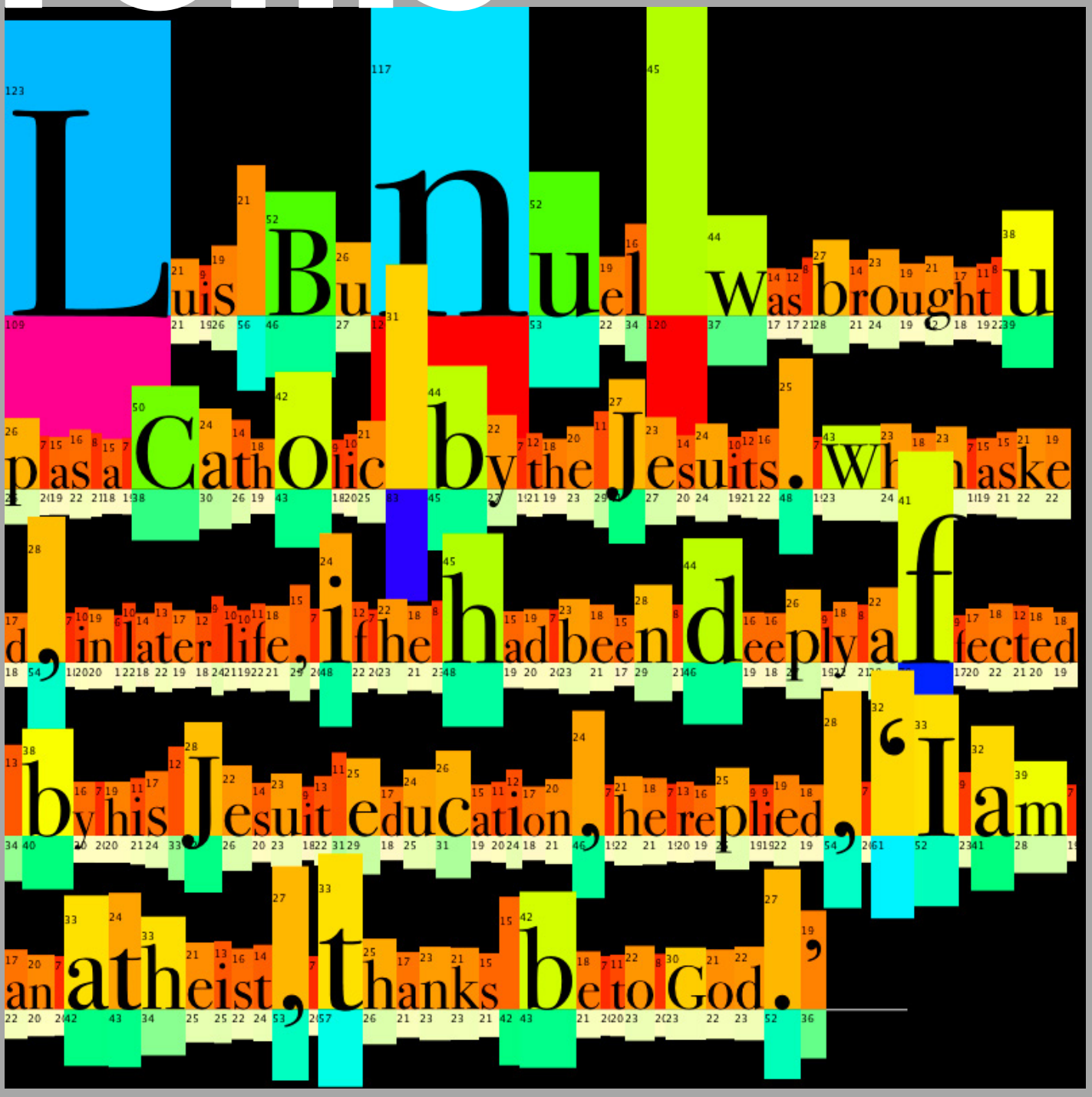
Fonts

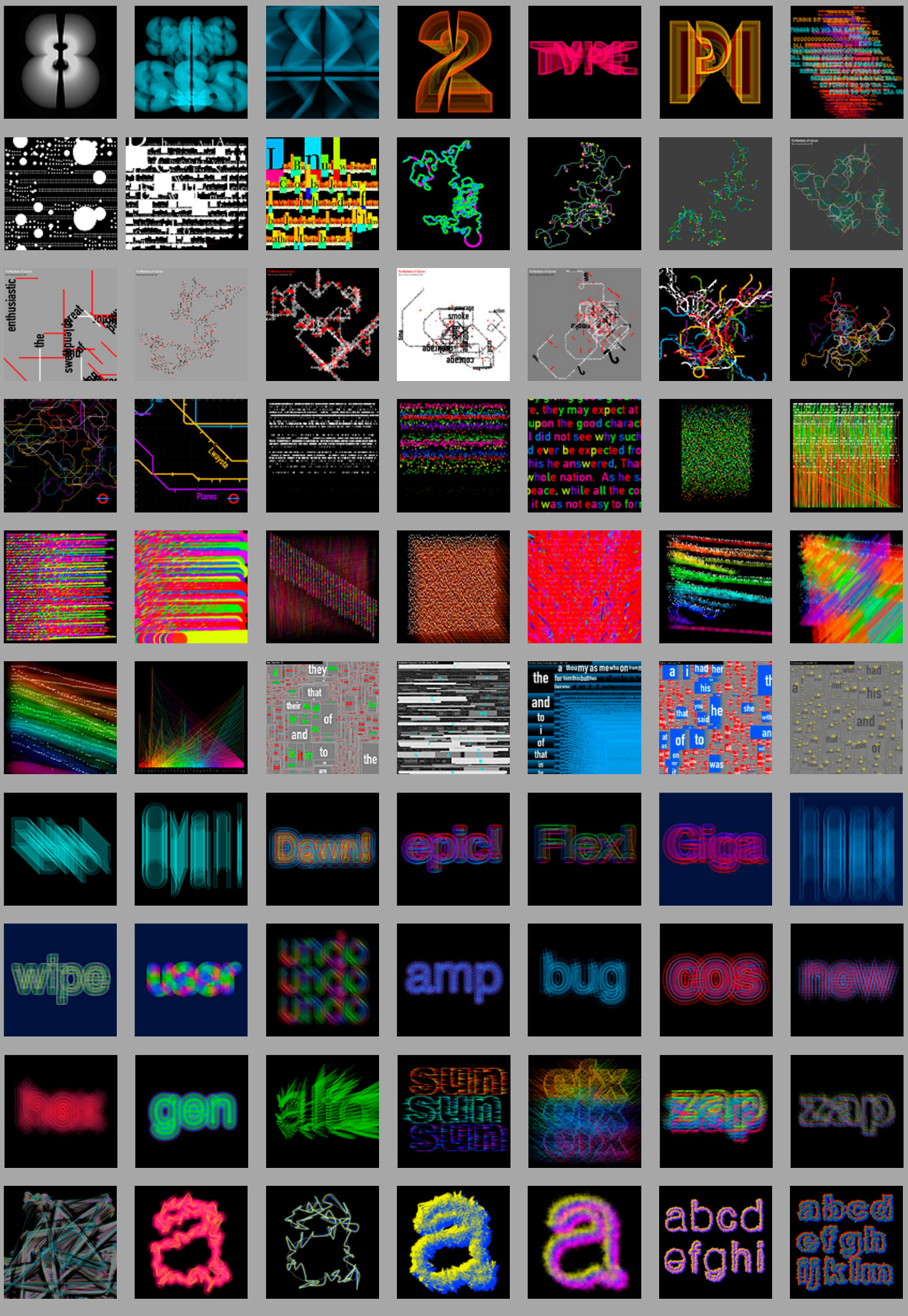

Luis Bunuel was brought up
a Catholic by the Jesuits. When asked
if he had been deeply affected
by his Jesuit education, he replied, 'I am
an atheist, thanks be to God.'

Fonts

I made my first font 45 years ago. There was a competition organized by the company Letraset. They asked to submit a self-designed alphabet. Every one of my classmate's made one character based on some drafts I had made. If you were to win the competition the alphabet would be commercially sold as a dry transfer. Dry transfers enabled commercial artists to produce high quality headlines by transferring pre-printed letters directly onto artwork. When computerization made its impact on the design world in the mid 1980s, Letraset converted their extensive typeface range into digital format. And Letraset's dry transfer lettering system lives on today and they still invite the submission of designs from any source.

Today there are about as many different typefaces as there are idiots. Typefaces have had their day. For spelling, linguistics and that kind of exaggerated educational hassle there is no place in our world anymore. The only way to reform the modern letter is to abolish it.

An Essay on Typography, 1931, Eric Gill, 1882–1940


TYPE
enthusiastic
the
Planes
they
that
and
to
the
Dawn!
epic!
Flex!
Giga
wipe
amp
bug
cos
new
hex
gen
sun
sun
sun
zap
zap
abcd
efghi
abcd
efgh
ijklm

Hello character

MyCodeHistory: 8 June 2014

This program is based on the same principles as the 'Hello color' and 'Hello shape' programs in the Generative Design book. Horizontal mouse movements control the size of the characters. Vertical mouse movements control the characters going up or down and leaving a trail. I thought the program was running too fast. So I added a frame rate of 1. Also added a transparency of 4. But that gave me no good results. What about the font itself? FuturaStd-Light seems to be a good candidate. Although the Light extension doesn't give me the Light version in Processing. I had this problem before so I will not get into that again. To avoid all these font related problems I decided to use one of our own typefaces (FIOTEXTRegular).

