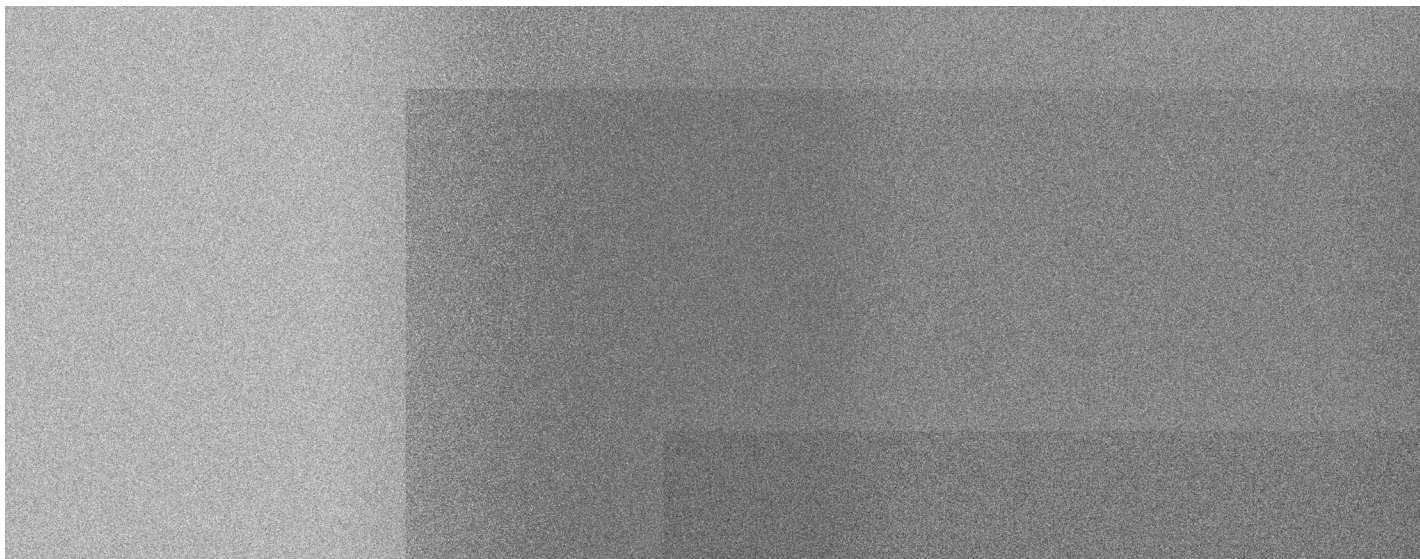


CODE, SHAPE AND MEANING

CAIO BARROCAL





“The machine, which is thought to be cold and inhuman, can help to realize what is most subjective, unattainable, and profound in a human being.”

- *Vera Molnár*, computer art pioneer.

CAIO BARROCAL

CODE, SHAPE AND MEANING

A STUDY ABOUT ARTS AND COMPUTING

Undergraduate thesis produced under the supervision of Dr. Daniela Kutschat Hanns.

São Paulo. November, 2018.

CONTENTS

5	Introduction
7	A historical parallel between arts and computer science
8	Mathematics and art
16	The development of computer graphics as a supplier of aesthetic relevance for computer art
27	Computer art
29	Computer art pioneers
43	Generative and algorithmic art
51	Current aesthetics and motivations
52	Coding for visual engagement
58	Coding for engagement through actions
63	Coding for engagement through involvement
68	Current tools for computer art
69	Processing
70	P5.js and processing.py
71	Processing case: Plausible, Possible, Potential, by Miguel Nóbrega
73	OpenFrameworks and C++
74	OpenFrameworks case: MULTIVERSE, by fuse*
77	Cinder
78	Cinder case: Red Bull Music Academy 'A Night of Spiritual Jazz' Installation by The Mill
81	Interview With Jarbas Jácome
87	Self created experiments
88	Cloudy Sky Simulation
90	Sea Waves or Mountains
94	Perspective Studies
96	Northern Lights
99	Colored Surfaces
102	Horizontal Energy
105	Curved Seduction

Introduction

Art, just like computing, has never been a limited and narrowly-defined field. Many are the applications, abstractions, movements, methods and motivations for creating things that, across the history, justify their own existences while part of great human disciplines.

Actually, labeling something as art by uniting technique and concept to materialize a given motivation, implies similar things to designing an algorithm or a system and justifying its efficiency and correctness. Leaving aside, for a moment, the commonly rational nature of computing problems, both areas deal with culturally specific communities, challenges and methodologies that, when solved, bring to life abstractions and objects that become a very part of our society's routine.

Written by a very almost graduated computer scientist who is also passionate about art and design, this undergraduate thesis seeks to investigate and also to present the art-technology field and, more specifically, computer art; its common tools, history, motivations and also the people whose life events and experiments contributed a lot for its establishment.

One of this project's intention is also to introduce the art-technology field as a fruitful medium for multidisciplinary collaborations between technical people (computer scientists, engineers, developers), artists and designers. Collaborations that, by taking advantage of the specificities of each one of these professional profiles, result in artifacts and experiences that are gaining more and more relevance in society today.

Chapter I discusses the relationship between computing and art by first investigating how mathematics, being the area that encompasses computer science, and art have met each other across history. Then the chapter explores how the developments in computer graphics aroused the interest of a community of artists that started to use the computer to create art.

Chapter II goes deeper on computer art, presenting its pioneers and some of their works; showing main methods for algorithmic creation and also introducing generative art.

Chapter III seeks to showcase contemporary art installations that employ computer art and generative techniques. The interactive installations discussed are grouped into three main categories: coding for visual engagement, coding for engagement through actions, and coding for engagement through involvement.

Chapter IV discusses the programming languages and frameworks most used for creating art with the computer today: *Processing*, *openFrameworks* and *Cinder*. Each one of the tools are accompanied by a case that exemplifies its power and expressiveness.

Chapter V is actually an interview with the Brazilian artist and computer art researcher *Jarbas Jácome*. In this interview, the artist comments on his trajectory, the creative process he employs and also on some of his projects.

Chapter VI, the final chapter of this project, presents some self created works. Aims of these experiments were to illustrate the techniques and concepts studied during the whole process of writing this undergraduate thesis.

A HISTORICAL PARALLEL BETWEEN ARTS AND COMPUTER SCIENCE

Writing about the relationship between computer science and art is not an easy task. Most efforts for using the computer to make art have begun on the 1960s and this happened in the midst of an effervescent and diverse artistic scene. People like *Vera Molnár*, *Manfred Mohr* and *Lillian Schwartz* are ones of the pioneers of the so-called “computer-art”. They started experimenting with computers and algorithms in a time when these machines were still very expensive and far from powerful.

Motivated by the ever-increasing improvements and investments in technologies associated with computer graphics, some artists saw the computer as an unexplored and possibly productive medium for artistic production. The collaboration between arts and technology became more evident and expressive and, during the second half of the 20th century, the computer reached the level of aesthetic relevance that it has today.

The origin and adoption of videogames; the generation of graphics and sounds for increasingly exciting audiovisual productions; the penetration of computing and technology into popular culture; and computer art itself were just some of the phenomena that put the computer at the artistic level which it occupies today. Society has reached a scenario in which it's getting harder to dissociate artistic creation from computing and technology.

It's a characteristic of artists to experiment and work with the most advanced mediums that their time presents them. Nowadays the mediums recognized as the most thought-provoking for users, and the most innovative for artists are all essentially computational: the tablets, smartphones, virtual and mixed reality devices, video games and interactive art installations with sensors and actuators (*Santaella, 2012*).

This historical parallel aims to establish guides for studying computer art as a daughter of the relationship between art and mathematics throughout history and, at the same time, a product of the improvements in computer graphics happening since the last century.

Mathematics and Art

Using the computer as a medium for artistic production is a “recent” phenomenon. Many relevant experiments took place in the 60s when programmers started exploring algorithms and machines to generate shapes and sounds. Mathematics and art, though, have met each other many times across history and, since computer science is essentially a subarea of mathematics, studying this relationship is significant.

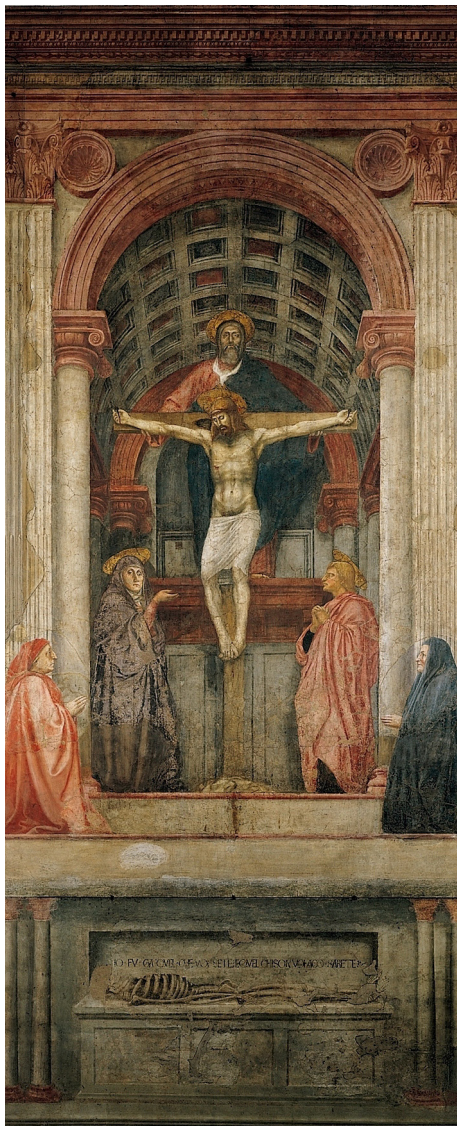
A good starting point for exploring how mathematics made its way into arts is discussing the developments in the understanding of perspective and bidimensional representation. Actually, since most screens and devices work with bidimensional technology and cartesian coordinates systems, representing a 3D world is still a very relevant and aesthetically challenging problem to be solved.

After the Middle Ages and the loss of most artistic sensibilities left by the Greeks and Romans, *Brunelleschi* (1377 - 1446) is credited with the re-discovery of depth and with the correct formulation for linear perspective and the understanding of scale. The architect and engineer was one of the pioneers of the Renaissance period, and is most famous by planning and inventing machines to build the dome of the *Cathedral of Santa Maria del Fiore*.

It was a monumental discovery when *Brunelleschi* experimented with a mirror to paint the Florence baptistry in perfect perspective and to develop his theory about the vanishing point (that there should be a single vanishing point to which all parallel lines in a plane converge). He influenced many other artists and mathematicians who were looking for ways to create convincing depth in paintings and who could perpetuate his findings (*Robertson; O'Connor, 2003*).

One of these artists was *Masaccio* (1401-1428), the Italian artist who painted the “*Holy Trinity*”, “*The Tribute Money*” and “*Virgin and Child with Saint Anne*” frescoes. Paintings that are credited for being the first ones created with *Brunelleschi*’s newfound mathematical principles and that, consequently, reached an unprecedented aesthetic result.

Following the Renaissance efforts of creating an aesthetically convincing perspective system, *Piero della Francesca* (1416-1492) wrote some fine mathematical texts discussing geometry, algebra and the application of perspective principles in art. Actually, besides being a very skilled artist, he was also a leading mathematician looking for ways to improve his artwork exploring geometry rules.



Masaccio. The Holy Trinity, with the Virgin and Saint John and donors. 1425/27.
One of the artist's most famous fresco.



One of Francesca's frescoes at the Basilica of St Francis, Italy.

In the first volume of the treatise *On Perspective In Painting* (near the 1460s), *Piero Della Francesca* establishes some geometry theorems and then discusses their application onto plane figures. In other two volumes, the artist examines three dimensional drawing techniques applied to prisms and deals with more complex shapes such as human body parts and some architectural ornamentation (*Robertson; O'Connor, 2003*).

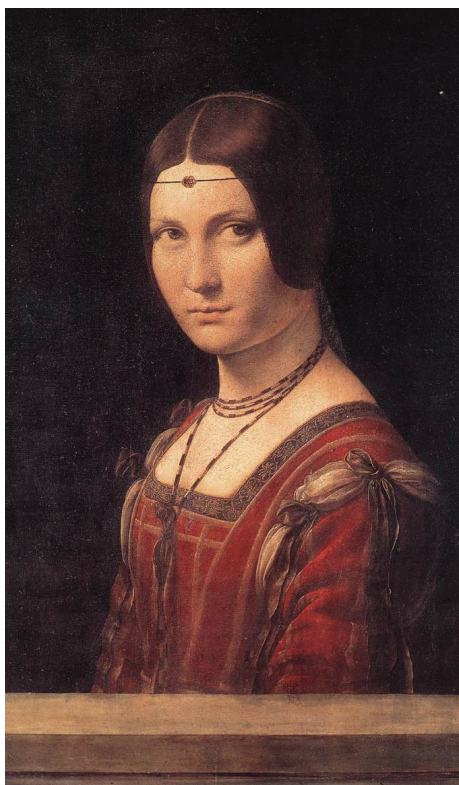
In their article *Mathematics and Art* (2003), *John O'Connor and Edmund Robertson* present a translation from the introduction of *On Perspective In Painting* in which *Piero della Francesca* defines painting according to his methods. The definition expresses *Piero's* very intentions of formulating a highly geometric, and therefore, mathematical explanation of painting and representation:

“First is sight, that is to say the eye; second is the form of the thing seen; third is the distance from the eye to the thing seen; fourth are the lines which leave the boundaries of the object and come to the eye; fifth is the intersection, which comes between the eye and the thing seen, and on which it is intended to record the object.”

Piero della Francesca's attempts of abstracting the real world into geometric elements express his fascination with geometry and mathematics and, also, how these rational concepts were actually a very significant part of his creative process. Having mathematics as a guide for painting, *Piero* could reach impressive aesthetic results and accomplished presentations of forms in space.



Piero della Francesca. Polyptych of Perugia. 1470.



Top:
Leonardo Da Vinci. La Belle Ferronniere. 1490.

Bottom:
Leonardo da Vinci. Annunciation (background
landscape). 1472.

Leonardo Da Vinci (1452 - 1519) was one of *Francesca's* most famous counterparts. Not only did he echo some perspective concepts developed by *Piero* and *Alberti* but he also made important contributions to the artistic community with his further studies of perspective and the optical principles of the eye. *Leonardo* was conducting his own studies to find out how the distance and the position of the spectator could influence his perception of the artwork and where this person should be positioned to perceive the correct perspective.

The Renaissance painter claimed that perspective could be well understood if compared to seeing a place or objects behind a plane of translucent glass and paying attention to the way in which the shapes intercept that surface. This “open window” approach was shared by many other Renaissance artists who were also investigating how to represent a real three dimensional world with a flat surface.

Advancing in his experiments of reaching a more realistic representation, *Da Vinci* distinguished two different types of perspective: *artificial perspective* and *natural perspective*. The natural one refers to projecting objects onto the plane by reproducing faithfully their relative size according to the distance between them and the painter, while the artificial perspective refers to projecting the objects onto the plane having the foreshortening effect in mind, so that, depending on the angle of the viewer, he would perceive his own perspective. To illustrate this difference, one can imagine a painter standing facing south towards a road that goes perfectly from east to west; in a natural perspective, this painter would draw the cars exactly how they appear, bigger on the center and smaller when close to the edges of this scenario, while in an artificial perspective, the painter would ignore this natural effect and draw each car at the same size, leaving for the spectator the role of perceiving them differently by varying his or her position in front of the portrait.





