it makes it harsh. Further away from the surface they give nice soft shadows. Is that true? Let's check that. All lights have now the same direction angle and concentration. Still not sure how the light direction works. Anyway I made all settings for all lights the same. Except for the positions. All lights are all located on the corners and they are on the 100-z position. Because I was not satisfied with the result I imported the light settings of the previous sketch and they worked surprisingly good. I have now a kind of sponge-like structure. And the lights are supporting that structure very well.

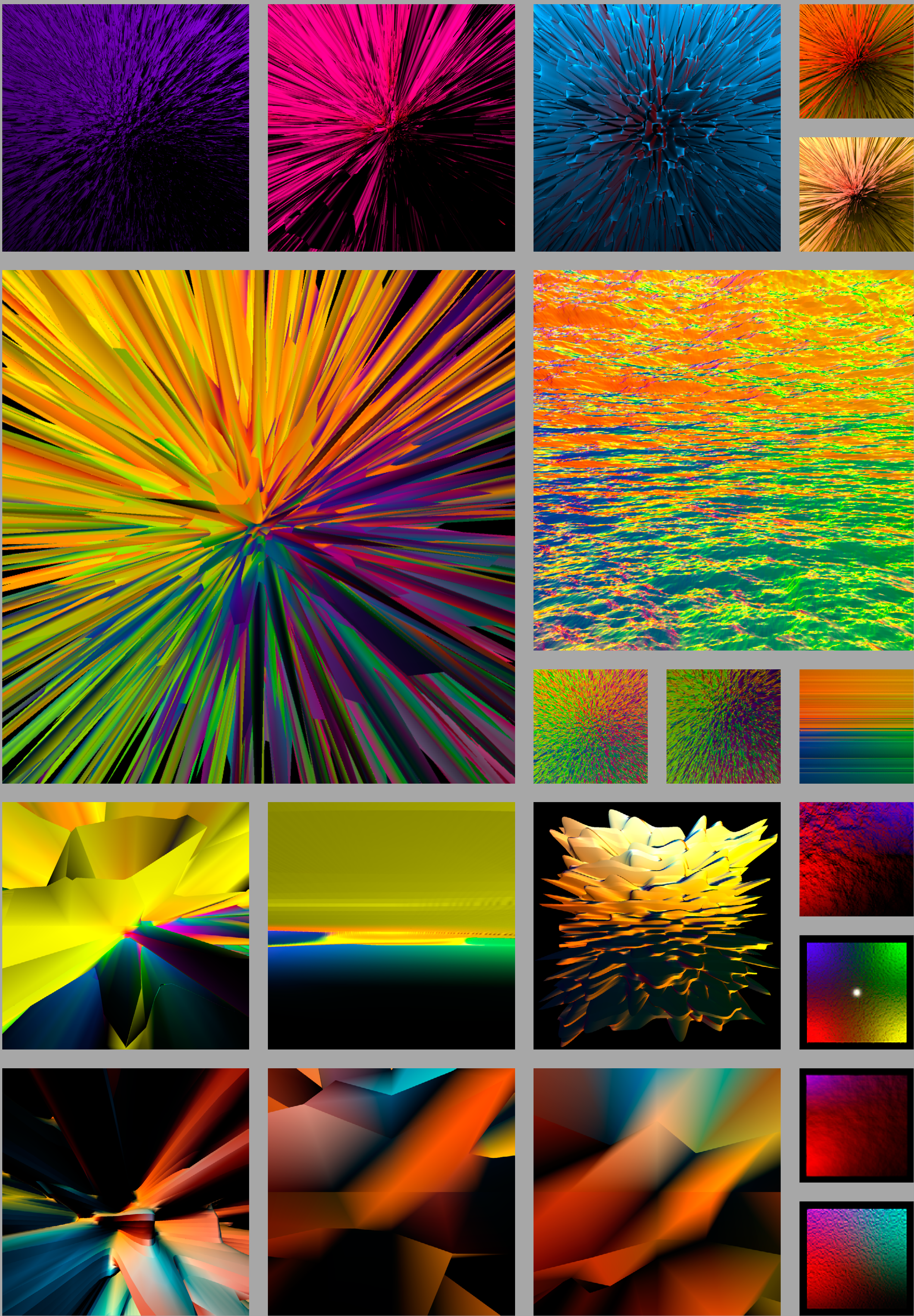
What about putting a white light in the middle? Interesting. Can I make a loop with lights? I'll give it a try. I get the following sentence in the message area: 'java.lang.RuntimeException: java.lang.RuntimeException: can only create eight lights'. That is weird. Thought it might be the loop itself that creates this problem. So I copy-pasted five lights to get ten lights and I still get the same error message. After some deskresearch I found that it is not possible to use more than eight lights. Oh… and there is one more thing: It seems that you can scroll with your mouse or pen in the display window to get other variations. Making variations has never been so easy.