Instead of displaying one character I can repeat characters. The only change I have to make is to type them in the program. I also tried to place colors under the number keys but that is obviously not a good idea because these numbers will sometimes appear in the display-window. Maybe I can switch them off. FrameRate (1.5). I did not know that frame rate could work with floats. With opaque characters the screen is easily filled but by using outline characters you might get interesting other results. And you don't need transparency. So I have thrown some code away. Another part is re-used. I have now the same version as the original program was. Except for one thing. It is an outline character. Made a few versions.

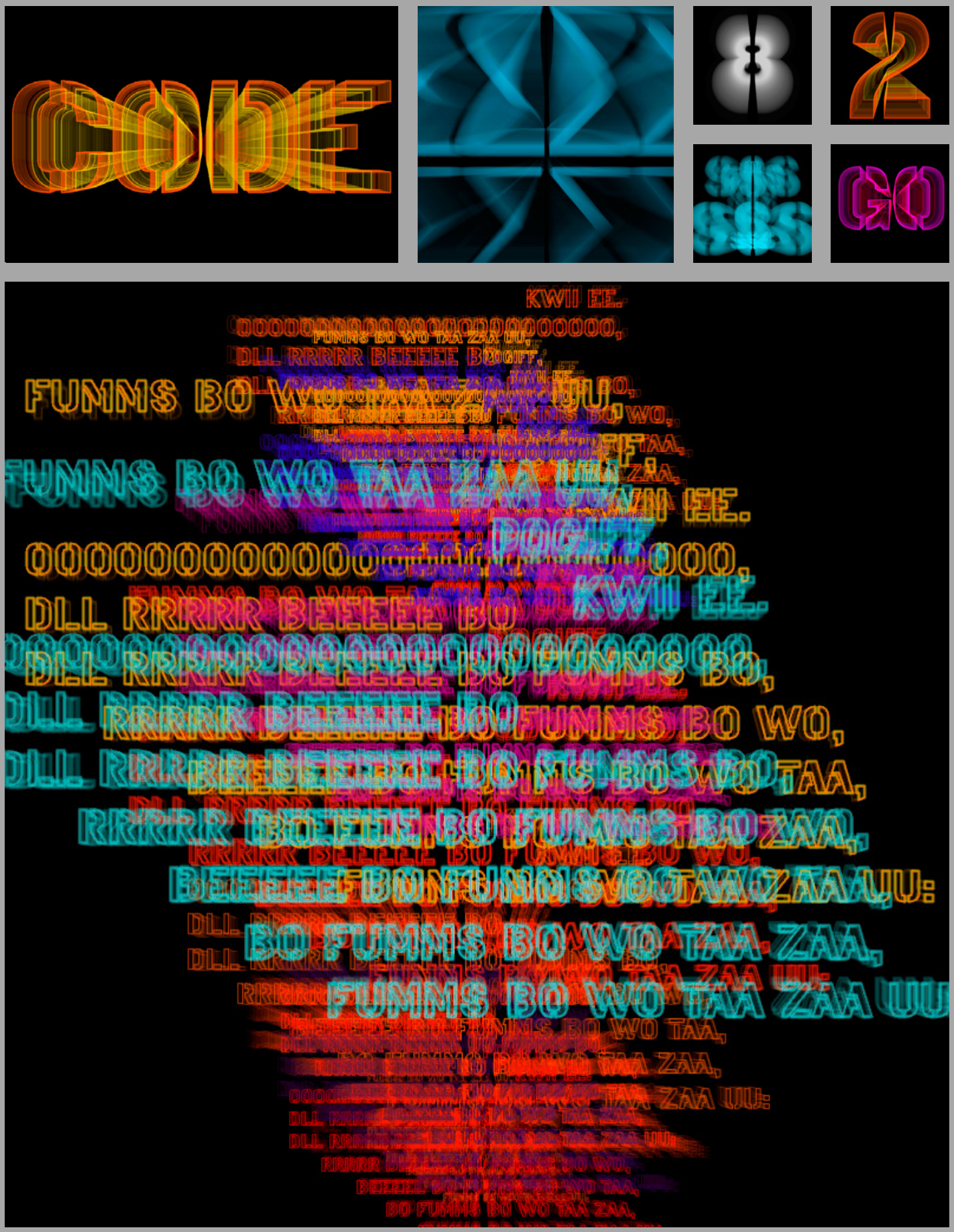
Color is chosen randomly. Just hit the space bar. To get straight lines I taped a Linex A 6036, 60 Degree Set Square on my Wacom tablet. If you draw lines by hand you easily get some unwanted curves or disturbances in the image. I think the colors are way too random. To make it more organized I will put them under the keys 1 to 5. Worked with the number 2 for a while now. Let's make words of two characters for instance. Are there any words of two letters? Yes, there are! I found them at the San José Scrabble Club. Includes all two letter words now acceptable in US Club and Tournament Play as of 2009. Used: AX, GO and PI. Removed my Linex A 6036 and replaced it with a much thinner Faber-Castell Typometer. In that way I can control my pen more accurate. I did not use that typometer for thirty years. But it still works. Did some tests with words containing three letters. While working I was watching the tennis games at Roland Garros. Thought that the word: ACE would come in handy. EON and SKY are also fine to use. Four letter words. CODE, TYPE and DATA were pretty obvious words to use. Experimented with bright background colors but I did not like the results.

For the last variations I would like to try how much type/characters you can fit into the display area without the words getting unreadable. I start with five characters in width and five in height. ADDAX, ADMIX, ATAXY, AXIAL and AXION. Six words in height and six characters in width. BABOOL, BAFFLE, BAGUET, BAFFED, BABKAS, BAGNIO. Fine. I got another idea… lets take some poetry from Kurt Schwitters's 'Ursonate'. Make a composition with it and use he program to create an image. Hmmm… it seems that it takes some time to generate the image. But it doesn't look bad. Its better than the separate words. But I think it needs some fine-tuning. I make one version in which the composition of the svg-file looks more like the original layout which Kurt Schwitters designed.

Fonts are always interesting. And fonts are not a totally new subject for us. We used to work in the type design-related industry for more then 10 years. Jeanne de Bont and I also designed our own typefaces. We didn't design many fonts and they were never commercially available. We mainly used them in our own publications, documentaries and animation films. We used fonts always to support the content of our projects.

…It has to do with mood-setting before the message is delivered. Typography is a hidden tool of manipulation within society. All schools should be teaching typography; we should be fundamentally aware of how typographic language is forming out ass-holes.

Neville Brody, English graphic designer, typographer, 1957—


CODE
GO
KWII EE.
FUMMS BO WO TAA ZAA UU:
BO FUMMS BO WO TAA ZAA,
FUMMS BO WO TAA ZAA UU

GDV 3.1.1 A text game

MyCodeHistory: 14 June 2014

I play squash tree times a week. But I don't like to play computer-games. And this second program is a kind of game. The vertical mouse position is responsible for the leading. The time it takes to enter a character determines its point-size. That is in short what its all about. My goal is to change the program in a way that it leads to interesting typography. In fact that is all-ready the case with this program but maybe I can change it in a way that it makes other use of this time-based writing tool. I started with changing the names of the global variables. Changed pMillis in PastMillis. Hope that is the right interpretation. Could have used the word 'leading' too. But LineSpacing is better to use with camel-cased text. Changed spacing into LetterSpacing. Changed the display size to 800 x 800 pixels. Oh that is something new: frame.setResizable (true); While the program runs you can adjust the size of the display window. Great! Removed the 'Type slow and fast!' text because if you know that once you know it for every time the program runs. Changed the size and color of the cursor. I think it gets too much attention. Made some images using only punctuation: degree sign, colon, solidus, section sign and the asterisk.

Tried to do something with rotation. Most of the time text is typed on a straight line. But what if this lines rotates? And what if it's rotating faster when more letters are added? That seemed a good idea in theory but not when I brought it into practice. What about giving the letters a kind of accent by displaying a rectangle below them. Position it under the displayed letter. In the first version the rectangle was white. That means it overlapped everything of the typed character that was displayed under its baseline. Filled the rectangle with a grey color. Made a few versions with some entertaining anecdotes.

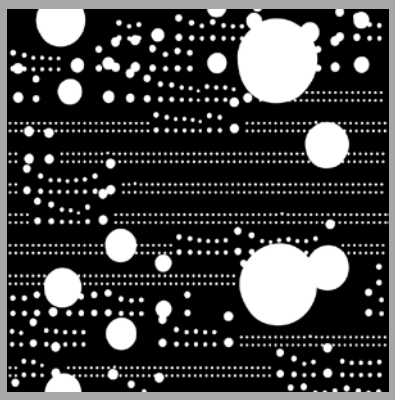
Maybe it's a good idea to introduce a sort off debug mode like we had in one of the last programs of the geometry chapter. I would like to know what point size each character has. So it's not a debug mode but it is some extra info about the typed character. If I use fontSize for returning its info I get floats. So I used round to convert it into int. Made a few versions also with some anecdotes. Showing a blue line indicating the character width. Although this is a bit silly because the character width is the width of the character itself. But it's giving me an interesting image and that counts too.

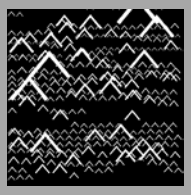
The program shows a blue line indicating the character width. Showing a yellow line indicating the point size. Changing the yellow line into a rectangle again. Changed the blue line into a blue rectangle. Made a few variations with anecdotes. Trying to connect the point size to a color. Used that color to fill the lower rectangle. Using the width of the character for coloring the upper rectangles. Changed the color mode to HSB because this is easier to understand for humans. And again I used some anecdotes for the text.

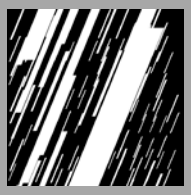
When you think about changing a program it may lead to an unexpected result. You had the idea in your head. But sometimes when its on the screen it doesn't look good. You could throw everything away. And start again. But changing a bad idea can also lead to better idea.