Top:
Albrecht Dürer. Erasmus of Rotterdam. 1526.

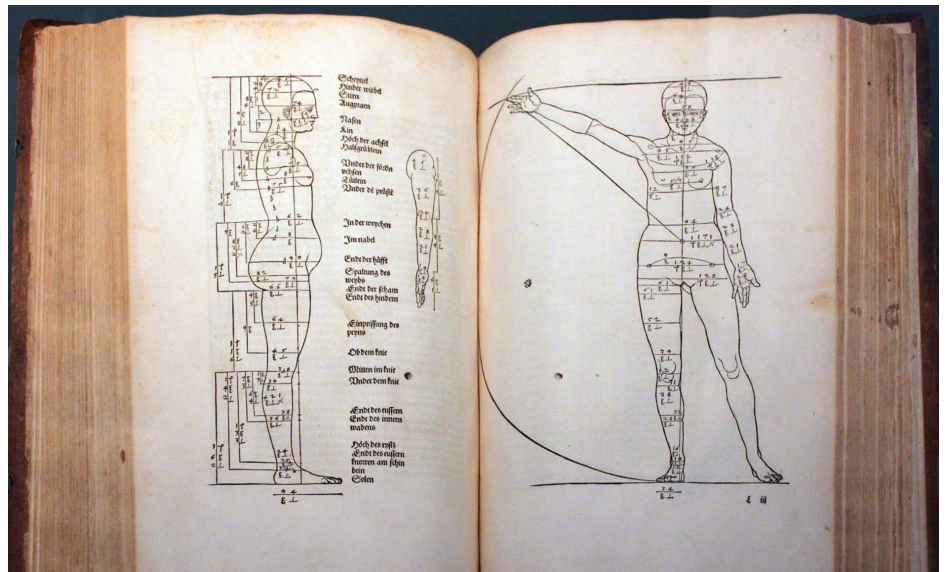
Right:
A page of Dürer's Four Books of Human Proportion.

Da Vinci's artwork and methodology express his own view of science and the functioning of the world. The deep understandings of perspective and the physics of shade and light that *Da Vinci* intensely developed across his life not only based his paintings and engineering projects but turned out to be his very signature and the epitome of a Renaissance man.

In his various sketches, *Da Vinci* registered ideas, theorems, prototypes, mathematical concepts and the real world itself. The artist also made several illustrations for mathematical books such as *Luca Pacioli's De Divina Proportione* (On the Divine Proportion), a book about mathematical proportions and their applications to art and architecture. Perhaps in *Leonardo*, more than any other Renaissance artist, mathematics and art were fused in a single concept (Robertson; O'Connor, 2003).

Besides pure perspective laws developed by the Italian painters, important contributions have been made by *Albrecht Dürer* (1494 - 1528), a German artist who has become the most famous one of German Renaissance. *Dürer* made an important addition to the studies developed before him by stressing the importance of light and shade in drawing the correct perspective and realistically representing shapes. Actually, various of his works are made only exploring representation and perspective through shade and light.

The German artist was personally interested in the human form and in methods for drawing it correctly, as demonstrated by his most famous written contribution: *Four Books of Human Proportion* (*Vier Bücher von menschlichen Proportion*). Books in which he explored human forms, seen from different angles and discussing about their proportions. *Dürer* also wrote a manual called *Underweysung der Messung* to introduce geometric theory for students that includes the first scientific treatment of perspective by an artist from northern-Europe.



Besides perspective, proportion has been another concept heavily studied by artists across history and that plays an important role with the overall harmony of an art piece and the perception of it. When creating with proportion in mind the artist is concerned not only with the size of one object in relation to another, but how each object's size relates to the whole composition. Mastering proportion is also necessary for generating the correct perspective.

Proportion, scale and perspective are some of the geometrical concepts that establish the balance of a creation. When seeing an art piece, people look instinctively for a sense of balance that brings familiarity and helps them understand it, so when created accidentally or purposely unproportional, this work can be disturbing because of its unfamiliarity. The proportion of elements in a whole is also associated with the perception of beauty, even though this very perception of beautiful proportions changes over the ages and is strongly associated with the cultural context.

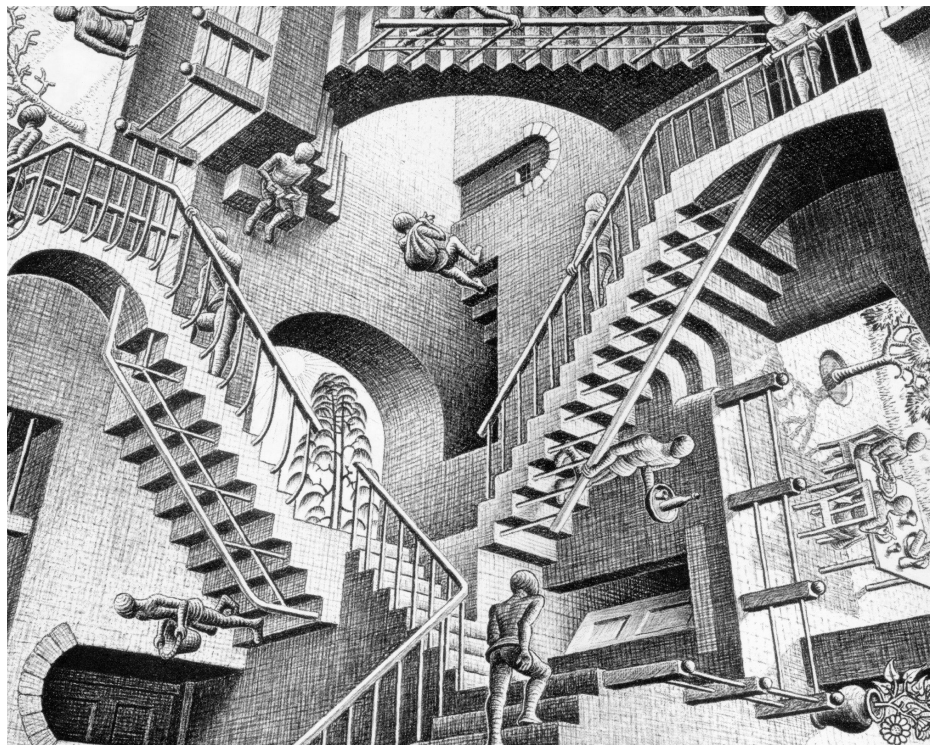
After studying the significant cultural impacts of fusing mathematics and art in a single approach, a perhaps more explicit way of discussing the importance of this relationship for computer art is exploring the use of patterns and their very algorithmic nature. The idea of iterating through a chosen number of repetitions and drawing accordingly to some specific rules is very familiar for a programmer.

Besides helping setting the rhythm of an art piece, the use of patterns may contribute for grabbing the spectator's attention, even if it's subtle or very apparent. Patterns can be incorporated into a work of art in a variety of ways: through the repetition of geometrical or natural shapes; by iterating through the same color palette following a logic; or even by the creation of a series of isolated works of art that, when put together, generates a pattern.

Actually, playing with patterns in its various ways can result in thought-provoking pieces of art. As stated above, human mind tends to enjoy and to look for a rhythm and a sense of balance when contemplating art pieces, so that when irregularities are found it can grab one's attention for a desired aspect of the composition such as an specific point, an implicit message, or the very disturbing nature of it.

M. C. Escher (1898 - 1972) is an example of a famous artist that creates pieces of art by playing with patterns, rhythm and the perception of the world. The Dutch graphic artist conceived his works using mathematical concepts that resulted in instigating aesthetic objects and a very characteristic style.

M. C. Escher. *Relativity*. 1953.
Litograph print that is also one of the artist's most popular works.



In his drawings and engravings, *Escher* sought to explore the space and the many possibilities obtained when trying to represent the three dimensional world with bidimensional analogies. He created shapes and scenarios using techniques such as rotations, translations and reflections, having as a guide the paradoxes that this translation of dimensionality naturally provokes. For the very mathematical nature of his works, *Escher* was considered an important geometrical mathematician.

Max Bill. *Tripartite Unity*. 1951
Photographed by Hans Gunter Flieg in 1951.



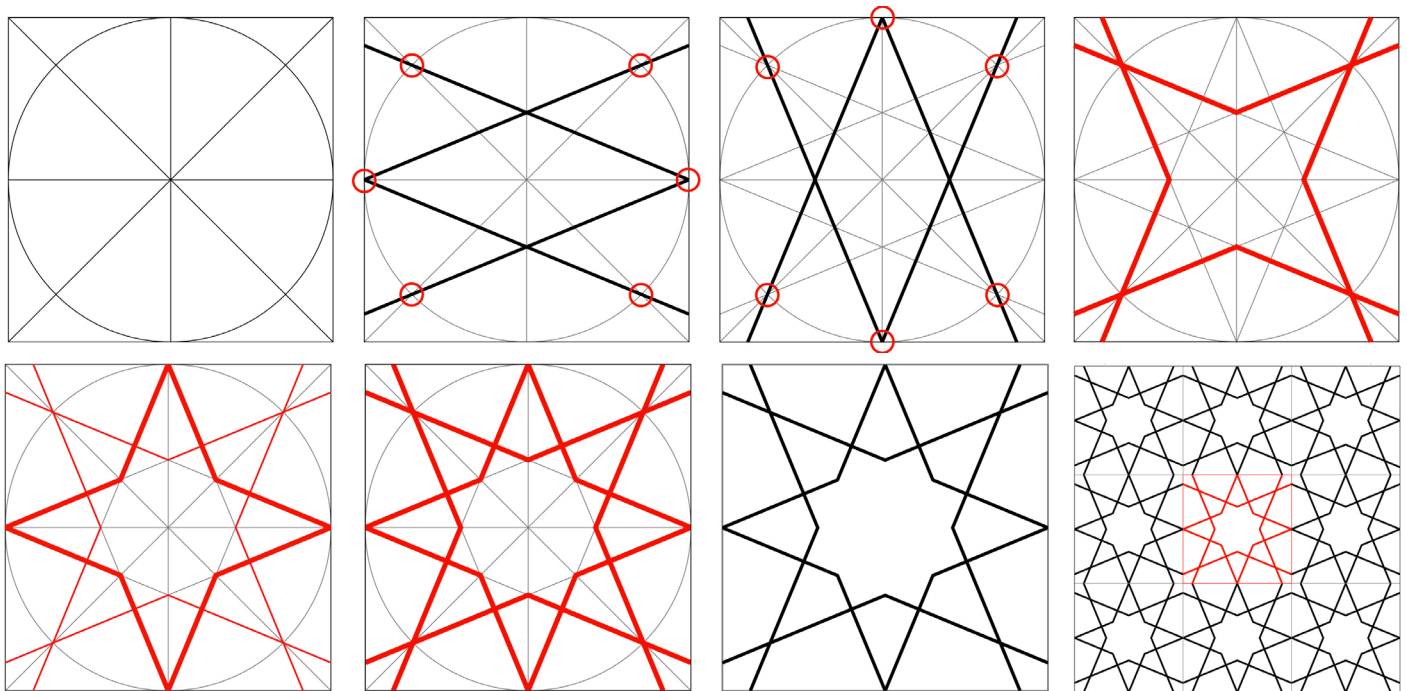
Escher's work plays off our desire for patterns and that is why it is so captivating, he creates an illusion that would not work if it did not rely on an uncertainty of pattern. The result is a piece with high impact that is memorable to all who view it (*Esaak, 2018*).

Another case of piece of art that relies on mathematical concepts to play with rhythm and patterns is *Max Bill's Tripartite Unity*. The Swiss artist conceived his sculpture by exploring the nature of the *Möbius' strip*, a one-sided non orientable surface obtained by cutting a closed band into a single strip, giving one of the two ends thus produced a half twist, and then reattaching the two ends (*Gray, 1997*).

For representing the infinity through the finitude of a strip and for the study of topology to conceive his art piece, *Max Bill (1908 - 1994)* was awarded in 1951 during the first edition of the *Bienal de São Paulo* with the *Prize for Sculpture*. Something that evidenced the collaboration between art and mathematics as a medium for creating relevant works and achieving interesting aesthetic results.

Discussing, more specifically, mathematical patterns in art, old Islamic artisans needed to be very skilled in analytical geometry in order to build the stunning ornamentation of repetitive patterns found in Islamic art. Actually, Islamic art is considered one of the earliest instances of art to use algorithmic foundations for its beautiful repetitive and complex patterns (Wever, 2014).

For being prohibited of using representations of people in holy sites, ancient Islamic artists incorporated mathematics, compass and ruler into their practice and developed a very recognizable aesthetic based on repeated geometrical constructions. The patterns are built by following a sequence of steps that perform geometric operations on a pre-established grid.



School of Islamic Geometric Design. Pattern 1.

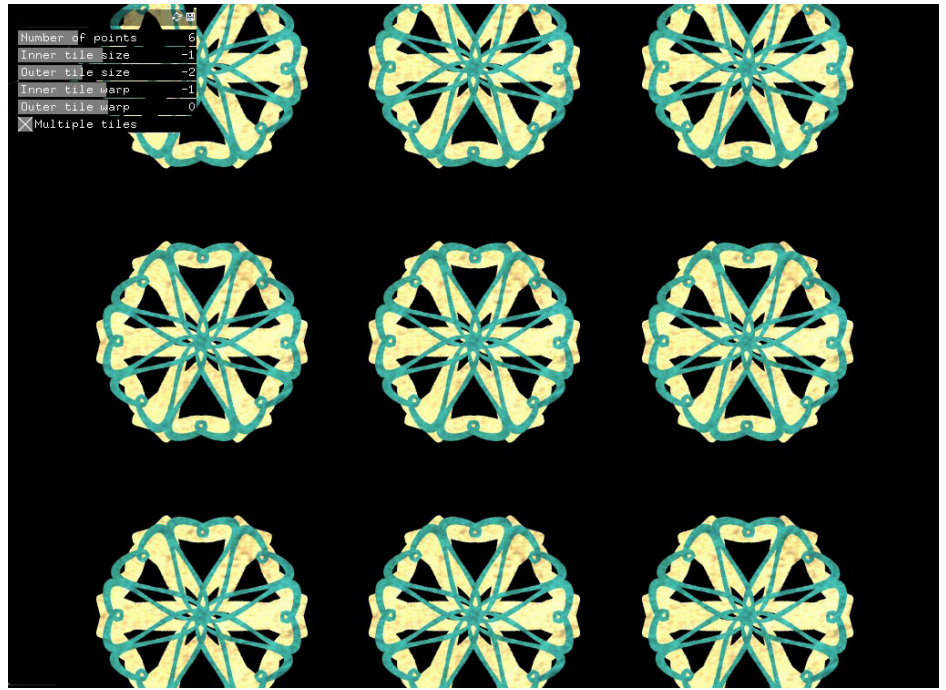
Steps for building this pattern:

- “1. Start with a circle in a square, divided into eight equal sections.
2. Draw four lines that pass through the intersections indicated with red circles. Consider the lines as two opposing V-shapes. The lines do not end in the corners of the square.
3. Draw another two opposing V-shapes, using the same intersections as in the previous step.
4. All the construction lines have now been drawn. Take a different colour pen or pencil and draw the red lines, tracing parts of lines you have drawn in the previous steps.
5. Still using a different colour pen or pencil, draw the four-pointed star, as indicated.
6. All the lines have been drawn, your pattern is complete.
7. Your pattern without the construction lines.
8. Now you can tessellate your pattern in a grid of squares to make a bigger composition.”