More information on the 'can only create 8 lights' issue.

The original title of this program was 'Noisy landscapes'. Alas I did not create any landscape with it. Instead I looked for other things. And I have to admit that I found some useful approaches which I never had used before. You never know where you end up when programming. And I see that as a good thing.

'Uncertainty is a quality to be cherished, therefore – if not for it, who would dare to undertake anything?'

Auguste Villiers de l'Isle-Adam, 1838—1898, French symbolist writer



Controlled dynamics

MyCodeHistory: 21 January 2015

The possibilities of the noise function in Processing are endless. In this program the noise function is used to define the direction of a swarm of agents. There are four versions of this program. The last three all work with the controlp5 library. But in the first version I started with creating pattern-like images.
I lowered the TileSize because you could see the noise pattern better.
I commented out the arrows which were in the original program. I noticed that the program created arcs and that some of those arcs were not fully closed. So I increased the radians of the local angle variable to 450 degrees. Which leads to some serious overlap of the circles. I also introduced numbers to show the noiseValue. But because noiseValue produces a float I had to use round to make the number more comprehensible. I only needed to do something about the calculation. I left it by that but I continued with multiplying the length of the white line in each grid-element. What even worked better was using disableStyle and change the properties of the lines. I Changed the color and size of the lines until they had reached a tile-size of 10 pixels. And those settings delivered me very natural hairy qualities. Made some variations with cross-, rectangle-, and star-like images.

From here on all upcoming programs worked with the controlp5 library. Controlp5 is a GUI library written for Processing by Andreas Schlegel. I did not work much with classes and libraries before. So this would be a good opportunity to start working with them. I changed the names of the global variables. The result was that the program didn't work anymore.
I got a stream of particles on the screen but they did not react on the controlp5 sliders when I changed them. I found out that the global variable names created the problem. So I replaced those in the controlp5 library tab.
Another thing was that saveFrame did not properly work. It gave me a gray image. This was because it creates a png-file with transparency. I solved the problem by changing the png-file.
I studied the controlp5 library for a while. And changed the line function into a point. That gave me the possibility to use the full range of strokeweight's without losing the speed of the vectors. To change the final images I added two different colors, blue and yellow for each mode as a start. Made a function called colorCycler which cycles very slowly through the HSB colormode. The program created very interesting images which change from growing cornfields through fields of concentrated ellipses and great waves.

The complexity and reduced chaos of the images can be increased with three-dimensional noise. Until now I thought that the P2D-renderer was meant for 2D space and the P3D-renderer for 3D space. But it seems that there are four render modes: the default renderer, P2D, P3D, and PDF. The default render mode delivers 'slow' but very accurate 2D images. The P2D render mode is using OpenGL and is faster but delivers less accurate 2D images. P3D render mode also uses OpenGL and delivers very good 3D-image quality. The PDF render mode is used for creating PDF output. Meanwhile due to the changes in Processing 2.0, P2D and P3D have been replaced with variants of the OpenGL renderer because the Processing team felt that OpenGL rendering is probably the future for most Processing work. Knowing that I noticed that the space bar functionality for triggering the noise seed had disappeared in the last three programs. I found that functionality quite useful so I have put that back again. The program creates wavy patterns but I also was able to create flag-like objects assembled by red, blue and green lines.