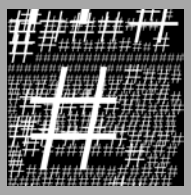
You cannot understand typography and typefaces without knowledge and you can't keep that knowledge only for yourself. Type design is a cultural act, not just a few lines of data in the corner of a hard-disk.

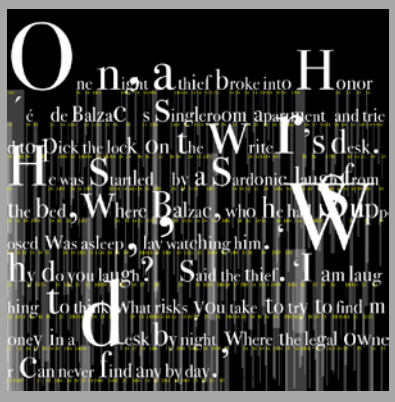
Jean-François Porchez, 1964–, French type designer

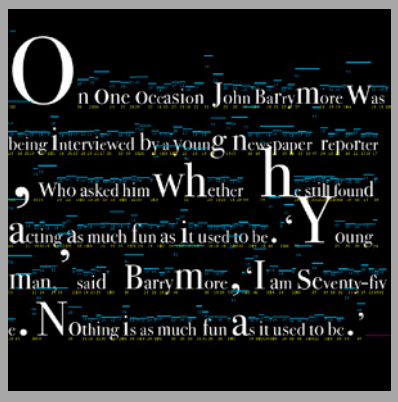


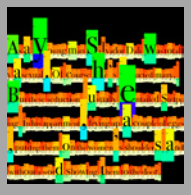












Text as lines

MyCodeHistory: 29 June 2014

Every character input is translated into a visual rule by this program. The program also decides what will be drawn and how position and size are altered. Some of the characters can be replaced by svg-modules. The previous program used time-based text writing but this program seems to be a different ballgame. Just to study the program I commented out all cases in the set of rules except for the first case which is the word-space. When I run the program there is no text visible, not even when typing. I get a white blinking cursor which rotates in a blue ellipse. So I uncommented the second if statement which triggers ShapeSpace2. I see a white blinking cursor followed by green and blue arcs when I keep the space bar pressed. Changed the color of all other modules. Even without text this is an interesting tool to make textures.

The original text of the program is using lyrics from the German electronic music band Kraftwerk. I replaced the text with the 'Manifesto of Futurism' by Filippo Tommaso Emilio Marinetti. I know that Marinetti was an ardent supporter of fascism. I have chosen this text mainly because of its length not for its content. The next step is to replace the color arcs with simple dots but keeping its functionality. I think these arcs are taking up to much attention. Step by step replacing all modules by colored dots. Some dots are positioned mysteriously. In the end I replaced all dots by rectangles. What will happen when I comment out all rotation? Then I get all text of the Manifesto as one long line of text. I restricted the rotation of the lines by 45°, 90°, 135° and 180° angles. This will give it a less random feeling. Maybe it's a good idea to make the often used characters small and the less often characters large. Or just the opposite. In the last version I used a white rectangle for the word-space. Space2 has one white and one red rectangle. The comma uses two white and one red rectangle. The period has two white and two red rectangles. The exclamation mark has three white and two red rectangles. The question mark has three white and three red rectangles. For return I used four white rectangles and three red ones.

At that moment it bothered me that I didn't get away with that 'Underground Map' idea. Mainly because its interface is difficult to handle. To create a map with this program you have to type text and think about the character-code at the same time. So I decided to import the text as a string. When you load the manifesto text the program interprets it as a chaotic cluster of text-lines. But I found the image not complex enough so I have put the Manifesto text in a for loop. It's going to be displayed three times and then it looks fine. There are mistakes in it but I like the look of it and it gives the impression of a map. Did not mind to much about the positioning of the text and all stations and icons are switched off. I have picked up 'Mr Beck's Underground Map' from our bookshelf. Maybe use Edward Johnston's 'Underground Railways Sans' typeface? Run a river through it? Adding a London Transport sign? Or a Generative Transport sign?

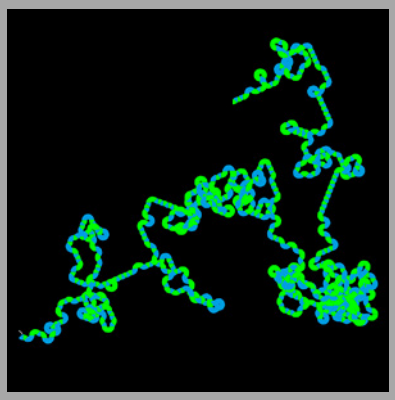
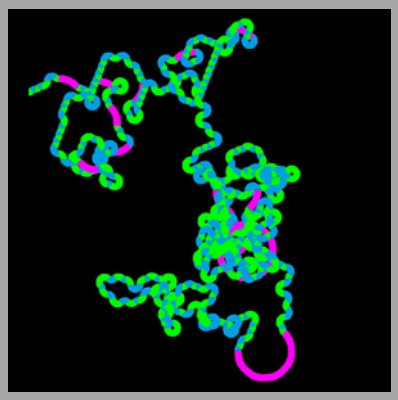
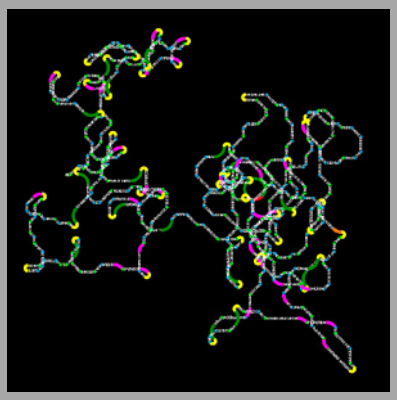
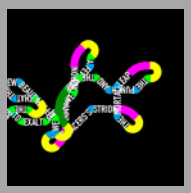
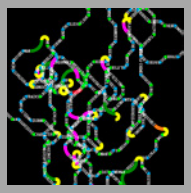
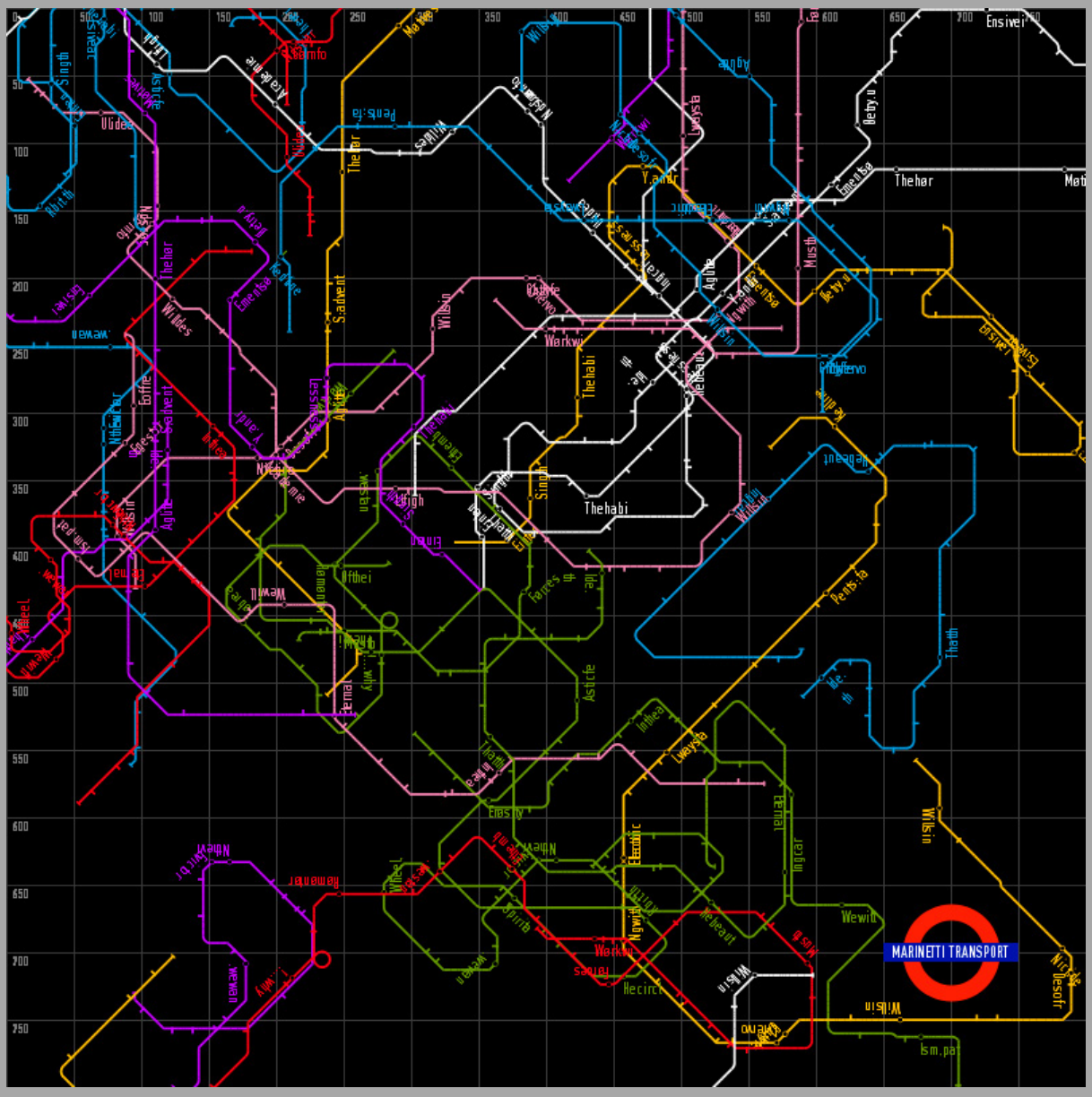
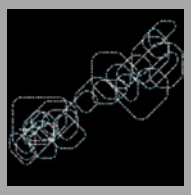
I increased the random function which is responsible for a new 'railroad' line when a return has been given from 300 to 3000. This makes the map a bit more open and less chaotic. It solves also most of the overwriting of station-names. Working with this map I find that Marinetti's poem works rather well. The text is about cars, railways, air-planes, boats and factories. They all have something to do with the map. I re-introduced the big station circles and the start of a new line. Also the small stations left and right are back now. The last thing I have to do is to avoid the overwriting of station-names. A station-name is place when the 'o' is typed. So I have to go back to the text and find all words with an 'o' which are close to each other and replace them with an 'ø' if I don't want a station-name to be placed.