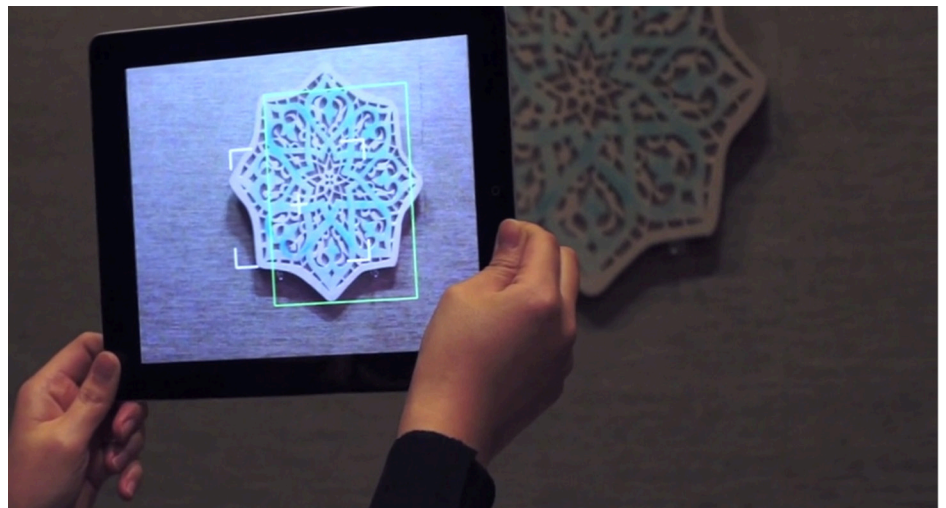
A good example for showing the relevance of the relationship between mathematics and art for computer art, and also the last example of this section, is a project developed by the *Metropolitan Museum of Art’s Media Lab* in 2014. The project’s purpose was to create an instigating augmented reality experience based on the Islamic art pieces the *MET* has in its collection.

The development team used *Processing* and *openFrameworks*, computer art tools discussed in this thesis to come up with an *iPad* augmented reality app that allowed visitors to explore how single tiles and incomplete fragments of these patterns would look like if they were covering an entire wall, a simulation of its original design. Also, the application allowed users to try different values for the equations and observe how these changes would affect the existing shapes and consequently generate new patterns.

MET Media Lab's Application. Screenshot. 2014
By changing the parameters at the top left of the screen, different shapes could be obtained.



Interacting with MET Media Lab's application.
This photograph taken by the developer and designer of the project, Betty Quinn, in 2014, shows a user interacting with the application.



The final result was an app that improved the museum's experience for visitors and also an opportunity for the development team to work with modern technology to enhance classic works of art.

The *MET Media Lab's* project shares our goals of understanding art and its natural relationship with mathematics, explore media technologies and their impacts on users when conceiving a work of art and using the computer as a medium for creating new and immersive artistic experiences.

The Development of Computer Graphics as a Supplier of Aesthetic Relevance for Computer Art



Ivan Sutherland using the Sketchpad. 1962.

Under a temporal and conceptual analysis, the use of the computer as a medium for artistic creation had evolved in parallel with the efforts of researchers, engineers and companies in computer graphics. As a new discipline and subarea within computer science, computer graphics had evolved from the exhibition of simple points on the mainframes equipped with cathode ray tubes to the creation of sophisticated videogames and other computer generated graphics.

The *Whirlwind* was a computer created by the MIT in 1950 and also one of the first devices built with computer graphics technology. The machine was equipped with a CRT screen dedicated to displaying images that later could be photographed so that they became images in paper. The *Whirlwind* also was equipped with a light pen that allowed users to interact, even in a simple way, with the picture displayed.

The *Whirlwind* represents a first moment for computer graphics in which the predominant machines were extremely expensive and slow, besides generating simple and aesthetically primitive images; basically, plots with numerical data points in it. Until then, it seemed very unlikely that the computer would gain the relevance it has today for art and entertainment.

It was also in the 1950s that the US government announced the *SAGE*, an air defense system in which missiles and aircraft were detected by radar and quickly had their positions displayed on a screen. After this, the operators could point the light pen to some of the projectiles displayed as an input for the computer to calculate their range and intercept routes. The *SAGE* system was one of the first developed with interactive computer graphics technologies (*Jankel; Morton; Leach, 1984*).

In 1962, *Ivan Sutherland (1938)* attracted the attention of the scientific community by writing a doctoral thesis in which he introduced a first methodology for studying computer graphics and formally placed the new discipline within computer science. In his thesis “*Sketchpad: a man-machine graphical communication system*” (MIT, 1962), the so-called “father of real-time computer graphics” proposed a system in which the user could draw shapes with a light pen and then modify them by using the keyboard.

Ivan Sutherland's Sketchpad is a milestone for both computer graphics and human-computer interaction disciplines for its extremely innovative character and for giving rise to modern *GUIs* (graphical user interfaces) and object-oriented programming languages. The project ended up putting computer graphics on another level of research and investment, and in 1988 he received the *Turing Award*, one of the most important awards in computer science.

First computer graphics works and publications called the attention of scientists that, during the 1960s, started to explore this technology. A famous piece of computer graphics and also one responsible for popularizing the new discipline within the scientific community was the movie “*Simulation of a two-giro gravity attitude control system*” (1963) created by *E. E. Zajac* (1926 - 2011). The scientist used the *IBM 7090* mainframe to produce the animation of an artificial satellite orbiting Earth and to illustrate how this satellite's altitude would vary as it orbits the planet.

Zajac's project managed not only to popularize computer graphics as a discipline, but also to showcase the power of vector displays; paradigm of representation and creation of forms that prevailed until half of the 1970s, when displays based on the rasterization of pixels took their place. Vector displays would draw shapes by defining the curvature, start and end points of each line that composes the final graphic, while in modern displays the computer would draw one pixel at a time; a procedure that demands more computational power but also results in more significant colors and shapes.



Frame from E. E. Zajac's *Simulation of a two-giro gravity attitude control system*. 1963.

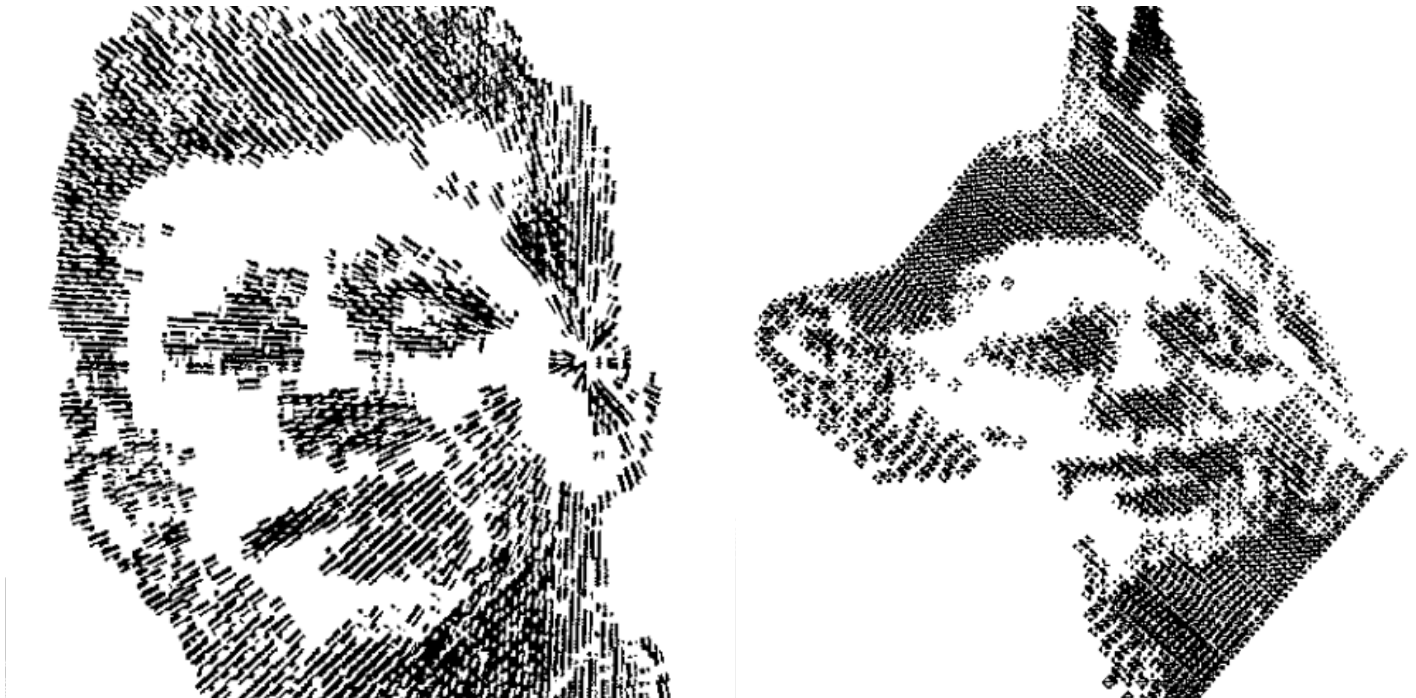
In the latter half of the 1960s and driven by recent improvements in computer graphics technologies, computer art began to gain relevance in the art world; the computer itself had become irresistible and imbued with its own complex cultural significance. Except for some specific innovative exhibitions, at first many art galleries began to exhibit computer art but under a “secondary” character and “without many noble aspirations”. (Taylor, 2014).

One of the most important events for spreading computer-aided creative pieces was *Cybernetic Serendipity*, an exhibition dedicated for computer art that happened at the *Institute of Contemporary Arts of London* in 1968. Although encompassing not only computer graphics but also music, poetry, dance and animation; the exhibition is significant for this study for being a landmark for the dissemination of the artistic qualities of the computer to the world of arts.

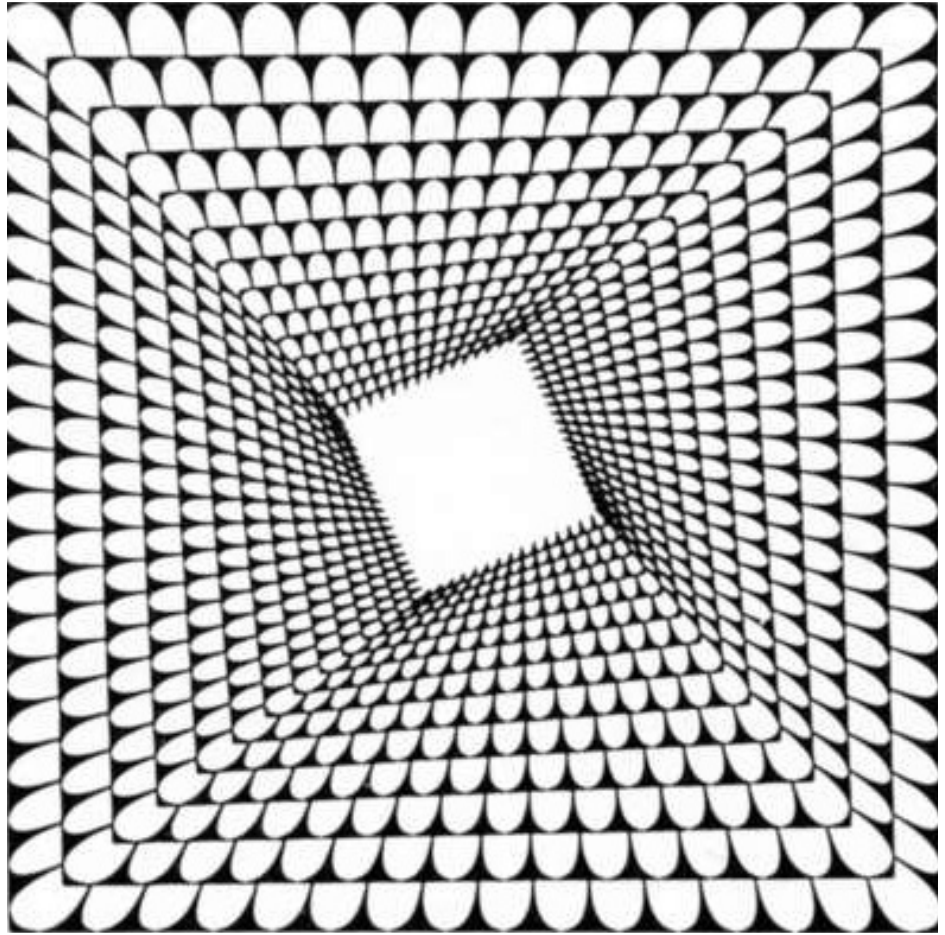
One example of work exhibited during *Cybernetic Serendipity* and also a representant of first aesthetic results was a series of computer-made deformations applied to a photograph of *John F. Kennedy*. The series was created by the *Computer Technique Group from Japan* with the help of *IBM* and pieces were developed in an *IBM 7090* using *FORTRAN IV* (Reichardt, 1968).

Both *Shot Kennedy No. 1* and *Kennedy in a Dog* (pictures below) were created by scanning an original photograph and then subjecting it to different processes of deformation. *Shot Kennedy No. 1* was created by converting data from the photograph into straight lines converging at one point at the ear, while in *Kennedy in a Dog* the same data would be inserted into data from a photograph of a dog (Reichardt, 1968).

The Computer Technique Group from Japan. Shot Kennedy No. 1 and Kennedy in a Dog. 1968.



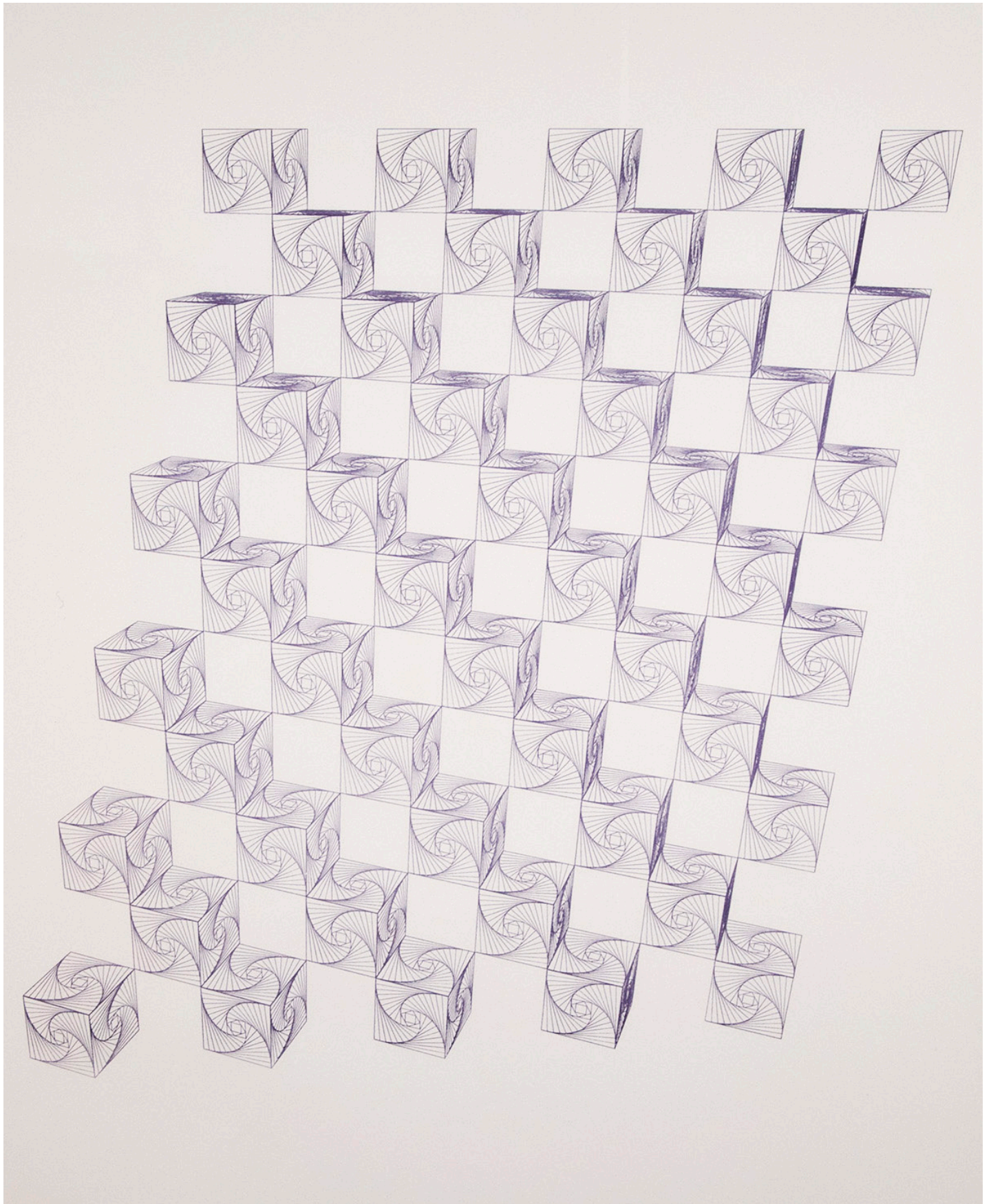
Donald K. Robbins. Variation on a theme from Jeffrey Steele. 1968.