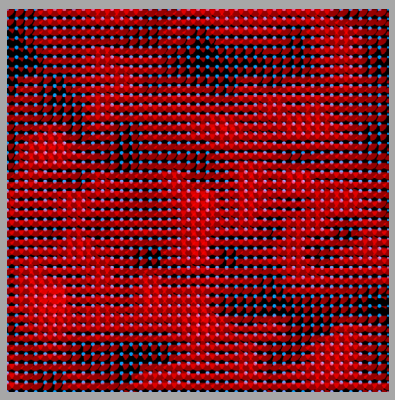
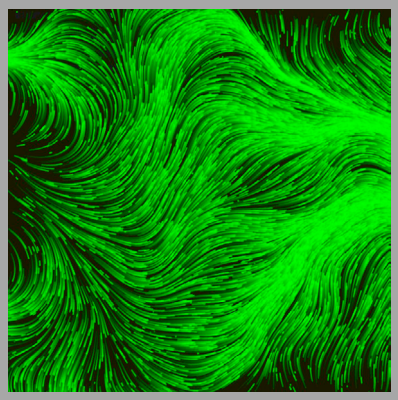
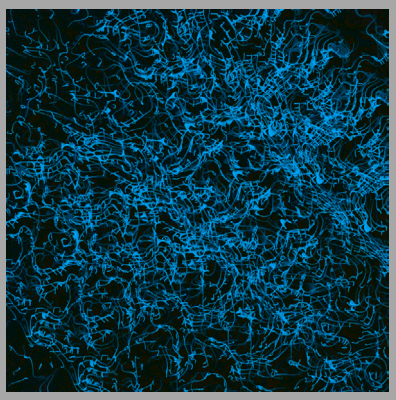
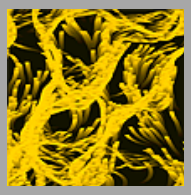
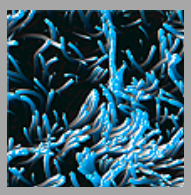
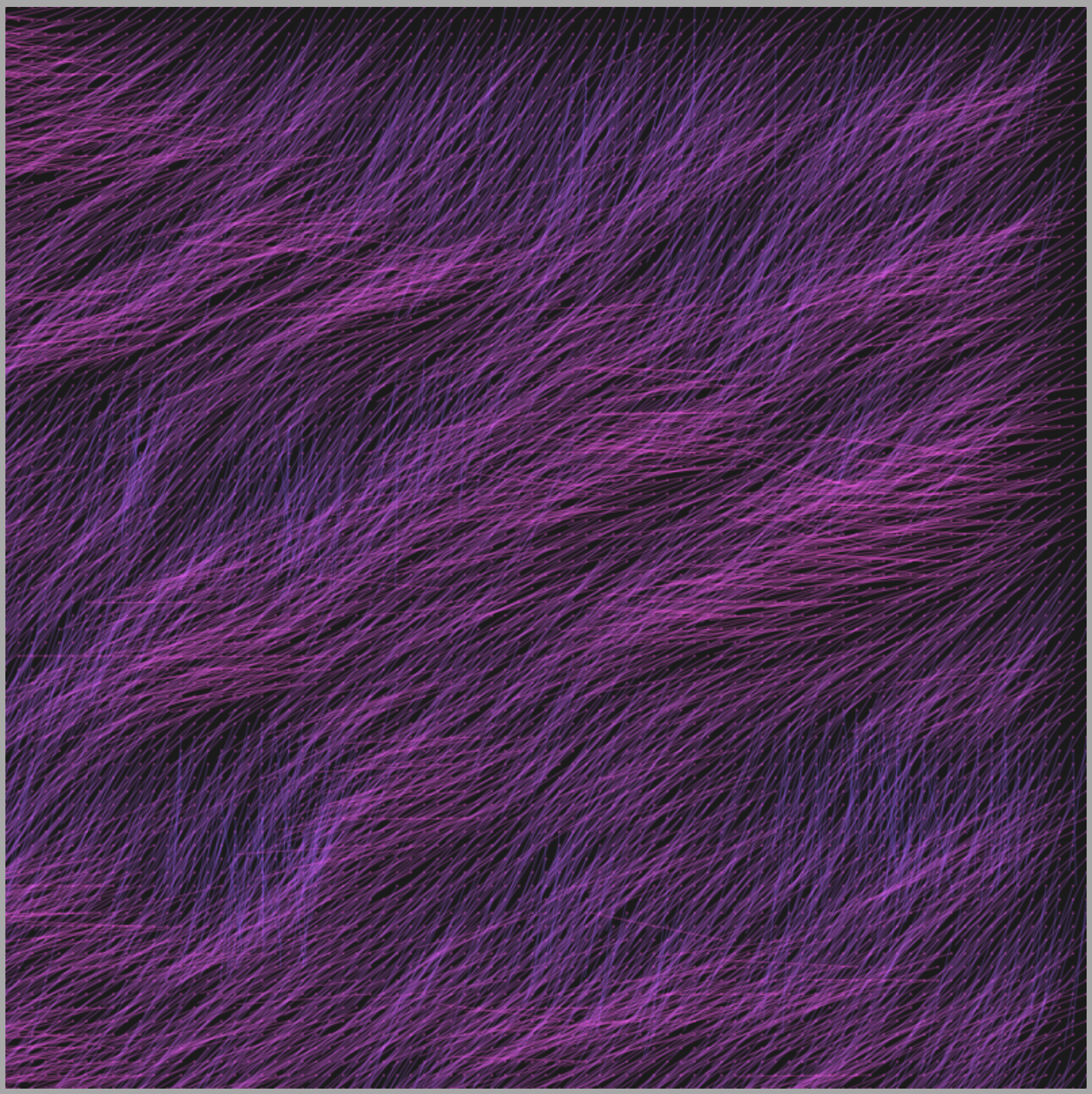
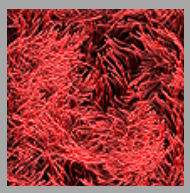
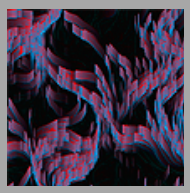
There is one more bonus program available which is not being discussed in the Generative Design book. But I've made some variations with it anyway. Did some experiments with rectangles, curves, waves and paper-like objects.