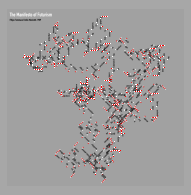
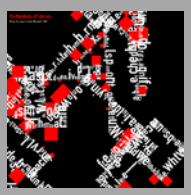
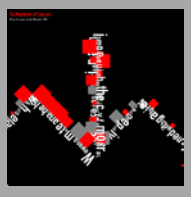
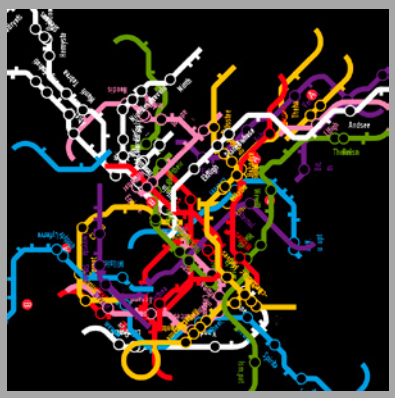
In the images that I have made with this program there are still some mistakes in getting the text aligned. But the overall look is good. When you study it's nice to know that you do not have to understand everything. It is also important that you do not reflect on mistakes too long.

http://www.amazon.co.uk/Mr-Becks-Underground-Map-Garland/dp/1854141686

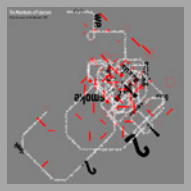
Wouldn't it be interesting if there was only one typeface in the world? Designers would really have to think about the idea behind their designs instead of covering them up with fancy typefaces. One, universal typeface would really strip away all the flashy emptiness in design. And, of course, that one typeface would have to be Helvetica.

Erik Kessels, 1966—, Dutch Creative Director KesselsKramer







Thehabi
MARINETTI TRANSPORT



enthusiastic
the






Lwaysta
Planes

smoke
time
action


Faust in Utopia

MyCodeHistory: 9 July 2014

This program is analyzing a text-file which on its turn generates images from the characters. Any text-file can be fed into the program. The color of the characters is connected to how often they appear in the text. But these values can also be used for other visualizations of the characters than color. There are four modifications of the program shown in the Generative Design book. In the beginning the characters which are not much used are getting a red color. Characters which are used most are yellow. Maybe its better to change from RGB to HSB. In that way I can control the colors better. Drawback is that the pdf-file gives weird color results. I wonder how long a text can be? I replaced the 'Faust_kurz.txt' file by a text-file with the complete book text of Thomas More's 'Utopia'. This made my system a bit less responsive. So I reduced the total text until there was only one paragraph left to analyze by the program.

In the second version of the program I changed a lot of times the background size and color. And than I made a mistake. Instead of making a strokeWeight 25 pixels thick I made it 250 pixels. There was still a previous typed zero in the line. That was over the top but when I made strokeWeight (100) the image looked pretty interesting.

I would like to find out what variable tracking does. It has a value of 29. I changed it to 50. It seems to increase the linespace. So I changed that global variable name into LineSpacing. Another thing was the color of the pdf-file. When I began with this program colorMode was initialized in the setup() block. Which does return terrible colors in the pdf-file. I moved colorMode to the draw() block. Lets see if that makes any difference. Success. It does solve the problem. So, to make pdf-files which use the same colors as the png-files you have to start colorMode in draw(). Not in setup().

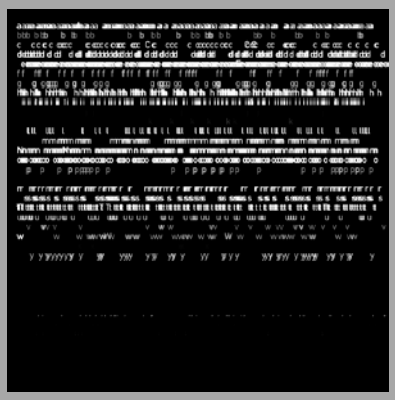
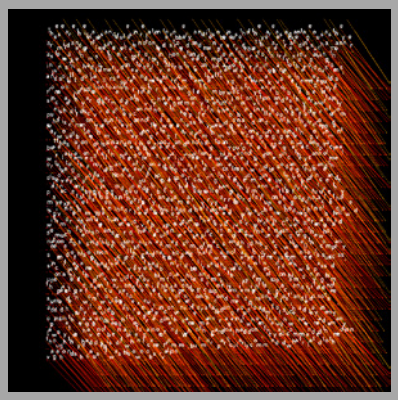
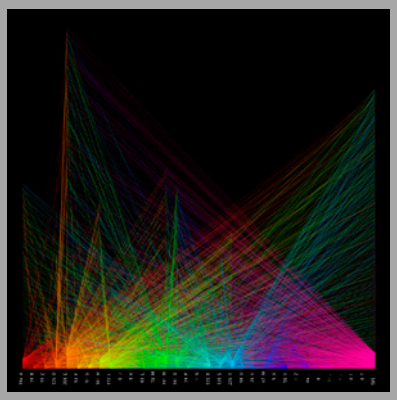
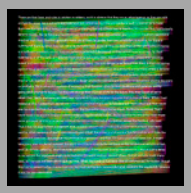
During working on the last variations I found one variable named charSize. And another one called charColor. These variables are not used in the program. I don't know why. So I commented them out. And as far as I can judge the program works normal. Another thing is that when I changed the German 'Faust_kurz.text' into the English 'Utopia.text' I did not realize that I could skip certain characters from the WhichCharacters string. ÄÖÜ and ß are characters which are not used in the English language. I also removed the alpha from the colors and set saturation and brightness to 100. Rotated everything —90 degrees. Everything is still not perfect. But this is a learning process I keep telling myself.

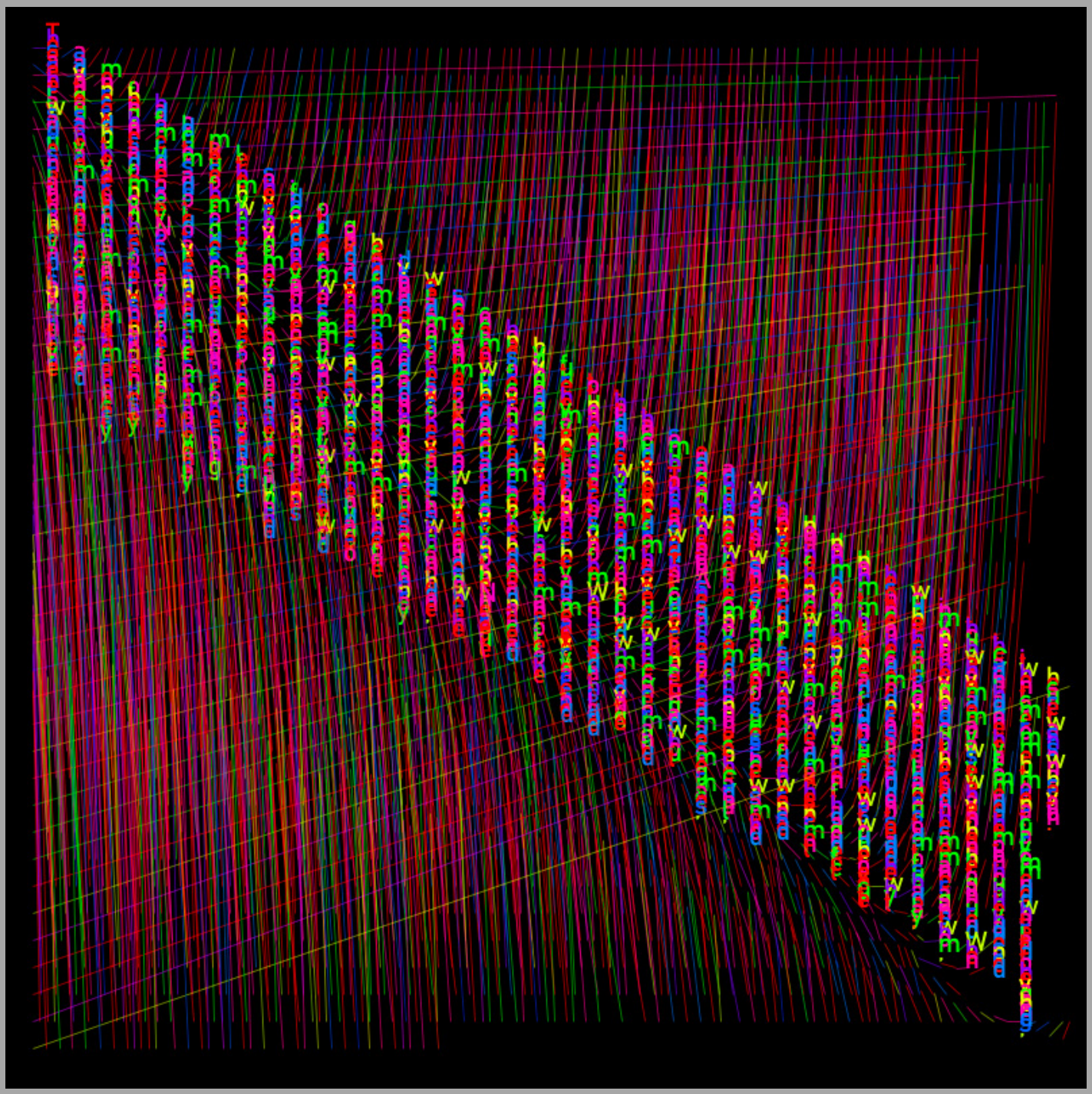
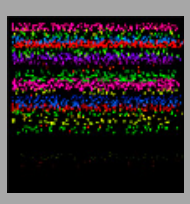
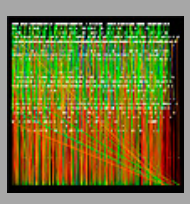
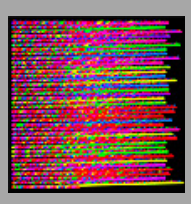
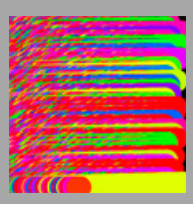
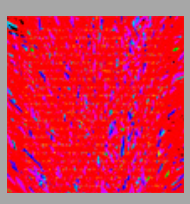
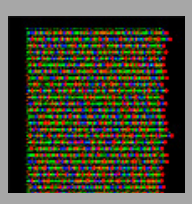
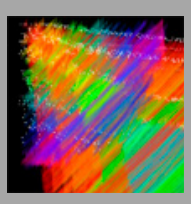
When you change the variable names to names which are more understandable to you it takes more time to get to the point where you can begin with the actual making of images. The chances are also bigger that mistakes sneak into the code. From the other hand you get more insight in the working of the program itself. Which gives you more opportunities to change things and make different images.