For the same exhibition, *Donald K. Robbins* developed computer art pieces based on the “bug problem” to come up with instigating patterns. This problem stands for placing four bugs on the corners of a square and observing which path they would follow if “asked” to crawl toward each other. The spiral-looking pictures below were obtained by taking a picture of these bug’s lines of vision periodically, and drawing lines to indicate this (*Reichardt, 1968*).

Donald K. Robbins reached an even more interesting result by recursively calling each of the bugs’ paths as different subroutines. The checkerboard on the next page was obtained by using this recursive method and a posterior shape-distortion to create an instigating three-dimensional pattern.

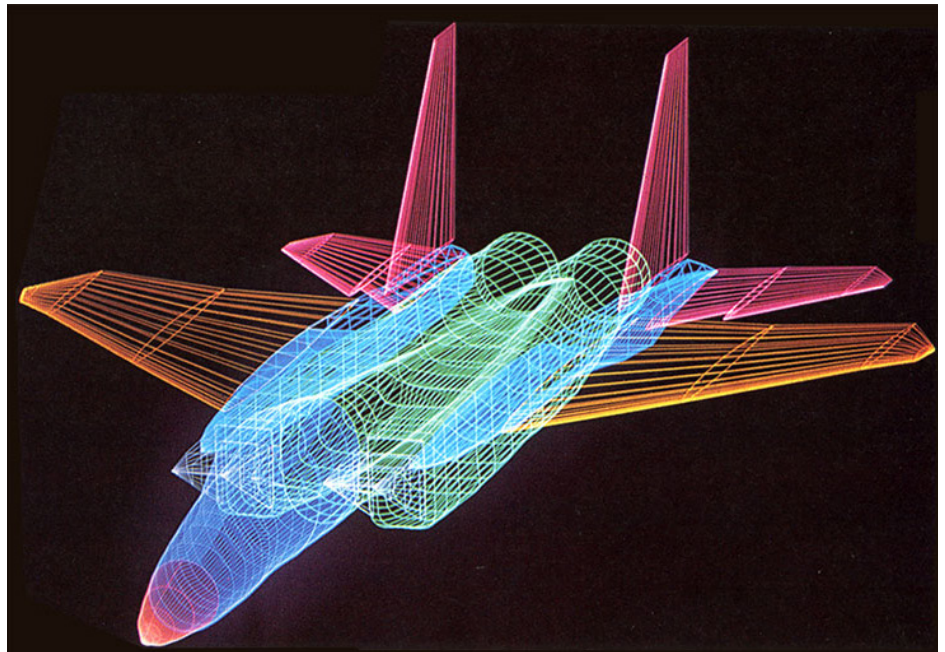
Next page:
Donald K. Robbins. Checkerboard pattern. 1968.



Right:
Wireframe of an aircraft.
Created with the Evans & Sutherland original
picture system.

Below:
A still from John Whitney's *Matrix III*. 1972.

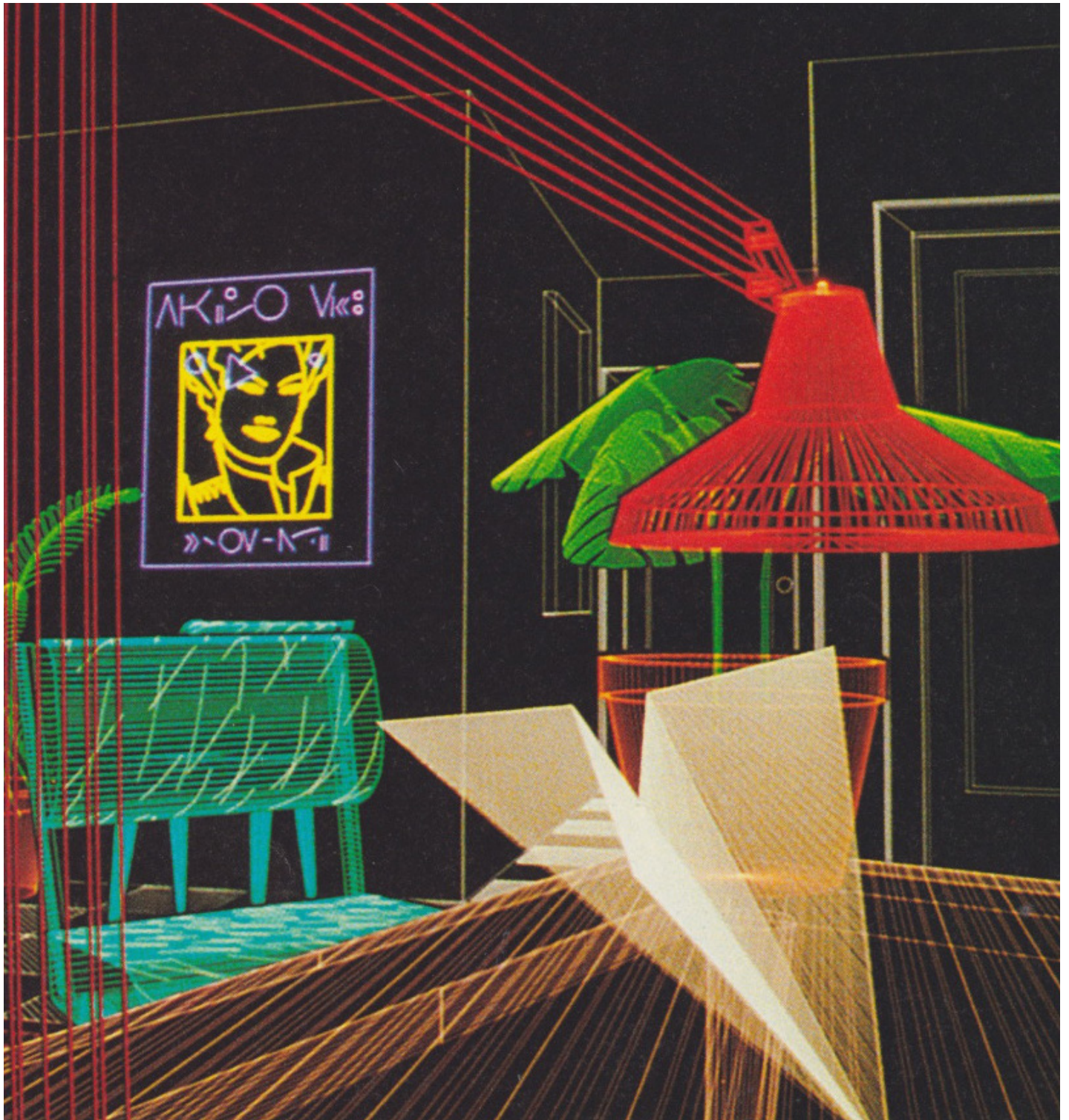
Bottom:
A still from Lillian Schwartz's *Olympiad*. 1971.



During the 1970s, research projects in computer graphics were viewed with great enthusiasm and high expectations. Most of the field's formal methodologies were developed during this time, and multidisciplinary efforts were undertaken in order to bring computationally generated graphics from the laboratory to television and other mass media.

The linear nature of still-prevalent vector displays was largely absorbed and applied by the industry in the form of creating wireframes of buildings, aircraft and cars. Many investments have been made in the increasingly sophisticated modeling of computer graphics as a more practical and quick way to foresee the final products.

Within arts, the vectorial paradigm manifested itself through the geometrical aesthetics present in the works of several of the artists who started experimenting with computer graphics in the period. Works of artists such as *John Whitney* (1917 - 1995) and *Lillian Schwartz* express the predominance of simple shapes and essentially geometric explorations of rotations, distortions, overlaps, and even prematurely, tri-dimensionality.



Robert Abel & Associates. A Panasonic "Glider" Commercial. 1981.

In the late 1970s, *Thomas A. DeFanti (1948)* developed the *GRASS* and *ZGRASS* programming languages, both very popular among artists. *GRASS* was created as a programming language to script 2D vector graphics animations and its syntax was very similar to *BASIC* but added many instructions for working with 2D object animation. *ZGRASS* was an evolution of *GRASS* that supported raster graphics.

GRASS and *ZGRASS* built-in functions included translation, scaling, rotation and color picking over time; a more visual approach to programming that allowed developers to focus on the visual results rather than technical specificities. For facilitating the programming of works with interesting aesthetic results and allowing even less technical people to develop their ideas, both languages have become a hit in the artistic world (*Jankel; Morton; Leach, 1984*).

One of the most famous works developed using *GRASS* was *Larry Cuba's* animated *Death Star* showed on the first *Star Wars* movie. Another representative of the very linear nature of computer generated graphics made possible by vector displays.

Right:
Equipment used to create the Death Star for the 1977 "Star Wars" movie.

Bottom:
Computer graphics created by Larry Cuba for the first Star Wars movie. 1977.



During the late 1970s and 1980s, three-dimensional rendering technologies started to become more accessible and efficient while many improvements were being developed by researchers and animation companies. It was in 1978, for instance, that *James Blinn* introduced a new mapping technique called *Bump Mapping* in which surfaces would look more realistic by simulating small displacements on it.

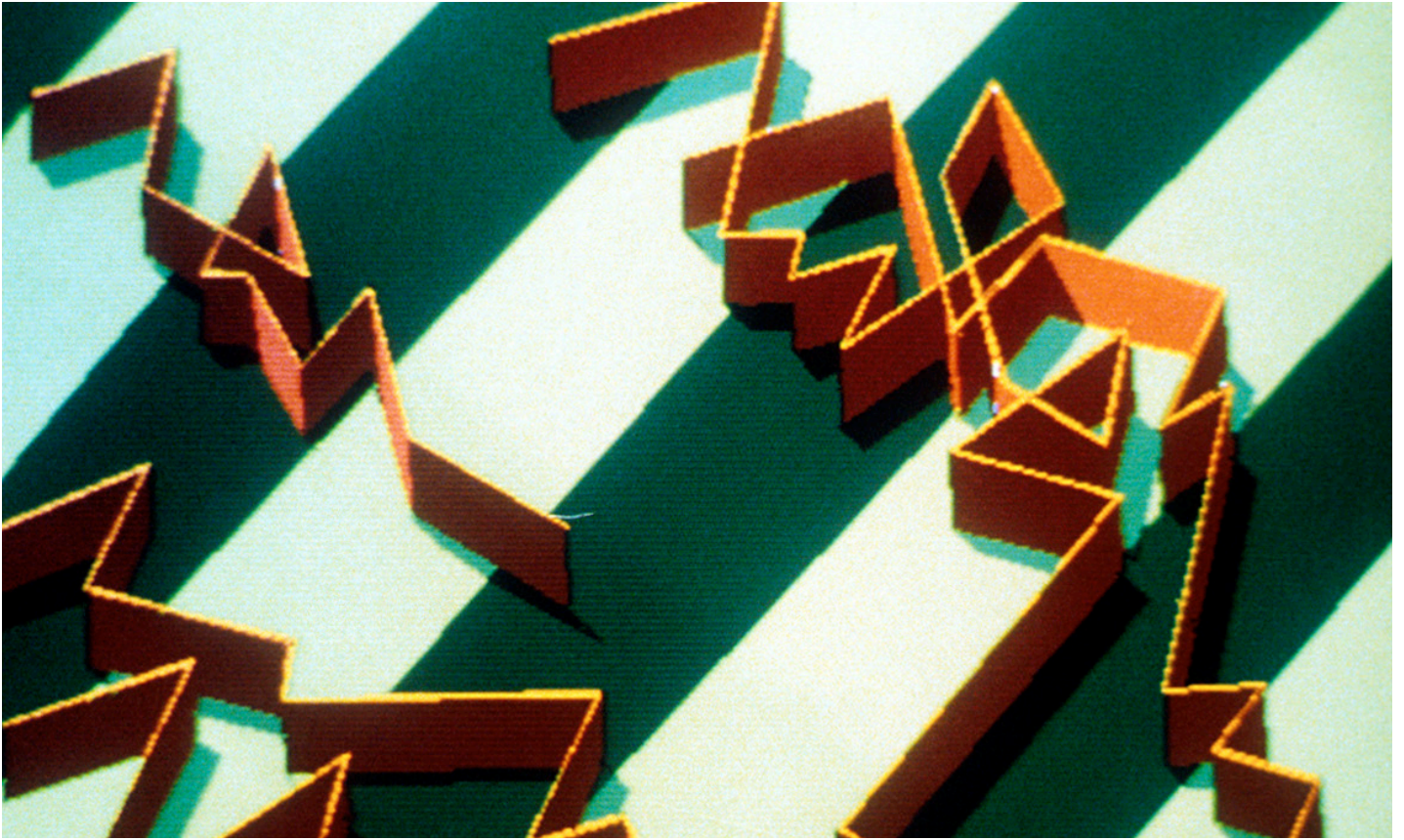
Three-dimensional graphics have come to occupy a position of great prominence among artists and computer graphics practitioners and, therefore many subsequent works of art relied on this new trendy three-dimensional aesthetics. It was in the 80s as well that computer graphics acquired a prominent position outside the laboratories and studios, calling people's attention.

All of the following works displayed are examples of ones with a three-dimensional appeal and also ones that were accepted for exhibitions in the form of works of art during the 1980s.