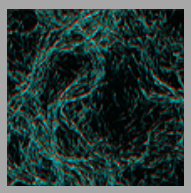
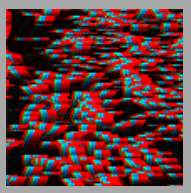
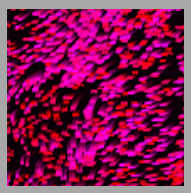
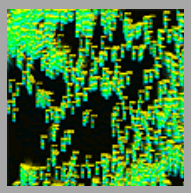
And I used text to create a texture. But because I thought it was not very functional to use 'just' text I used the year, month, day, hour minute and second function as input. Something strange happened here. I defined the time and date variables at the top of the program. But when I run the program it did not refresh the seconds. After a while it was clear to me that you should initialize the date and time functions in the draw block of Processing. Otherwise the date and time functions will not update.
I replaced the numbers with 800 white swans. Replaced those on its turn with arrows. And I finally ended with colorized circles which might look more like tubes but they are just circles.

The use of libraries can enormously increase the possibilities to create images. But it depends on the flexibility of the library how original the images can be. In this program the controlp5 library is a very flexible one. There are enough possible ways to create images which do not look like they are made with a kind of filter which anyone could switch on or off.

'Don't try to be original, just try to be good.'

Peretz Rosenbaum (Paul Rand), 1914—1996, American art director and graphic designer.

PuhVectors in space

MyCodeHistory: 30 January 2015

The main difference with the previous program is that this program uses a three-dimensional space in which the agents can float reasonably free. The program has two versions. But first a question: hint (ENABLE_DEPTH_TEST); what does it do? As far as I could trace the function it allows you to combine 2d and 3d drawing. But I am not sure about that. Anyway… I am using it and maybe I will find out what it does later. To begin with I think I am going to need an on/off-key for the transparent blue cubical box which is located around the agents. Sometimes I do not want to see that box so I defined the variable ShowBox. Pressing the 'b' or 'B' key toggles between ShowBox and do not ShowBox. About Marius Watz's TileSaver class: What does the global variable QualityFactor 3 do? My display window size is 800 x 800. The QualityFactor is set on 3 as a default. And 3 x 800 = 2400. So maybe if I make the QualityFactor just 1 it spits out an 800 x 800 png. And that is the case! So QualityFactor = the size of the image. As a result I have renamed that global variable to ImageSize. Further I would like to bring in some color. Where can I do that? I saw that color is used in the next program. And the color is generated in the Agent class. I copied that piece of code and adapted the colors to my needs. Also added transparency. Strange enough the setting does not give me an idea of a 3d environment. Are those lights working? Even when I switch them off the scene seems to be lighted. And because I don't know why they are on I turn them off (like we all do in our energy saving habitat). Continuing with replacing PVectors. PuhVector! As Daniel Shiffman mentions it in the accompanying videos of 'The nature of Code'. A PVector is a way to store two values. Or in this case three values because I'am working in 3d. Made every agent-movement very slow. It is changing lines which render for 3000 pixels. So they render also outside the box. Eventually they stop in the box. If you put the box upside down and looking into it you get a total different view of the same object. I let it render for fifteen minutes while I'm cleaning our espresso machine.

In the second part of the PuhVectors in space I was very annoyed by the fact that I could not find a way to control the lights in the previous program. The lights just didn't have any effect. So I tried it once more. Switched off all lights. And created just one spotlight. On a certain moment I saw that a tiny piece of light was hitting a small part of some ribbon. So I tried to manipulate the spotlight in such a way that it had more effect on every ribbon. And… success! I could uncomment the coloring of the ribbons because to get an ideal coloring by light I have to color all ribbons white. I have the impression that the lights are not working on lines. In the agent's draw block I have changed drawMeshRibon for drawLineRibbon. And I get white lines. Which is the real color of the line. Another thing is that the spotlights do not give enough light. So I used an old trick which I used when I was using NewTek's Lightwave 3D program. I duplicated the spotlights exactly at the same locations. And that helped. The brightness of all the spotlights changed to twice as high. I continued with putting four lights on each corner-point of the cube. And I doubled those lights to increase the brightness. By the way… I think that Marius Watz's TileSaver class did a great job. Using it you can always make very large images without losing lots of image quality.

More about: hint (ENABLE_DEPTH_TEST);

Three-dimensional space is certainly adding a lot more possibilities to make interesting graphics. But it is also much more difficult to control in Processing than in specialized 3D software. Sometimes it is difficult to believe that three-dimensional space is just an illusion.

The imagination is a place where hypotheses and conditionals rule, and where part of the fun, and most of the point, lies in saying the unsayable in order to test the truths of what's most often said.

Adam Gopnik, 1956—, Canadian American writer, essayist and commentator.