Designed from 1954—1957, Adrian Frutiger's typeface called Univers was notable for being one of the first sans-serif typefaces to form a consistent wide-ranging harmonious and modular type family. Frutiger knew that he only could accomplished that by determining the complete family range as a part of the design process and by building the family within a strict modular framework.

When we both worked for Total Design, from 1979—1984, Wim Crouwel assured us that you could create every imaginable type of graphic design, using only the Univers font family.

Wim Crouwel, 1928—, Dutch graphic designer, type designer, and typographer.

Writers blocks

MyCodeHistory: 17 July 2014

The previous program counted all the characters and generated visualizations of them. This program reads and processes large amounts of text (also known as books). It counts the frequency of words and represents them in a diagram using rectangles of variable size. I used five different books to import in the program. I began with the full edition of Sir Thomas More's 'Utopia'. Cleaned up the Utopian text-file. Changed the background, line and font color. Used the fill function to fill all boxes in the WordItem tab. In that way I can control the colors of the individual boxes better. I also wanted to show which text was displayed by mentioning the title, author and date of that book. Top left in the display window seems to be an ideal place to do that. It would even be better when I put the title under a key. By default the title is on. Hitting the zero-key will switch it off. Hitting it again will switch it on again. Defined an if block for the DrawTitle function.

I imported 'The Importance of Being Earnest', by Oscar Wilde. Added some refinement into the element fills. The smaller the element the darker it will become. This doesn't work very well in the SliceLayout and the StripTreemap versions. Next I imported 'The Devine Comedy' from Durante degli Alighieri and started changing the color mode from RGB to HSB. Used a for loop to create one gradient foreground per module. For another variation I have chosen 'Dubliners' from James Joyce. Used the for loop for gradients in a horizontal way. Spend a lot of time to get the gradient right.

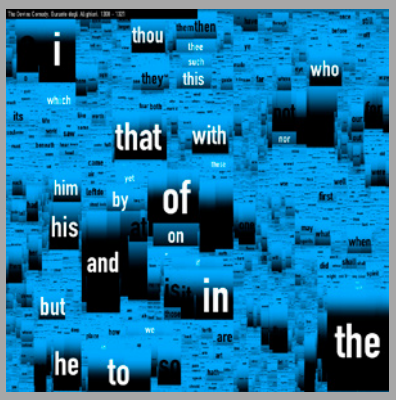
To give you an impression what kind of difficulties you might enter when working with this program, I displayed a number that indicated the amount of times a word occurred in the text. To test if that was possible I made a very simple text-file with just eight words in it. I wanted to know which function was responsible for counting the words. In my naivety I just entered a println (word, (word.length())); command in the WordItem tab. That gave me a list of words and the number of times they appeared in Franz Kafka's book 'The Metamorphosis'. I did a search on the word 'fresh' and I got 6 hits. When I do this in the list created in the command display I only get 5 hits. I tried another word. 'Between'. In the original text file 'between' is counted 7 times as a full word. In the command display I get zero hits. Original: morning, 22 hits. Console 7 hits. Lets take a very common word: 'the'. Original: 1148 hits. Console: 3 hits. The word 'and'. Original text: 624 hits. Console: 3. What is going on here? Could it be that the smaller the number in my console, the more it appears in the original text-file? I just assumed that 'the' is the most common word in the original text. It appears 1148 times. That would make that line: text (1148 – word.length(), x + (w / 2), y + (h / 3)); I am almost sure that this is wrong. All my console numbers are way above 1100. Word.length gives me not the amount of times a word appears in the text but the amount of characters that the word has. So the calculation is wrong. Time for a new approach. The square amount of pixels for one element is calculated by w x h. This must give me an indication for the amount of times a word occurs in the text. I already found out that the word 'the' occurs most in the text. And because it has the largest box. Width (w) times height (h) gives me the number 29807. The word 'staggered' only occurs once (w x h gives me 51). Calculating (w x h) / 26 gives me a very close outcome for the number of times a word occurs in 'The Metamorphosis'. Finally I connected the point size of the numbers according to the words appearance in the text.

I have no doubt about that libraries allow you to construct treemaps very easily. But they also limit me in my graphical expression. It does not matter who or what you import it will always going to look like what the library dictates. When using this program I was only able to do some decorative adaptations. But there is also the lack of information on how you can use a library to its ultimate limits. I guess I have to study them a bit more.

Lettering is a precise art and strictly subject to tradition. The 'New Art' notion that you can make letters whatever shapes you like, is as foolish as the notion, if anyone has such a notion, that you can make houses any shapes you like. You can't, unless you live all by yourself on a desert island.

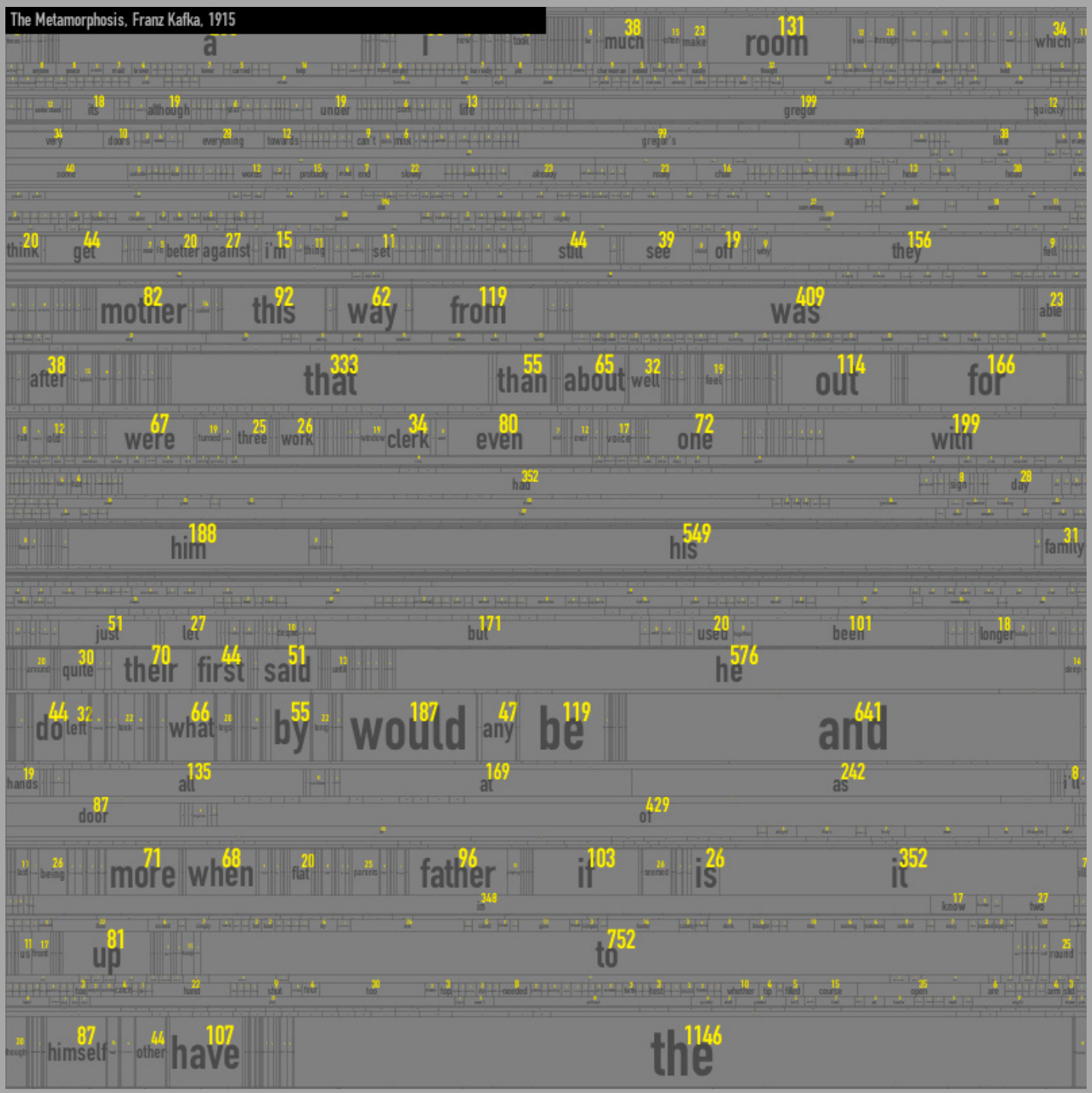
Stanley Morison, 1889–1967, British typographer, designer and historian of printing.