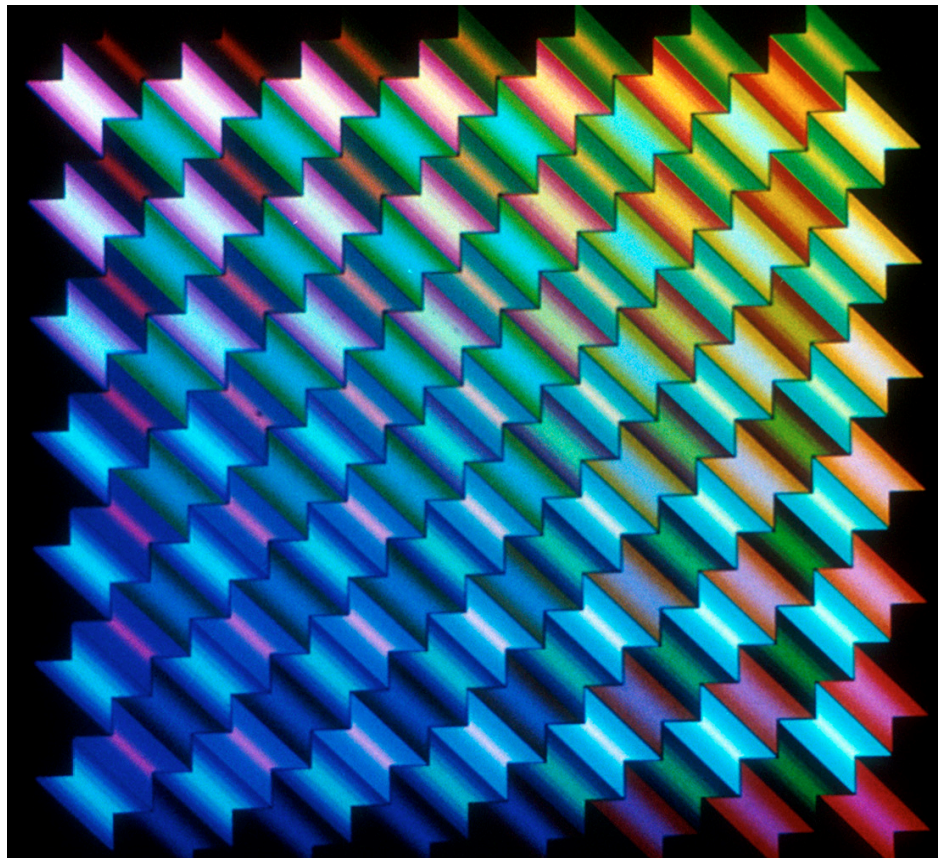


John Whitney and Gary Demos. Art gallery scene with geometric primitives. 1981.

As 3D graphics became a hit and as more efforts were being made in literally simulating the real world (or abstractions of it), the understanding of perspective, laws of nature, color theory and proportion became mandatory. *John Whitney's* gallery is an example of artwork that relied on perspective and proportion to create an environment.



Harry Holland. Santy Fold. 1984.



Paul Jablonka. Mural. 1984.

Computer graphics continued to advance, and with that, several possibilities arose for the artists who would come to work with the computer. From the 1990s, it became clear that the origin of interactive devices, sensors and actuators; the improvement of graphic interfaces and commercial software for artistic creation; the origin of many programming languages and techniques; and the web revolution itself would offer many different paths for the whole artistic community.

The very term “computer art” acquired an old-fashioned character, as specific techniques of computer graphics and artistic creation with the computer resulted in works of art very different from each other. There emerged, then, denominations that would better classify works of art and, at the same time, function as subareas for digital art creation.

COMPUTER ART

As stated by *Frank Popper (1918)*, there are many influences for the beginning of the efforts in computer and virtual art during the last century. In his book “*From Technological to Virtual Art*”, the historian of art and technology and *Professor Emeritus of Aesthetics and the Science of Art at the University of Paris VIII* discusses the various art movements of the early twentieth century that influenced a growing class of artists who were beginning their experiments with the computer.

According to *Popper*, the luminous aspects of kinetic art; the effervescence of cinema and animation as experimental mediums; the Dadaist and Surrealist movements and *Pop Art* contributed with different aesthetic and semantic characteristics that shaped the debut of making art with the computer.

The movement had a troubled beginning as an art discipline due to resistance from conservative artists and art critics who questioned its relevance and legitimacy as an artistic practice. It was argued that the predominantly technological and scientific focus of first publications in computer art, as well as the difficulty of establishing methodologies and definitions for the practice, placed computationally produced works in a category of “non-art” that would not find space in exhibitions and competitions (*Taylor, 2014*).

The very multidisciplinary nature of making art with the computer made it hard for the discipline to be defined and placed among artists. From the earliest times, the scene is composed of artists, engineers, designers, computer scientists and mathematicians who, working together or apart, have different perspectives on the work to be developed and on computer art as a whole.

On its way to becoming the more innovative and experimental medium, the computer was employed by a range of artistic practices that included the visual arts, cinema, choreography, literature and music (*Taylor, 2014*). And, despite having a greater concentration in the visual arts (focus of this work), the term “computer art” has been used, since its origin, in the most varied contexts.

Computer art went through several transformations of aesthetics and meaning throughout the twentieth century. While some artists have explored intensely the creation of artworks generated by algorithms, others have appropriated multimedia and internet technologies incorporating interactivity to their performances. With the advent of personal computing and significant improvements in graphical interfaces, another group has also emerged: commercial software users artists.

From the 90s, the very term “computer art” acquired a nostalgic and old-fashioned character. The term has began to be used, almost exclusively, to describe the initial phase of artistic experiments with the computer, going back to a time of pioneering in which equipment was old fashioned and the aesthetics of the results obtained was extremely simple.

Artistic experiments happening in digital mediums assumed a distributed scope, something that also contributed for the previous term to be left behind. From the 90s, artworks did not necessarily occur on a single computer; it became possible to design an artwork that could be processed by several cores, with several monitors, on mobile devices and through the internet. It also increased interest in sensors and other technologies that added interactivity to the performances.

The term “computer art” gave place to many other denominations that, besides describing “subareas”, sought to define more assertively the artistic practice in question, such as “generative art”, “net art”, “software art” and “algorithmic art”. It is appropriate to say that this work has a greater focus on algorithmic and generative art, types of digital arts in which works are generated by algorithms and programming techniques.

Computer Art Pioneers



Top:
Georg Nees in 1986.
Photo by Alex Kempkens.

Above:
Max Bense in 1964.
Photo by Goebel Weyne.

Right:
Rot 19 cover.

Georg Nees (1926-2016) and *Frieder Nake* (1938) are both scientists most famous for being the first ones to put up shows of computer art during the early 1960s. Both belong to a group of artists that today is called “The Algorists”, artists who employ original algorithms in the process of creating their art.

Nees is a German mathematician and physicist who, then, became a doctoral student of philosophy at the *University of Stuttgart*. He is also credited for being the first person worldwide to publicly exhibit computer art pieces, partly due to the contact with other scientists, artists and professors who were investigating aesthetics and innovative ways of creating art during the 1960s.

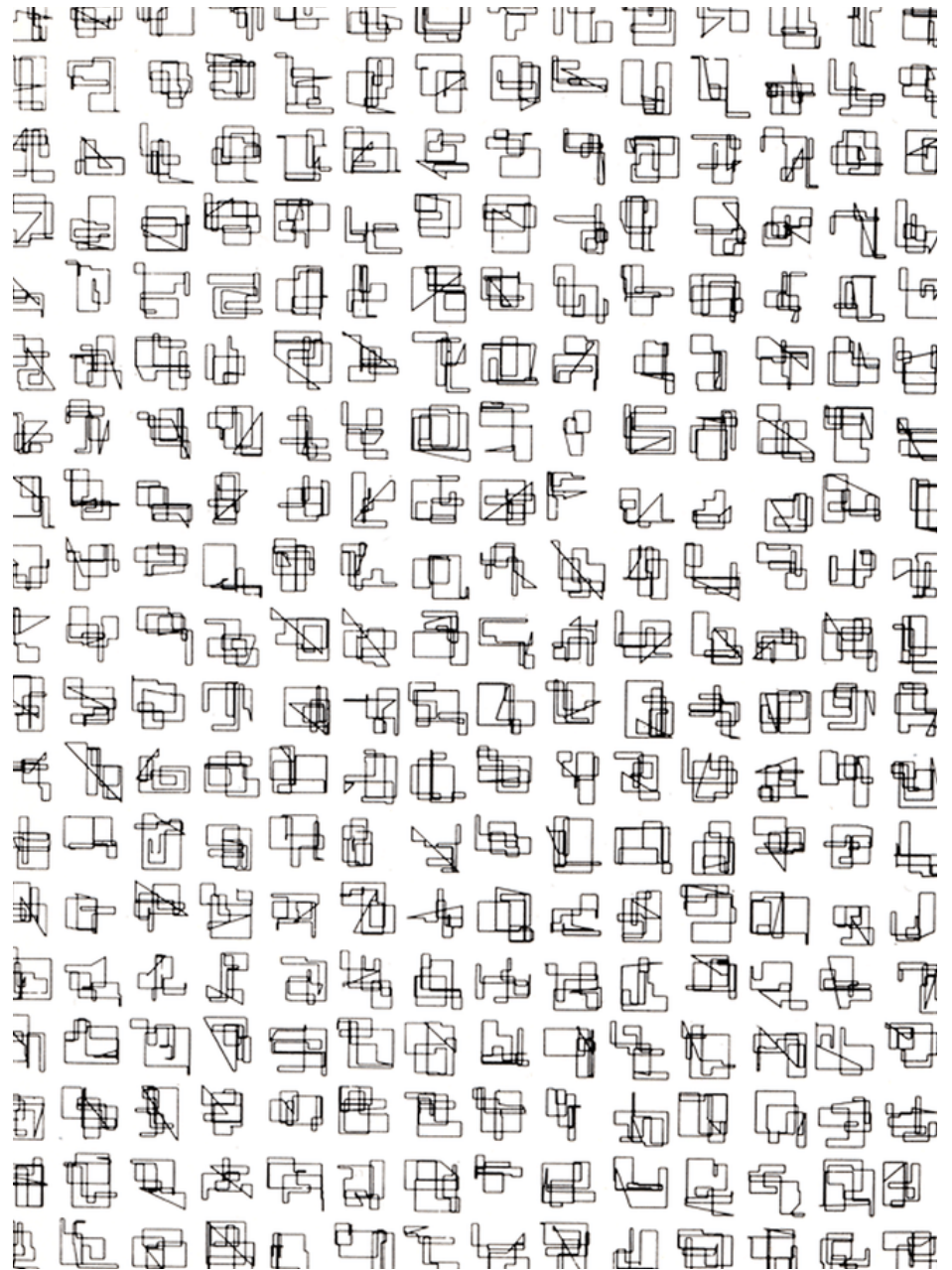
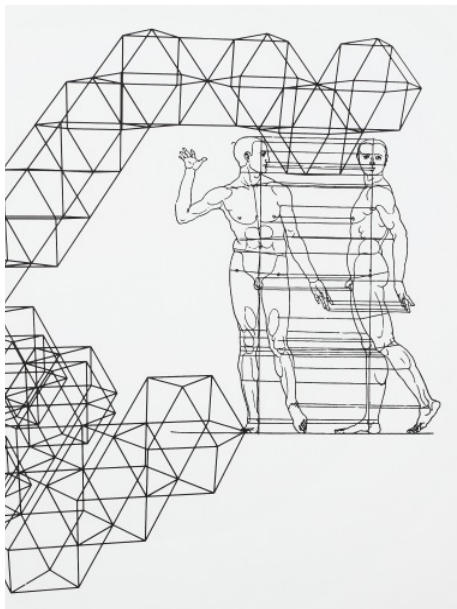
The German philosopher and writer *Max Bense* (1910-1990) was *Nees*’ supervisor and also one of the responsables for introducing him to the artistic medium that would come to be called “computer art”. *Bense* is known for his work in philosophy of science, logic, aesthetics, and semiotics; and after getting in touch with *Nees*’ early computer graphics experiments in 1964, he decided to invite the mathematician and physicist to exhibit computer graphic works at his experimental gallery. At that time, *Bense*’s gallery was dedicated, mainly, to concrete art pieces such as text and graphics, a very rationalist approach to art that suited very well algorithmically generated drawings (*Compart*, 2018).

Nees and *Bense* were also the publishers of “*rot 19. Computer-Grafik*” (1965), a small booklet that is one of the first, if not the first, publications ever on computer art. *Rot 19* contains many of the artworks *Nees* exhibited in the show at *Max Bense*’s gallery in 1965 accompanied by pseudo-code explanations of the algorithms used.

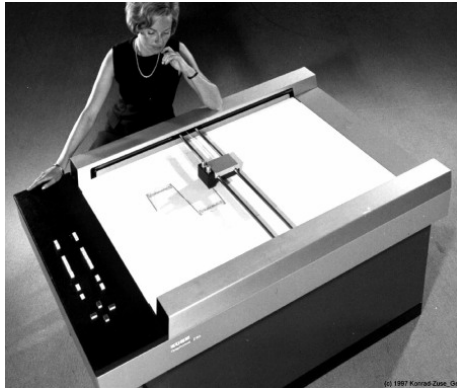


Georg Nees. 23-Ecke (Polygons of 23 Vertices). 1965.
This work was first published in the rot 19 booklet.

Georg Nees and Ludwig Rase. Kubo-Oktaeder.
1971.
Photo by Boris Cvjetanovi.

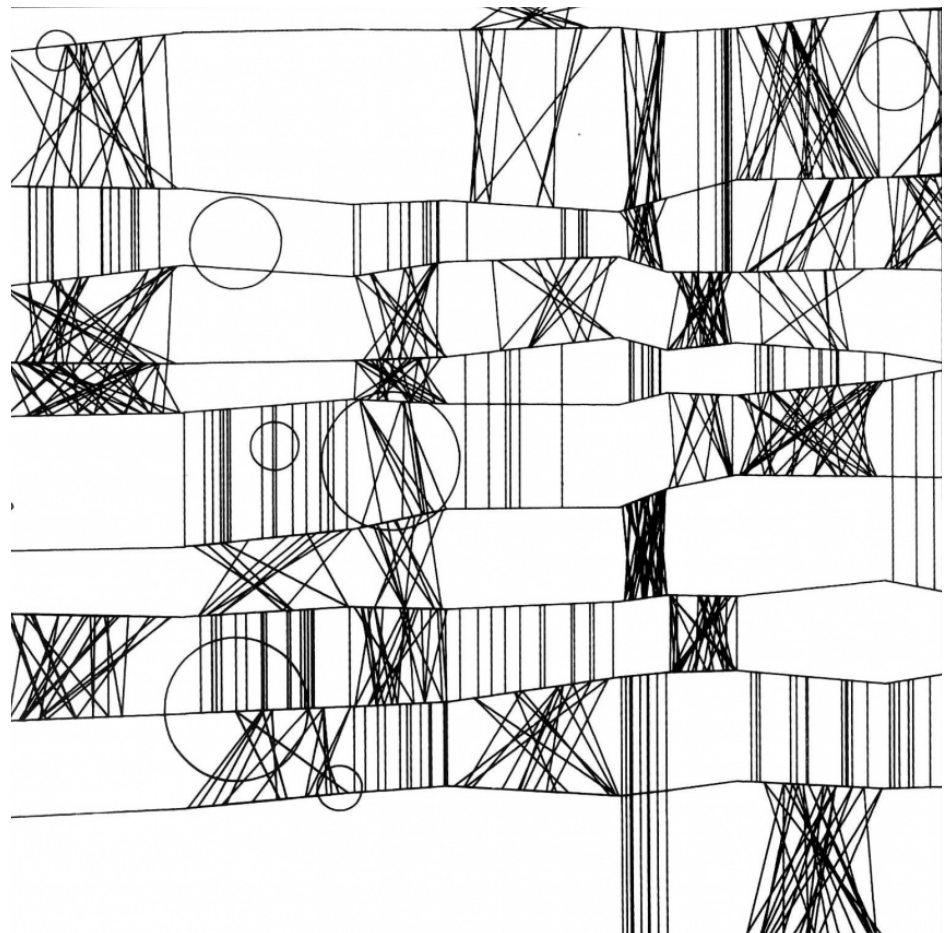


In 1965, and also influenced by *Max Bense's* works and publications, the German mathematician *Frieder Nake* prepared himself for his first computer art exhibition at the *Galerie Wendelin Niedlich* in Stuttgart. At that occasion, *Bense* was invited to talk at the opening of the exhibition and ended up suggesting that some of *Nees's* works should also be presented. That was the first time *Nake* and *Nees* got together as computer art pioneers (*Compart, 2018*).