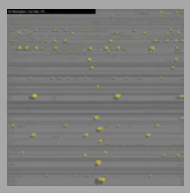

a
i
they
them
that
their
he
all
not
for
by
be
of
as
and
is
it
in
to
so
are
but
the

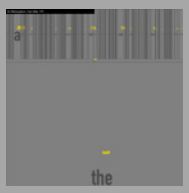

i
thou
then
they
this
who
that
with
not
him
by
of
his
on
and
in
but
the
he
to

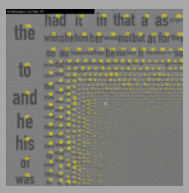

the
his
in
had
her
and
of
to
he
a
was

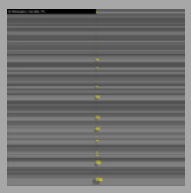

a
i
his
the
had
her
he
in
and
at
of
to
was

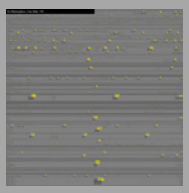

The Metamorphosis, Franz Kafka, 1915
a
I
much
room
which
its
although
under
life
gregor
quickly
very
doors
everything
towards
can't
again
like
think
get
better
against
i'm
set
still
see
off
they
mother
this
way
from
was
able
after
that
than
about
well
out
for
were
three
work
clerk
even
one
with
had
day
him
his
family
just
let
but
used
been
longer
quite
their
first
said
he
do
left
what
by
would
any
be
and
hands
all
at
as
door
of
more
when
flat
father
if
is
it
know
two
up
to
round
himself
other
have
the



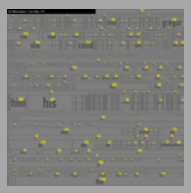

a
the


the
had
it
in
that
a
as
to
and
he
his
or
was






a
was
and
at
as
that
had
her
of
his
but
he
to


his
him

Modifying the outline

MyCodeHistory: 29 July 2014

Fonts, or typefaces all consist of a large number of characters. These characters are made as outlines fonts (beside bitmap fonts). This program modifies these outlines making use of the Geomerative library (by Ricard Marxer). Using this library you can generate additional points on the outline of the character. These points can than be used to modify the shape of the letter. When I started working with the program I decided to skip the program line which generates dots and I concentrated on lines. I changed the angle of the horizontal lines to vertical. After a while I switched off the lines and continued working with dots. Every point of the typeface is being used to create an ellipse. What about replacing all the ellipses with arcs? That worked! What about dividing the circle into 3 other parts? Works also! Now I can give every circle its own color. When I divide the HSB range of 360 with 6, it will give me 6 different colors for 6 parts of a circle. The upper half is a blue arc while the lower half is a red arc. Repeated the word 'undo' four times in the vertical direction. In the end I have thrown away almost all my modifications and continued with the original code. Used rectangles again instead of ellipses. And I also used the rounded corners. Four colored layers of the word 'wipe'. Defined 10 ellipses per vector. Got some free animation too. Used the animation to make a last version. But now with lines. The lines alone are not so interesting so I started mixing them with the previous version. And that gave me quite an interesting effect

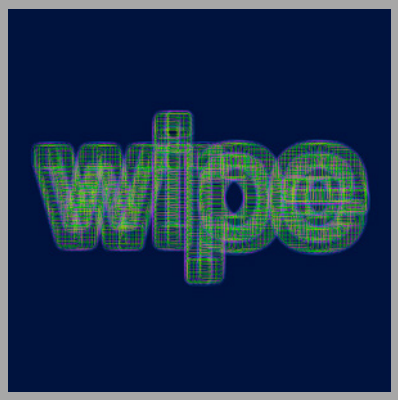
For the second part I changed the background and color-mode. Again I will search for four-, well lets take a three-letter word for a change. The word can now be displayed larger so you can see much more of its structures. After positioning the word its time to change the colors and the svg-files. During changing those svg-files I wondered if you could add more svg-files to a case. In the previous program I used twice the same svg-file to create a word. I thought the image would be way too disturbing using two different svg-files. But that was a mistake. Three different svg-files per case is a bit over the top. What will happen when I use only lines in the modules? Well… it seems there is not happening so much! So I used Adobe Illustrator's tweak filter over the lines. That made a difference, but not such a difference that I could appreciate it. Concluding its better to make a kind of regular patterns in Adobe Illustrator because the end result in Processing is much nicer to look at. Just straight lines give a beautiful effect. But I still wonder why it does not make a very big impact on the image itself.

In the previous program I mentioned that I was a bit sceptical about the use of libraries. But I was wrong. For this exercise I worked with the Geomerative library of Ricard Marxer. And learned that this is how you should present a library. Beside this the library is also well documented in a technical way. There are tutorials and much more information on how to use it. And this is what you need when you are involved in a learning process.

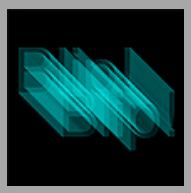
In 1990 Erik van Blokland and Just van Rossum designed the RandomFont called Beowolf. 'Beowolf became an example of what digital type could be: not that the random effects were aesthetically appealing, but it was proved that fonts were no longer a physical object but they became a number of instructions. It also showed us that code and design can merge into a single process, resulting in one single object. Code has a major influence on design, and I find it to important to leave it to anonymous engineers.'

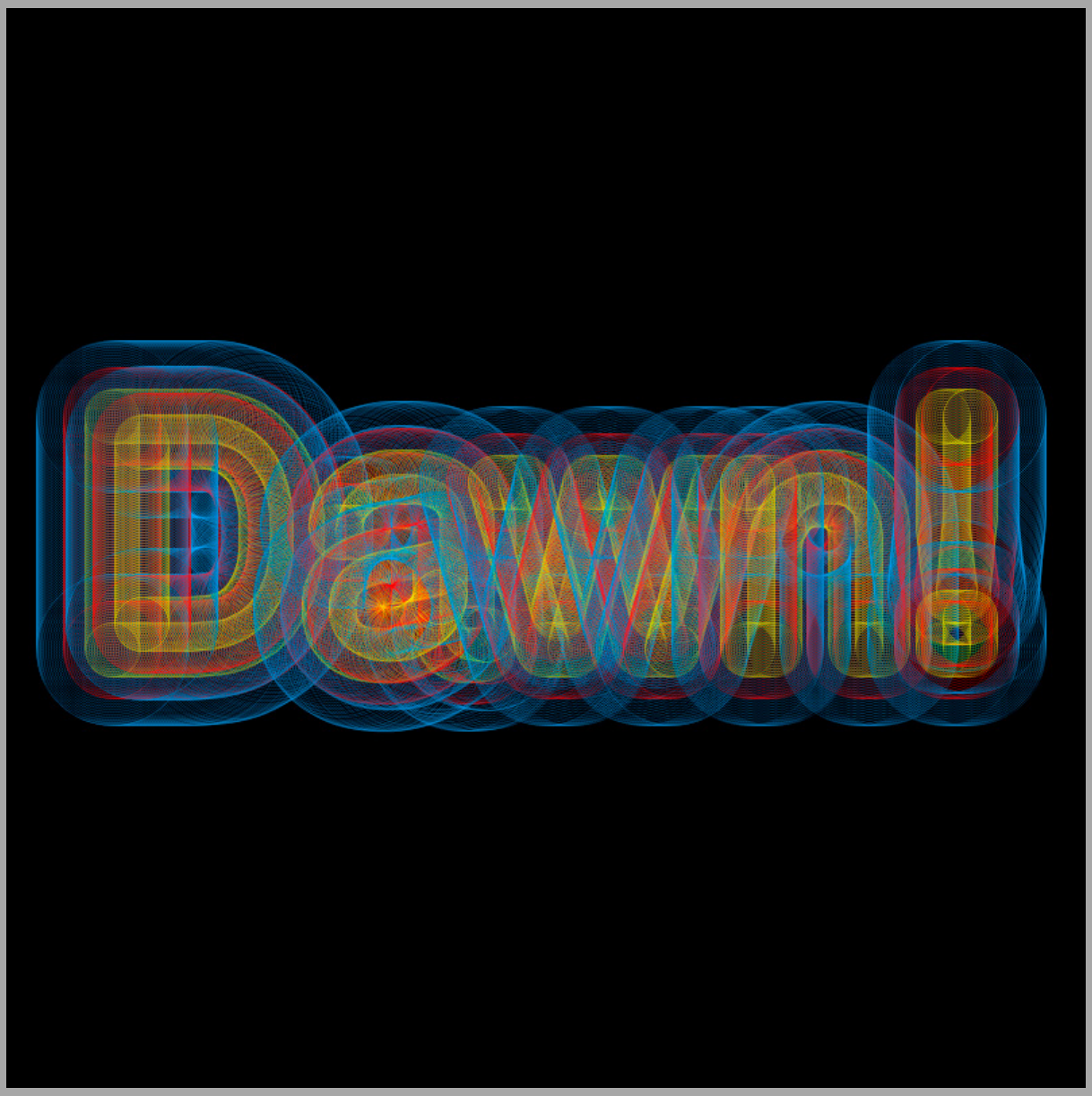
Erik van Blokland, 1966–, Dutch typeface designer, educator and computer programmer, interview in 'Processing Second Edition', 2015 MIT Press

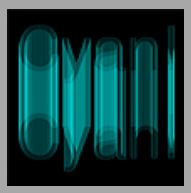

epic!