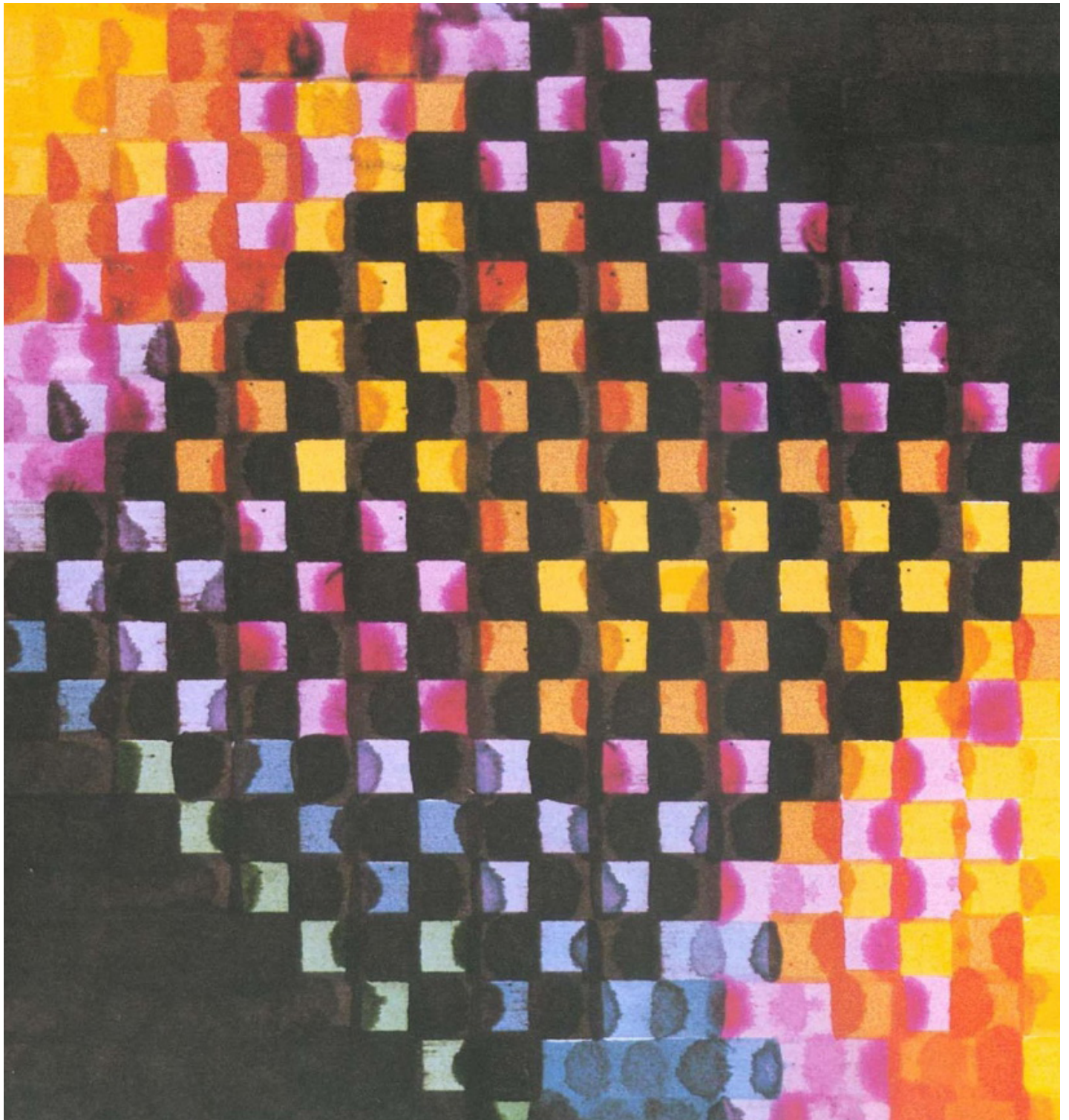


Top:
A photo from the Graphomat Z64.

Right:
Frieder Nake. Nr. 2 (Hommage to Paul Klee). 1965.



As an artist, *Frieder Nake* became more famous for his colored computer drawings that explored the visual expression of series of matrix multiplications, imagery with undeniable artistic intention. He has contributed to all major exhibitions of computer art, including the remarkable *Cybernetic Serendipity* (1968) discussed in the first chapter of this work (*DAM*).



Frieder Nake. Polygon Drawings. 1965.



Vera Molnár in her atelier. 2017.
Photo by Galerie La Ligne, Zürich.

Vera Molnár (1924) is also among those at the forefront of computer art, she is a French artist with Hungarian origin. *Molnár* started her artistic path in 1947 at the *Faculty of Plastic Arts* in Budapest, where she was trained as a traditional painter and established a style based on geometric abstractions.

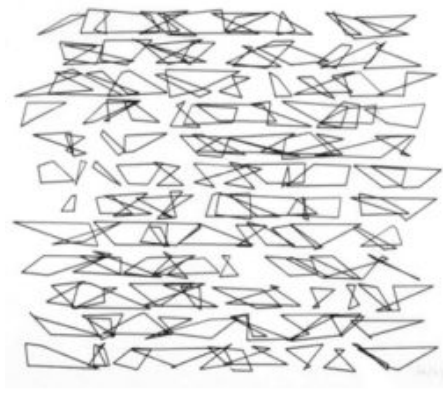
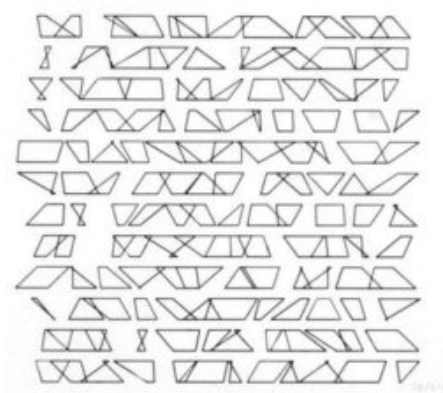
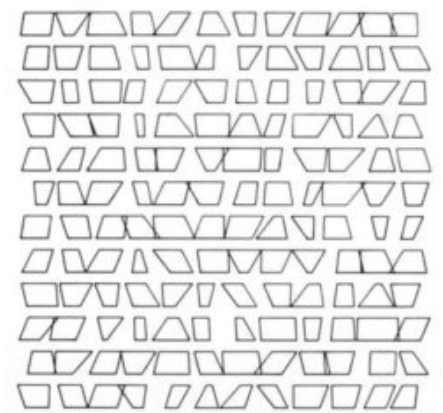
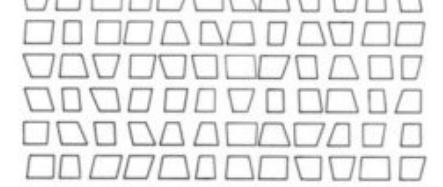
Following the geometric European trend during the 1950s, the artist composed her paintings with a symbolic vocabulary based on squares, rectangles, lines, circles and curves. At first, she explored the aesthetic results obtained by choosing and combining simple shapes and colors.

By later developing an intense reflection on the ways of artistic creation and the possible mechanisms, *Molnár* became interested in the practices of *Piet Mondrian* (1872 - 1944), *Kazimir Malevich* (1879 - 1935) and of the concretists; she, then, approached the scientific community and in particular, the mathematicians.

Her contact with mathematical logic and with the study of shapes and space led her to incorporate patterns into her works and also to develop an iterative mode of creation which, in her own words, consist of “a series of small probing steps, altering the dimensions, the proportions and number of elements, their density and their form, one by one in a systematic way in order to guess what kind of formal modification challenges the change in the perception of my picture: perception being the basis of aesthetic reaction.” (*Molnár*, 1975).



Vera Molnár. 2 Rectangles. 1949.



Molnár's creative process is essentially an algorithmic one and also very natural for those with a greater affinity with computational techniques. It is important to point out that increasing accessibility and mastery of computing by “non-scientists” is an extremely recent phenomenon. *Vera Molnár*, therefore, stands out for studying computer science since the beginning of her trajectory as a computer artist in the 1960s.

Iterative procedures are extremely laborious and tedious when done manually. Also as a negative point, when using the hands it's possible to obtain loss of regularity and consistency in the design of the forms, mostly straight. In her search for mechanical alternatives to her way of drawing, in 1968 the artist discovered the computer and the benefits it could bring to her creative process (*Molnár, 1975*).

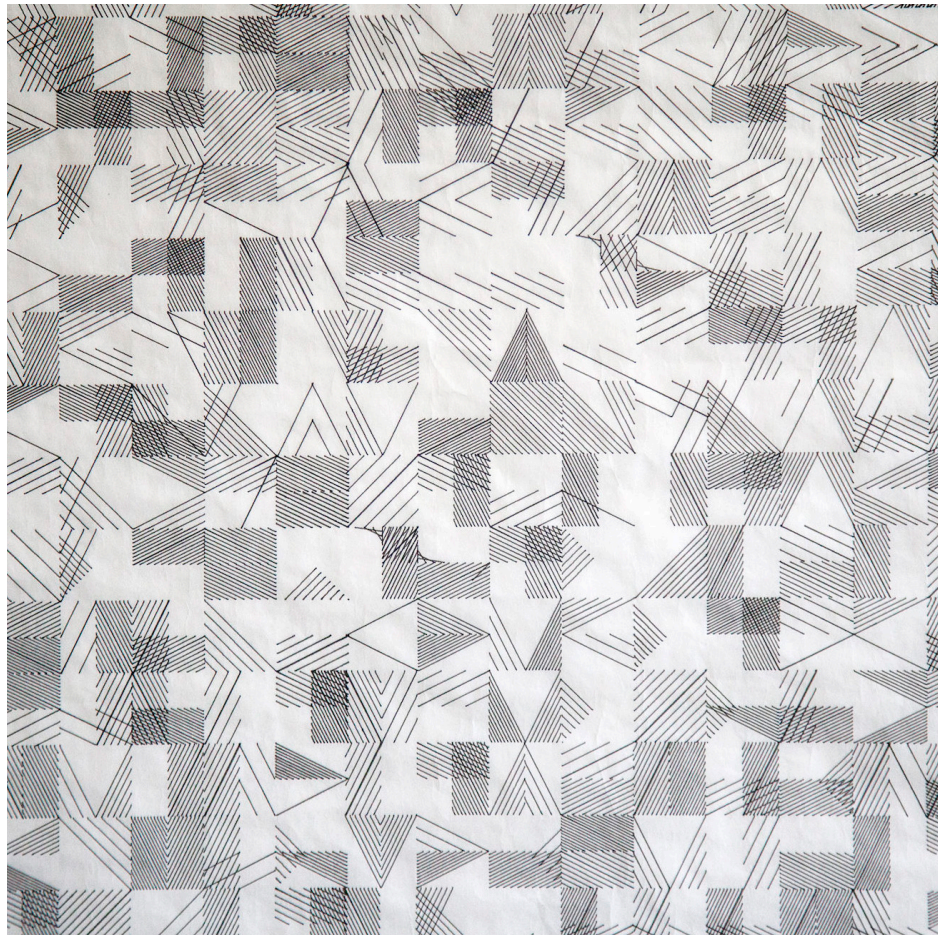
In the following passage, from a 1975 interview, *Molnár* refers to her creative process, and particularly to the work on the left.

“Using a computer with terminals like a plotter or/and a CRT screen, I have been able to minimize the effort required for this stepwise method of generating pictures. The samples of my work I give here in illustration were made interactively on a CRT screen with a program I call RESEAUTO. This program permits the production of drawings starting from an initial square array of like sets of concentric squares. The available variables are: the number of sets, the number of concentric squares within a set, the displacement of individual squares, the deformation of squares by changing angles and length of sides, the elimination of lines or entire figures, and the replacement of straight lines by segments of circles, parabolas, hyperbolas and sine curves. Thus, from the initial grid an enormous variety of different images can be obtained”.

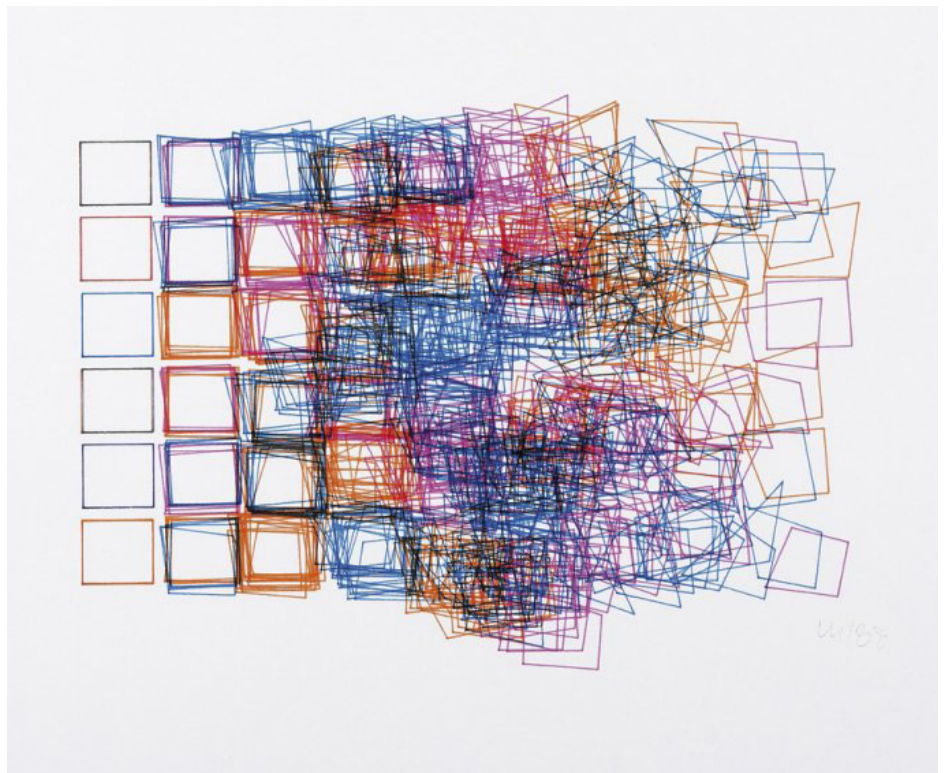
The artist argued early on that the computer could satisfy artists' desires for innovation as well as have it as an encouragement for the mind to work in ways other than conventional. She conducted studies and research on artistic creation with *Fortran* and *Basic* and, for her innovative work and positioning, she consolidated and helped founding the *Centre de Recherche d'Art Visuel* in Paris in 1960, a group that, among other studies, explored scientific and computational approaches for the arts.

Vera Molnár. 144 trapezes. 1975.

Vera Molnár. A la Recherche de Paul Klee
(Searching for Paul Klee). 1970.



Vera Molnár. Structure de quadrilatères. 1988.