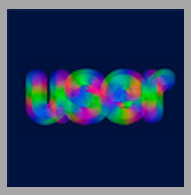

wipe

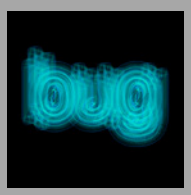

Flex!

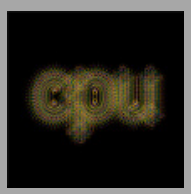


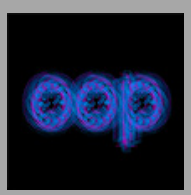

Dawn!

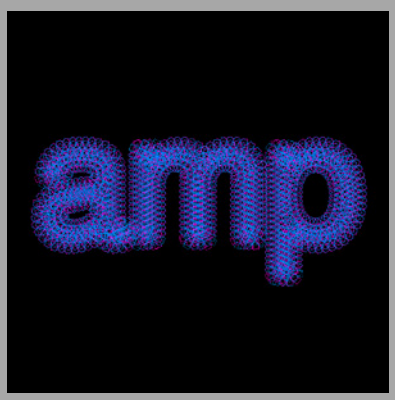


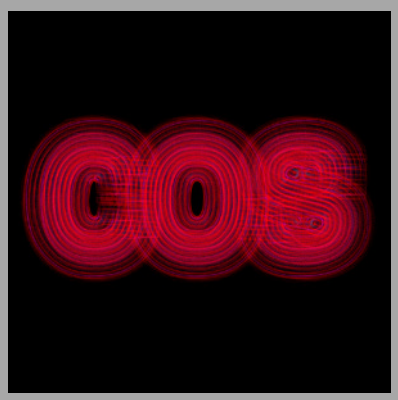


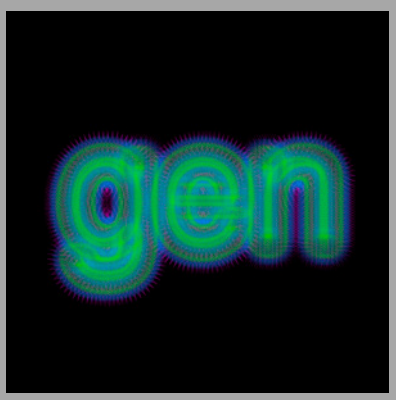


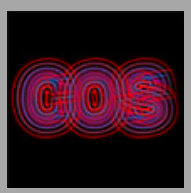


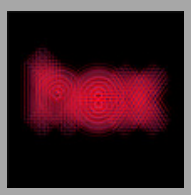



amp


cos


gen


cos



Modifying Casteljau curves

MyCodeHistory: 6 August 2014

Peugeot's Pierre Étienne Bézier popularized the bézier curves but he did not actually invent them. The curves were developed in 1959 by Paul de Casteljau while working at Citroën. With this program the shape of the characters bézier curve can be modified. Horizontal mouse movements control the height and vertical mouse movements control the rotation of the bézier curves. I copied and pasted a line with the bézier statement and multiplied the values of the curves 1.5 times. But multiplying the curve values 1.5 times seemed way too much. Beside that the program generated leaf-like objects around the characters which I could not appreciate. But I couldn't resist to change the fill color to green. What will happen if I exaggerate the length of those leaves? It's going to look more like feathers of a bird. But it's also getting too chaotic for my taste.

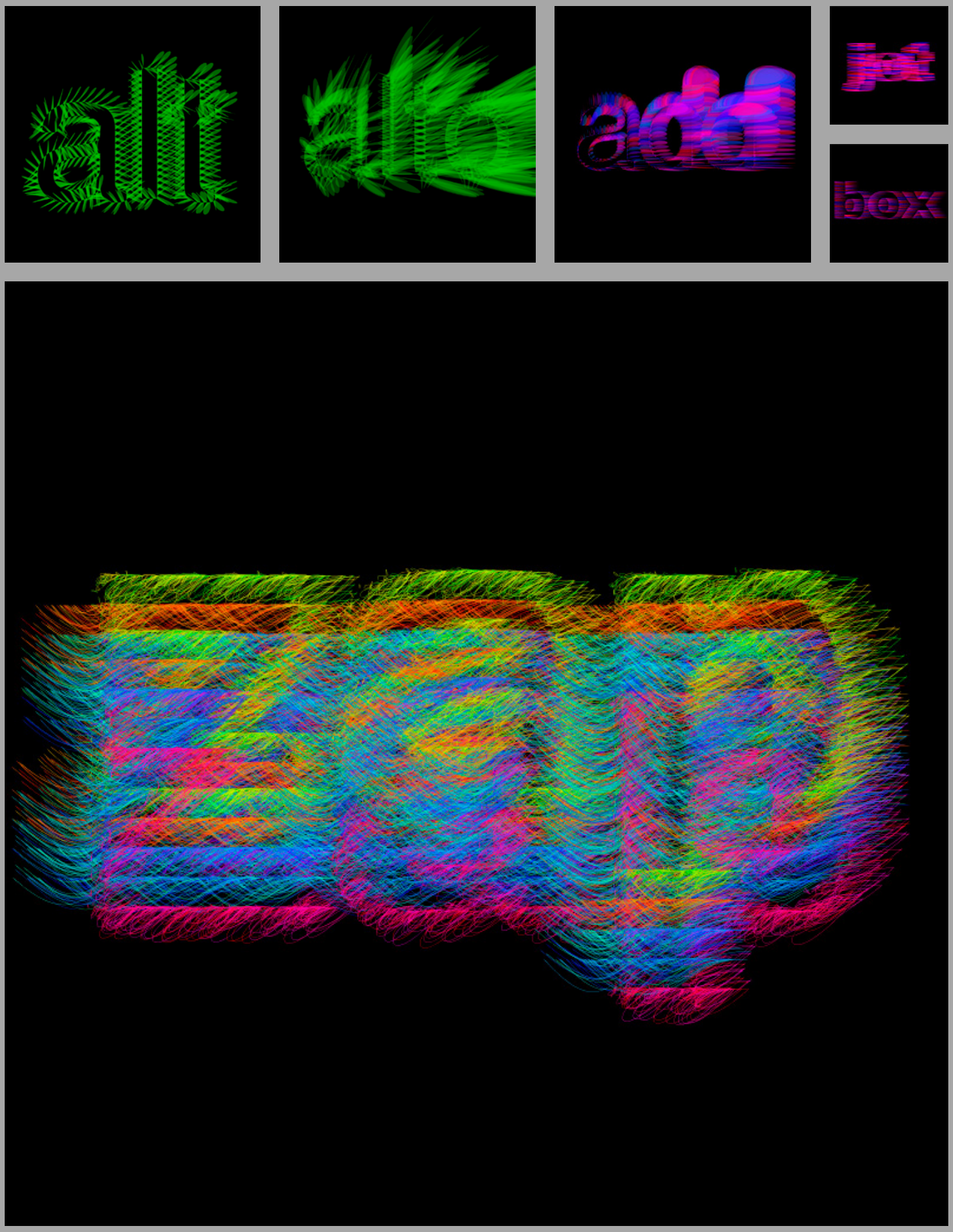
I removed my second bézier program line and added a random fill just before the curves were drawn. And that's almost a good effect. I still need to change the typeface to bold which shows the effects better. Arial Black. ttf would do the job. I always thought that Arial was Microsoft's rip-off from Helvetica. But I was wrong. When Adobe was busy with developing PostScript, Monotype was providing fonts for IBM's first laser printers. This led to the design of Monotype's Arial typeface in 1982, by Robin Nicholas and Patricia Saunders. Several years later, Arial was also licensed to Microsoft and was supplied with all versions of Windows. But there are also visual differences. Helvetica's terminal strokes are either horizontally or vertically cut. While those of Arial are slightly on an angle. The overall shapes in Helvetica are more straight. Arial's shapes are more rounded.

I was considering to add a noloop in the draw block because the program generates unwanted animation. I had to create a key-function to switch the animation on and off. And that introduced another strange artifact. The right side of the character is repeated. I guess it has something to do with the translate function. So I got rid of the noloop. But what does this RCommand.setSegmentator(RCommand.ADAPTATIVE); do? The Geomerative reference says: Use this to set the segmentator type. ADAPTATIVE segmentator minimizes the number of segments avoiding perceptual artefacts like angles or cusps. Use this to have polygons and meshes with the fewest possible vertices. I changed all settings back to the original program except for the font. Increased the value for the curves y-distance to 25. Decreased the curves x-distance to –10. That gives interesting shapes but the characters shape has totality vanished. Lower values are completely silly.

It seemed that I inadvertently solved the 'animation' problem of a few previous program-versions. I did not realize that there was a feature in the code to add text. When you hit a key it will be added to the text in the display window. I commented it out in the keyPressed block. I didn't need it. So I ended up with a kind of flame throwing characters. I switched off the fill statements by replacing them with stroke statements. Used the local variable jitterPoints to check what kind of effects were possible. This did not lead to any interesting things. So I used the fill statements again and removed the strokes. Placed the word 'sun' three times above each other. I noticed when you place two (or three) words on top of each other (with different colors) you get another interesting effect. Made three variations with those settings.

During the design process it is possible to create images you don't llike. You have three options for this situation. Trash it and start again. Or try to change the design until you like it. However, most of the time I do both. I will change the design until I like it and when I do like it, I start all over again until I have enough of it.