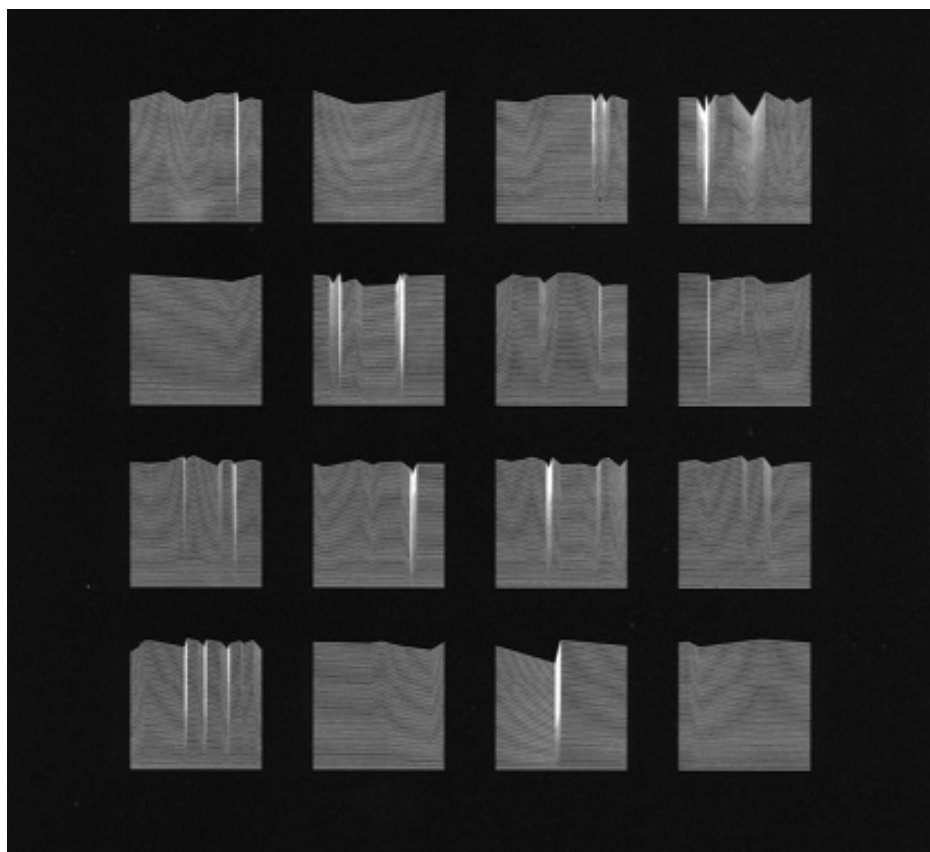
Manfred Mohr in 2011.
Photo by Bitforms Gallery.

Manfred Mohr (1938) is another artist considered a pioneer in computer art. Born in Germany, *Mohr* has a style based on computationally generated geometric elements. The artist began his *Fortran* programming studies in the late 1960s, and has since then the computer as an intellectual and physical extension of his creative processes.

At the beginning of his journey as an artist, *Mohr* adopted an abstract Expressionist style, which, under the influence of researchers such as *Max Bense*, gave way to computationally generated algorithmic geometry. The artist has had his works exhibited in several museums around the world, such as the *ARC - Musee d'Art Moderne de la Ville de Paris*, the *Josef Albers Museum* and the *MoMA - Museum of Modern Art in New York*.

Mohr says that his artistic goal is achieved when a computationally generated and finalized work can be dissociated from its logical origin and yet exist convincingly as an independent abstract entity. An essentially algorithmic process, but in which the greater interest lies in its visual aesthetic results.

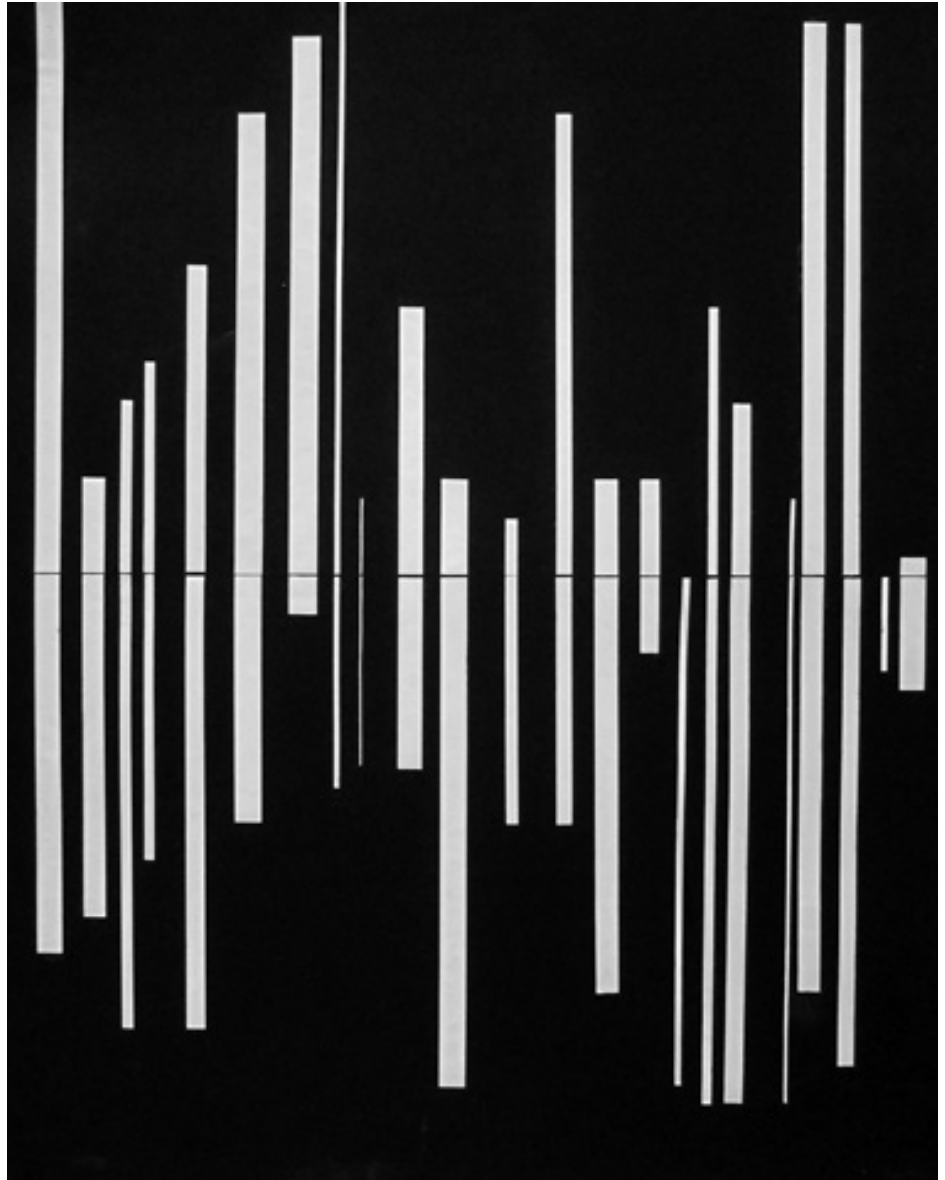
His oldest explorations with the computer are grouped and exhibited on his website (*www.emohr.com*) under the name of “early algorithms” (1969-1973), all of them accompanied by a brief explanation of the algorithm used. The collection highlights the aesthetics of the early phases of computer art and expresses the essentially geometric character, still in black and white.



Manfred Mohr. *Matrix Elements*. 1970.
Algorithm: “On each square of a matrix, a set of random points above a horizontal line are connected and then in defined steps linearly transformed to their positions on a horizontal line”
- Manfred Mohr.

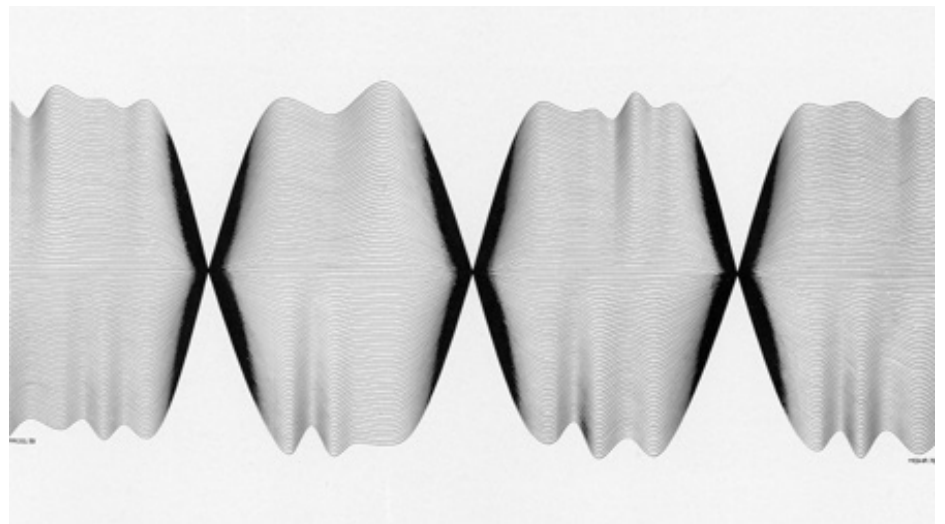
Manfred Mohr. Computer Generated Random Number Collage 1. 1969.

Algorithm: "Around a central line, random numbers determine the position, height, width, and existence of the rectangular white lines. This is a visual music collage, bringing to mind rhythm and frequencies" - Manfred Mohr.



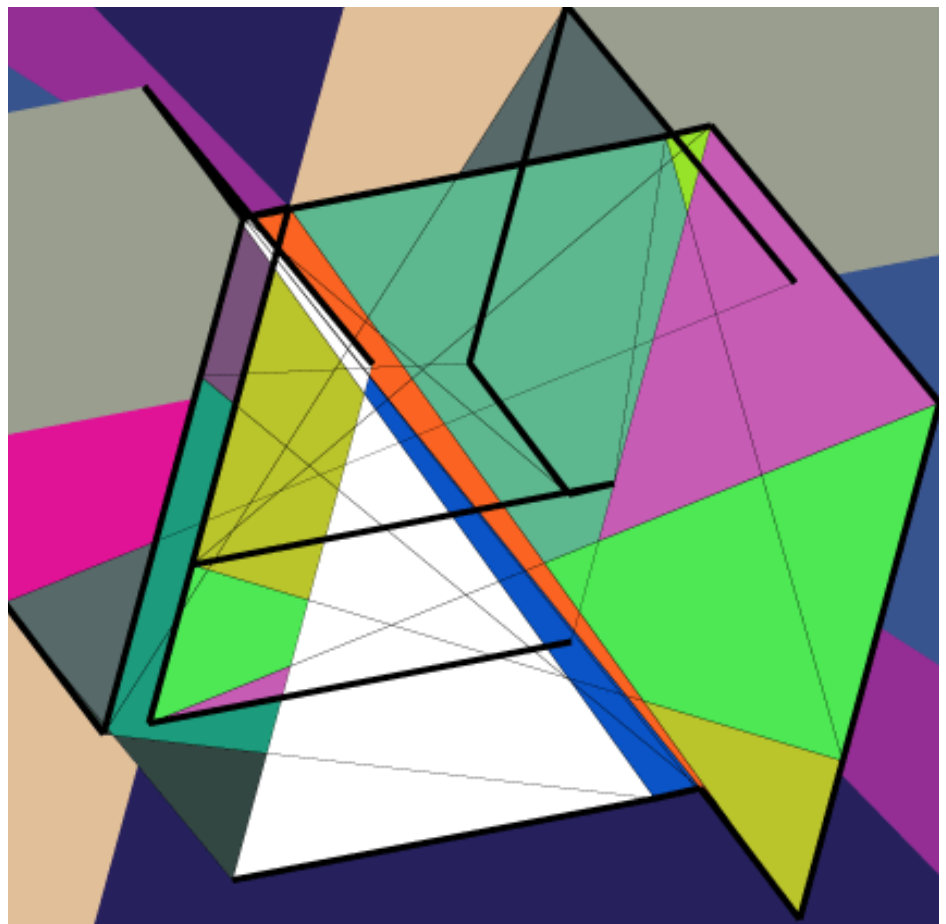
Manfred Mohr. n+ 3Hz. 1970.

Algorithm: "Two sets of inversed modulated sine curves are linearly transformed to the zero line. To form a modulated sine curve, points are randomly chosen within imposed limits of the outline of a sine curve and then interpolated into a curve by 3rd degree spline functions" - Manfred Mohr.



After working for almost three decades in black and white, *Manfred Mohr* started developing his most famous artwork: *space.color*. This work-phase is based on the 6-dimensional hypercube and its projections onto a bidimensional plan.

Mohr's algorithm explores the very geometric nature of the 6-dimensional hypercube: randomly choosing some of the structure's "diagonal-paths" and projecting them at a bidimensional plan. The artist defines the so-called "diagonal-path" as the connection between two endpoints of different diagonals of the hypercube and, since this complex structure has 32 diagonals there are 720 different "diagonal-paths". The colors are randomly set by the algorithm, which can originate a lot of different pictures depending on the "diagonal-paths" chosen.



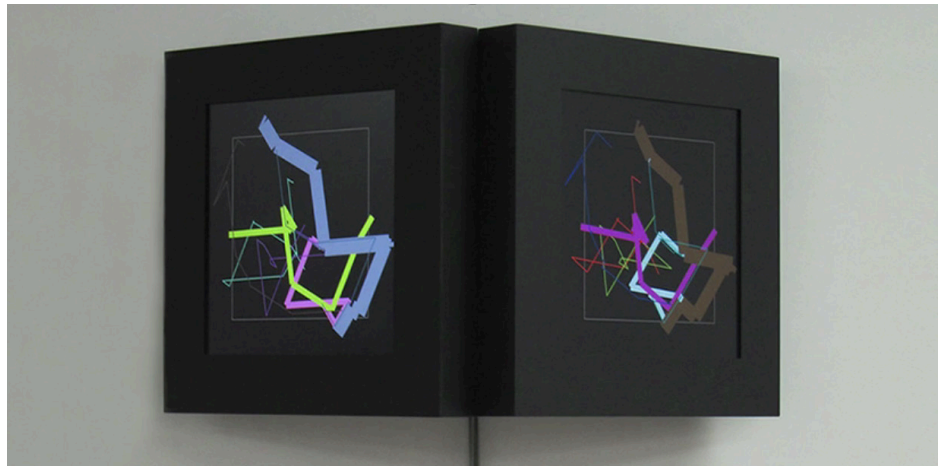
Manfred Mohr. *space.color*. 1999.

Almost 7 years later, *Mohr's* continued exploring the geometrical properties of the hypercube but, this time, with animations and a different color palette. *Klangfarben* (2006-2007) was one of *Mohr's* work-phases in which he explored the 11-dimensional hypercube following almost the same "diagonal-path" logic from before. Besides the still images, his *Klangfarben* consisted of two squared LCD-screens, a computer and the software he wrote.

Klangfarben installation view at Mueller-Roth Gallery, Stuttgart. 2007.

Algorithm: "From this repertoire four sets of eleven diagonal-paths with three distinct line widths are chosen as basic elements for each work. Every time this screen work is switched on, one out of the four sets are randomly chosen. The right screen shows a graphic construct consisting of 2 to 10 diagonal paths rotating in slow motion and all colors change randomly every 10 seconds. Single diagonal-paths fade in or out during the color changes in a cyclic but random order so that the back most diagonal-path always moves to the front. The last image, before each color change, is sent from the right screen to the left screen and stays there until the following image is received 10 seconds later."