I mentioned (on page 14) that in 1990 Just van Rossum and Erik van Blokland (LettError) created their randomfont FF Beowolf. Now, more than 25 years later, you can also change outline fonts using this Generative Design program. In 1988 Just van Rossum was a trainee at Océ Research & Development's Typographic Design Department to gain some field-experience. I also worked for Océ at that time. Just was always working very late so I asked him one day what he was doing. He said: 'I am working in PostScript!' Well, I knew what PostScript was, but we were designing screen-layouts for the Océ 6000 system. The first (and last) desktop publishing system ever developed in the Netherlands. I wondered how you could work in the PostScript language? So I asked Just and he showed me a screen with an incredible amount of code. I was wondering why would a designer dive into such complex matter? And why didn't I ask Just that question at that time? We were surrounded by dozens of software engineers who wrote all the software for us. There was no reason for me to learn programming.


alt
alt
add
jet
box
sum
sum
sum
sum
sum
sum
sum
sum
sum
fix
fix
fix
fix
fix
fix
fix
fix
fix
zap

Stupid agents as outlines

MyCodeHistory: 16 August 2014

In the previous program I saw that it's relatively easy to modify a font until it becomes unreadable. In this program the points of a character act like stupid agents. The longer the stupid agents are on their way the more the character gets eroded. To begin with I wanted to use only one character in the program: the lowercase 'a'. Using only one character I could better see the visual impact. I moved the 'a' to the middle of the display and let the program run for a while. I also switched off the frameRate. And that made the program run way too fast. Beside of that the program is getting very chaotic after some time. In return I modified the variables StepSize and DanceFactor. StepSize is responsible for the distortion and DanceFactor is responsible for the amount of wiggling. Enlarged the ellipses on the character and replaced RGB for the more human friendly HSB colorMode.

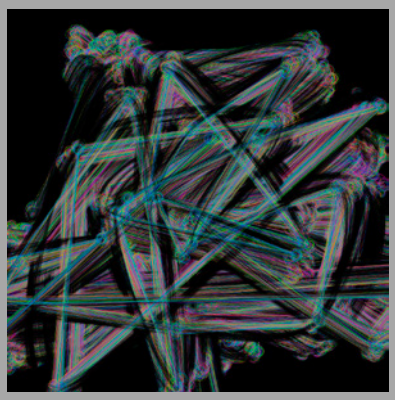
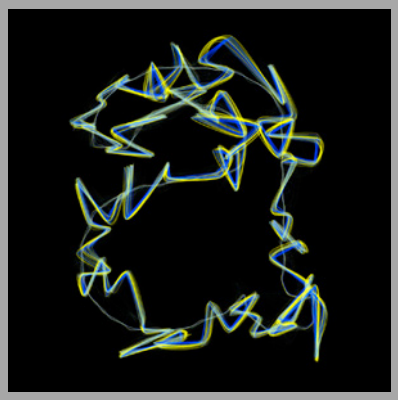
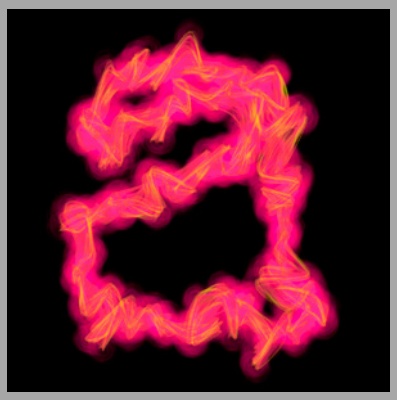
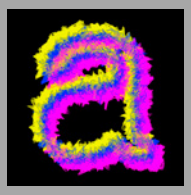
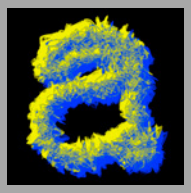
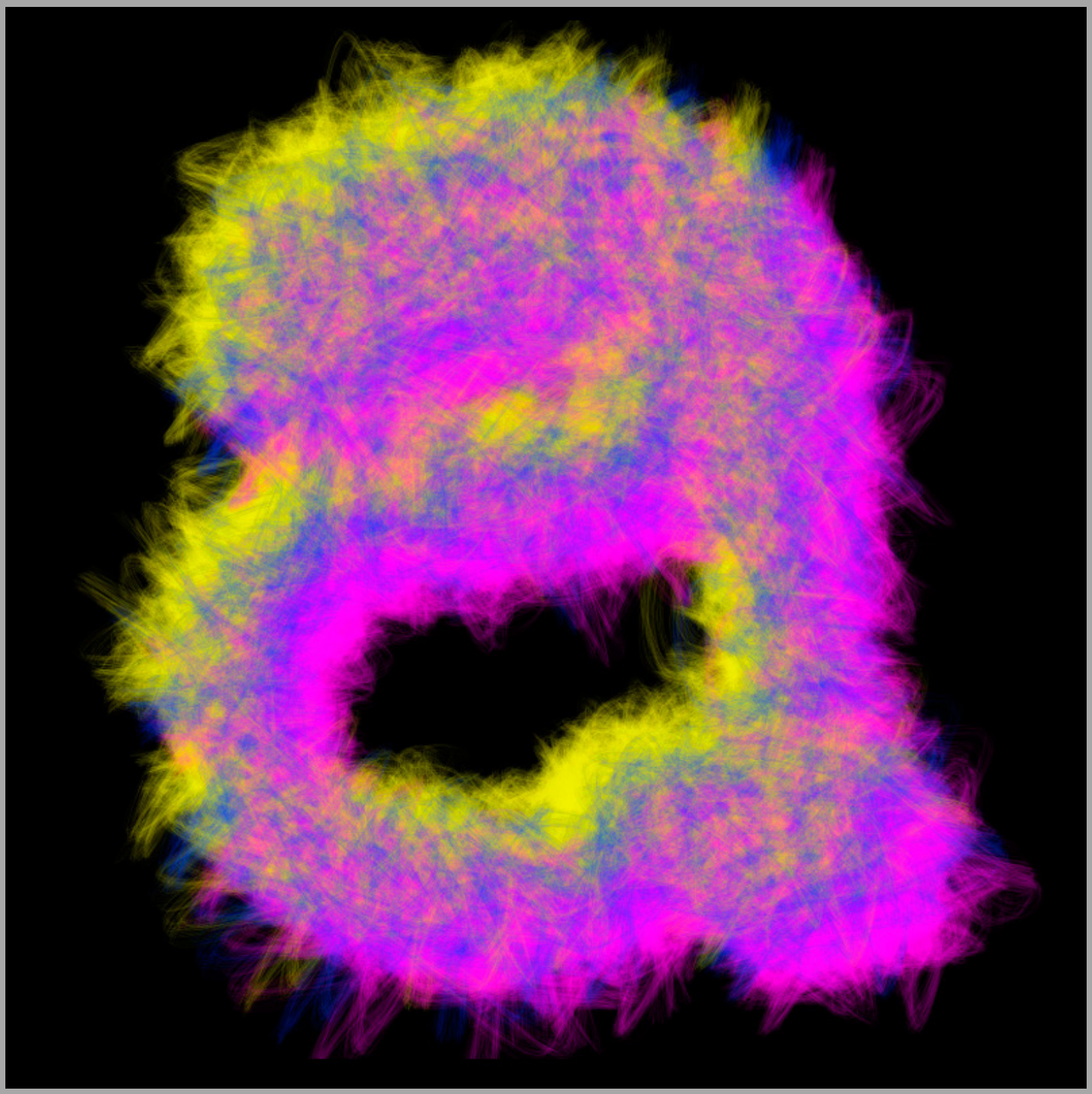
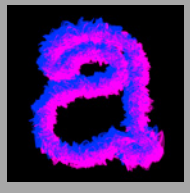
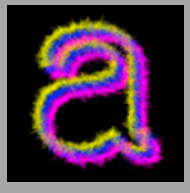
Let's see how to control that chaos a bit. I switched on the frameRate again. Made the font size 800. Randomized the fill color settings between 300 and 360. Randomized the color settings for lines between 50 and 60. Increased also the alpha settings to 4. Adjusted the DanceFactor to 10. Commented out all the ellipses and made the strokeWeight 0.001. I found that the distortion was still going way too fast so I decreased the StepSize to 0.1 (which turned out to be too less). 0.5 Seems to be a good-one. Just for fun: RCommand.setSegmentLength (1) gives a complete different picture if you let it run for a minute or five. Interesting is that the original shape of the character remains pretty well intact. It seems that StepSize is the most important variable. When you rise its number to a 100 you see why.

Why does the program display the same character twice but in a different color? Commented out the first beginShape Endshape block. Ok. Now it only displays the blue character. So I could add another block in a different color? That works too. Got a blue and magenta version of the 'a'. Uncommented the first block and I have three different colors of the same 'a'. Increased the StepSize to 6 which gives me a kind of fluffy character. I tried another font: Times Roman. That didn't work at all. Also the character spacing was wrong. Changed it back to Arial Bold but letter-spacing is still wrong. Made a few proposals with very 'airy' or 'smokey' characters.

I have to make one remark about the letter-spacing, which is very bad. It does a terrible job in the larger typeface ranges. Headlines are out of the question in Processing. To be honest I had to correct them in Photoshop. Finished making variations and made a few versions with rectangles which reminded me that I also could use my own shapes. Which, reminded me that the possibilities are endless.

In 1983, Susan Kare designed the fonts for the first Apple Macintosh computer. She named them after railway stations along the old Main Line: Overbrook, Merion, Ardmore and Rosemont. But Steve Jobs complained that no one ever heard of these small towns. They should be replaced by names from the metropolises of the world. And that is why, according to Susan, you can find fonts with names like Chicago, New York, Geneva, London, San Francisco, Toronto, and Venice on Apple computers.

In 2005 I interviewed a typographer during a typography conference and asked him: 'Are there any bad fonts?' To which he replied: 'There are a lot of terrible fonts. For instance all fonts which are named after a city.'

abcd
efghi

slum
swap

ANO
NYM

SOS
SOS
PIPS

abcd
fghij
klmm