- Manfred Mohr.



Mohr is an example of artist who explores the very geometric nature of structures. His artwork is strongly based on mathematics, aesthetically and algorithmically; and his art works as a good introductory inspiration for those who come from a very mathematical background and are looking for ways to understand how their knowledges and techniques can become art.

Lillian Schwartz (1927) is another computer art pioneer and also the last one mentioned on this section. The American artist is most recognized for working with computers to create graphics, films, animation, special effects, virtual reality and multimedia, a group of art techniques that exceeds visual art and aesthetically defined most of computer generated graphics on TV, movies and advertising during the 70s and 80s.

Right after the *World War II*, *Schwartz* began her artistic education by studying Chinese brushwork in Japan, something that aroused her interest and made her decide to study the fine arts in the following years. During her formal education path as an artist, she studied with fine art professionals such as the landscape painter and lithographer *Joe Jones (1909 - 1963)* and the painter *Giovanni Giannini (1930)* (*Schwartz, 2013*).

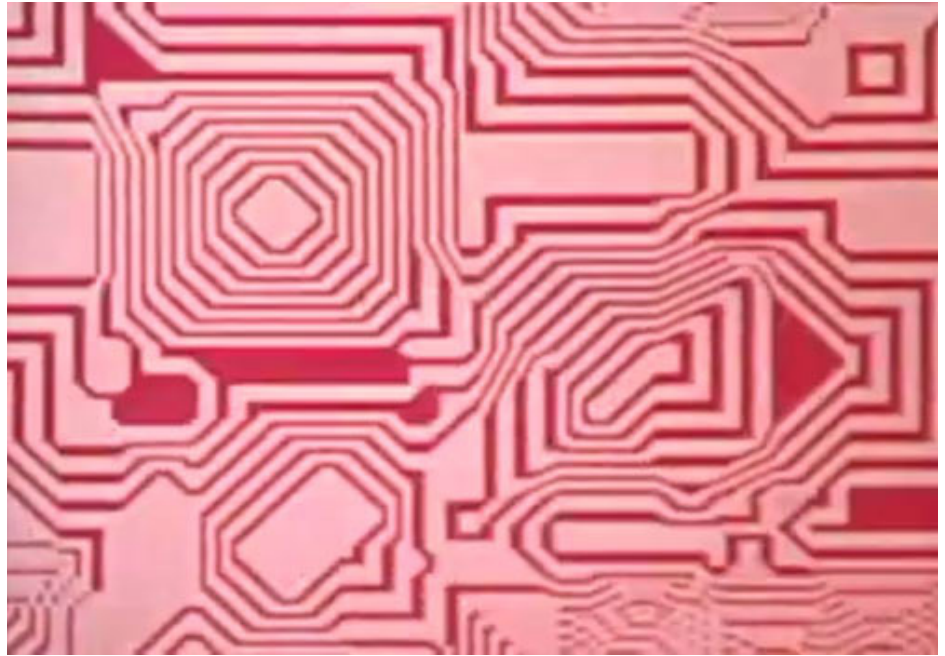
Even though she has been curious about different materials for artistic creation since young, her very experimental mind manifested itself more intensely later, in 1966, when she started working with mechanical devices like pumps and light boxes and even more when she joined the *Experiments in Art and Technology* group, a community of artists and engineers investigating the intersection between art and technology.

Schwartz started working with computers during the 60s as an attempt to meet her experimental desires of the moment and merge art and technology to come up with innovative creations. One of her first projects was a series of computer-animated films she produced when collaborating with the software engineer *Ken Knowlton*. The movies were produced by using outputs of visual art pieces that were created by generative algorithms written by *Knowlton* and edited by *Schwartz*.



Lillian Schwartz at Bell Labs.

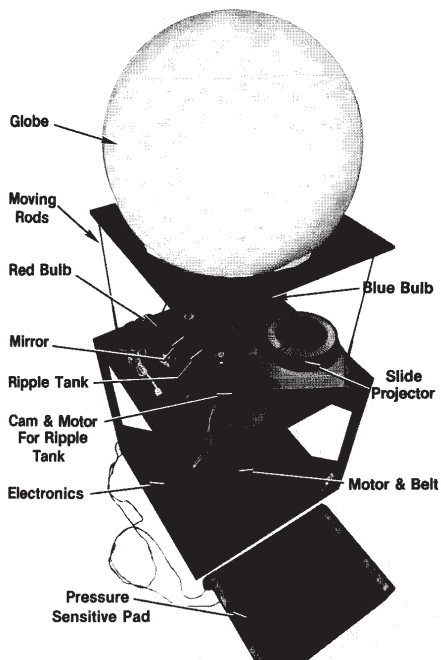
A still from the *Pixillation* movie, 1970.
One of the first products of the collaboration
between Schwartz and Knowlton.



Around that same time, *Schwartz* started taking programming classes to improve her repertoire of artistic creation techniques. At the beginning of her computer art experiments, she worked with the computer graphics languages *BEFLIX*, *EXPLOR* and *SYMBOLICS*, to build her own style of creating paintings and films originated by the combination of hand painting, digital collaging and computer image processing techniques.

Actually, *Schwartz* is a very relevant example of computer art pioneer to this project for showing other motivations and processes when creating art with the computer. Differently from the very algorithmic and rational nature of artworks created by *Molnár*, *Mohr*, *Nees* and *Nake*, *Lillian Schwartz* sought to employ the computer as an artistic medium beyond its procedural functioning. Even though the focus of this project is on algorithmic and generative art, when studying the creation of art pieces with the computer, one will find it relevant to understand *Schwartz* works as a way to promote the formation of aesthetic and semantic repertoire and as an alternative approach that can result in interesting pieces of art.

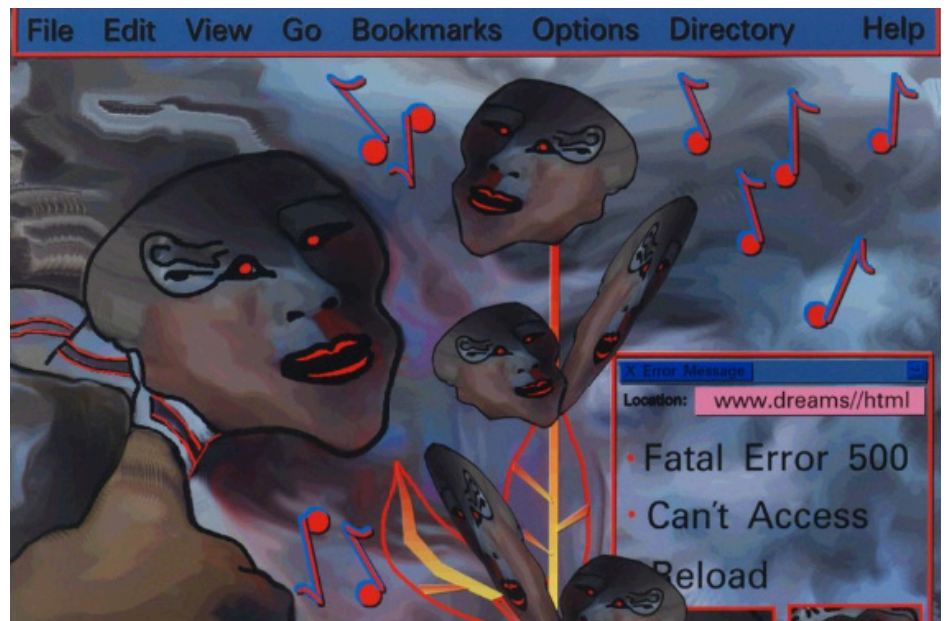
The artist was also interested in understanding the perception of color and sound to create interactive experiences for users to experiment and react. Recognizing her efforts, in 1968, *The Museum of Modern Art* selected her kinetic sculpture, *Proxima Centauri* for an exhibition. This sculpture consists in an interactive structure in which the observer could step on a pressure-sensitive pad and, depending on the movement and pressure, a dome in the top would generate vigorously dramatic effects with lights and colors (*Schwartz*, 2013).



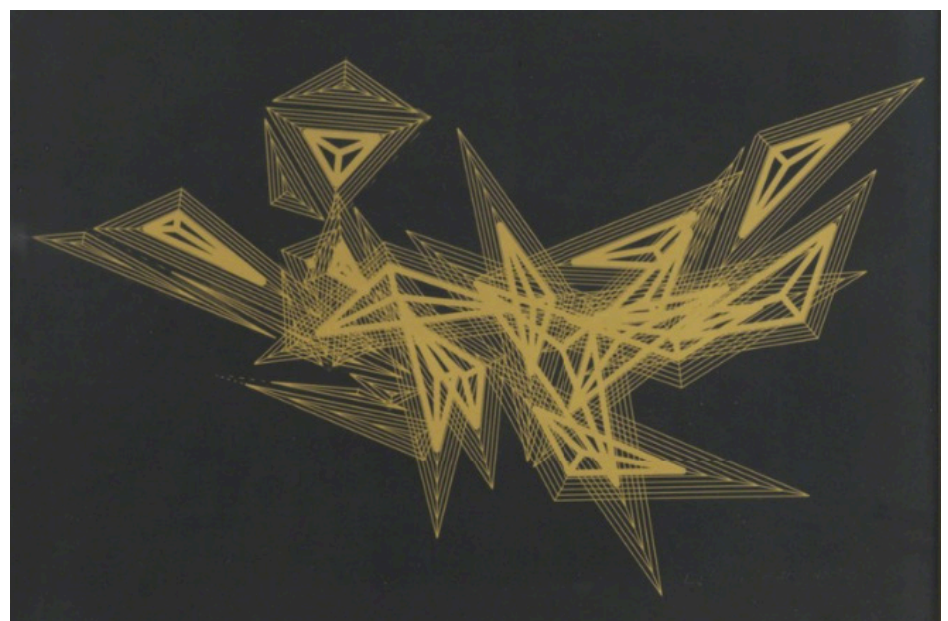
A paper prototype from Schwartz's *Proxima Centauri*.

Through her visionary and experimental projects, the artist placed herself in a very remarkable position within the computer art and computing community, something that allowed her to expand her work into the computer area when she acted as a consultant at the *AT&T Bell Laboratories*, *IBM's Thomas J. Watson Research Laboratory* and at *Lucent Technologies Bell Labs Innovations*.

During her artistic trajectory, *Schwartz* had many artworks selected for group and solo exhibitions at remarkable museums and art galleries such as the *Museum of Modern Art* in New York, the *Metropolitan Museum of Art* and the *Grand Palais Museum* in Paris.



Lillian Schwartz. *Dreams*. 1984.



Lillian Schwartz. *Charms*. 1970.

Generative and Algorithmic Art

Even though the terms “generative art” and “algorithmic art” have been in general use only since the 1960s, both describe concepts and artistic practices people have been using for a long time. A very famous example of generative art within music is *Mozart’s Musikalisches Würfelspiel (Musical Dice Game)* in which a minuet was composed by cutting and pasting together prewritten sections, making selections according to the roll of a dice (*Pearson, 2011*).

Another example of a generative approach for artistic creation would be a modified version of the instructions given on the first chapter of this project for building Islamic art patterns. Imagine if instead of simply following the series of geometrical operations prespecified to reach the final shape, one tosses a coin to randomly determine if the rule to be applied should be or not ignored. This procedure would result in an unexpected pattern.

Actually, the very practice of relying on some sort of instructions but also letting some randomness happen is the basis of generative art. The American philosopher and visual artist *Philip Galanter* defines this set of instructions as a “system” and establishes a very useful and complete definition for the artistic practice as whole:

“Generative art refers to any art practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art.” - *Philip Galanter*.

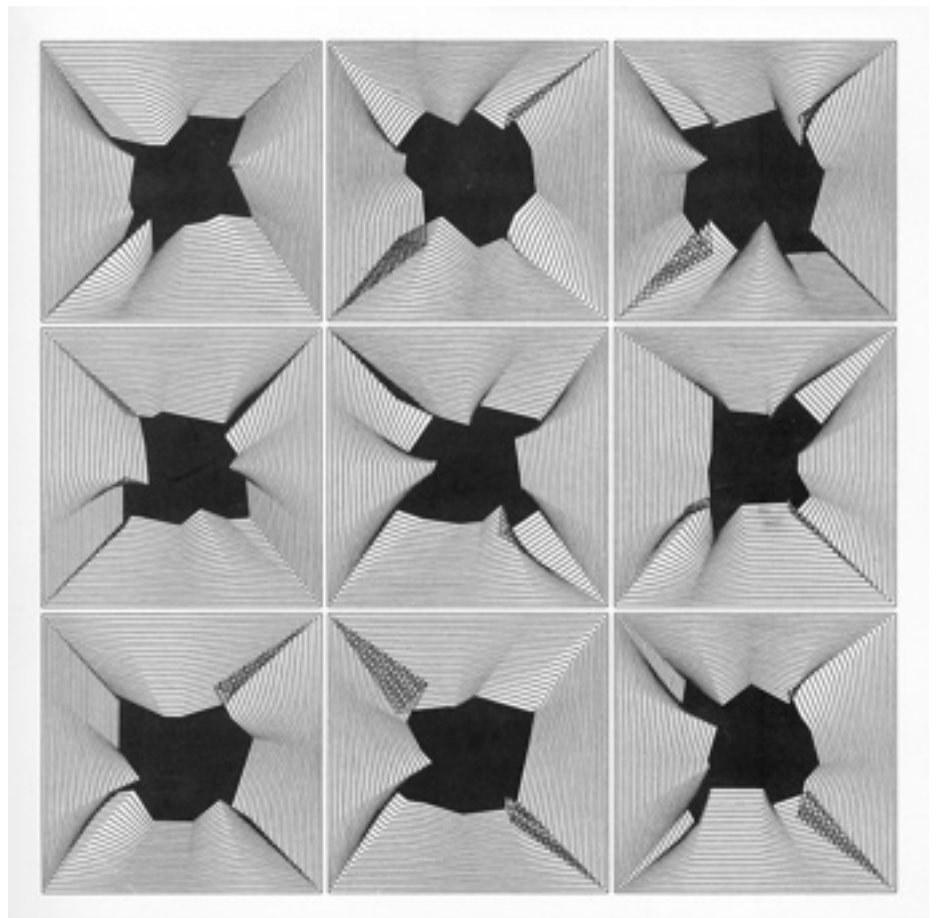
Galanter says on his article “What Is Generative Art?” (2003) that, when creating the text excerpt above, he sought to reach a definition that respected known clusters of past and current generative art activity; allowed for yet to be discovered forms of generative art; existed as a subset of all art while allowing that the definition of “art” could be contested; and was restrictive enough that not all art is generative art. It’s also useful to keep in mind that, rather than any specific art movement, generative art is a practice for obtaining an artistic object that may belong to many different artistic movements and may have many different motivations (*Galanter, 2003*).

In the same way that defining art itself has always been a complex task, defining any artistic practice is equally hard, especially when this practice became better known only recently and relies on materials and methodologies other than fine arts derived ones.

Since when it was first introduced to the artistic community, algorithmic art as a whole (what includes generative practices) has faced resistance because of its mathematical and technocratic nature. Also for attributing to the machine a great part of the creative process that before belonged entirely to the artist.

As computing is getting more accessible and relevant for popular culture though, the number of artists employing generative techniques is increasing; something that may help the whole generative art community to find a bigger place among other kinds of artists and within more conservative art galleries and museums.

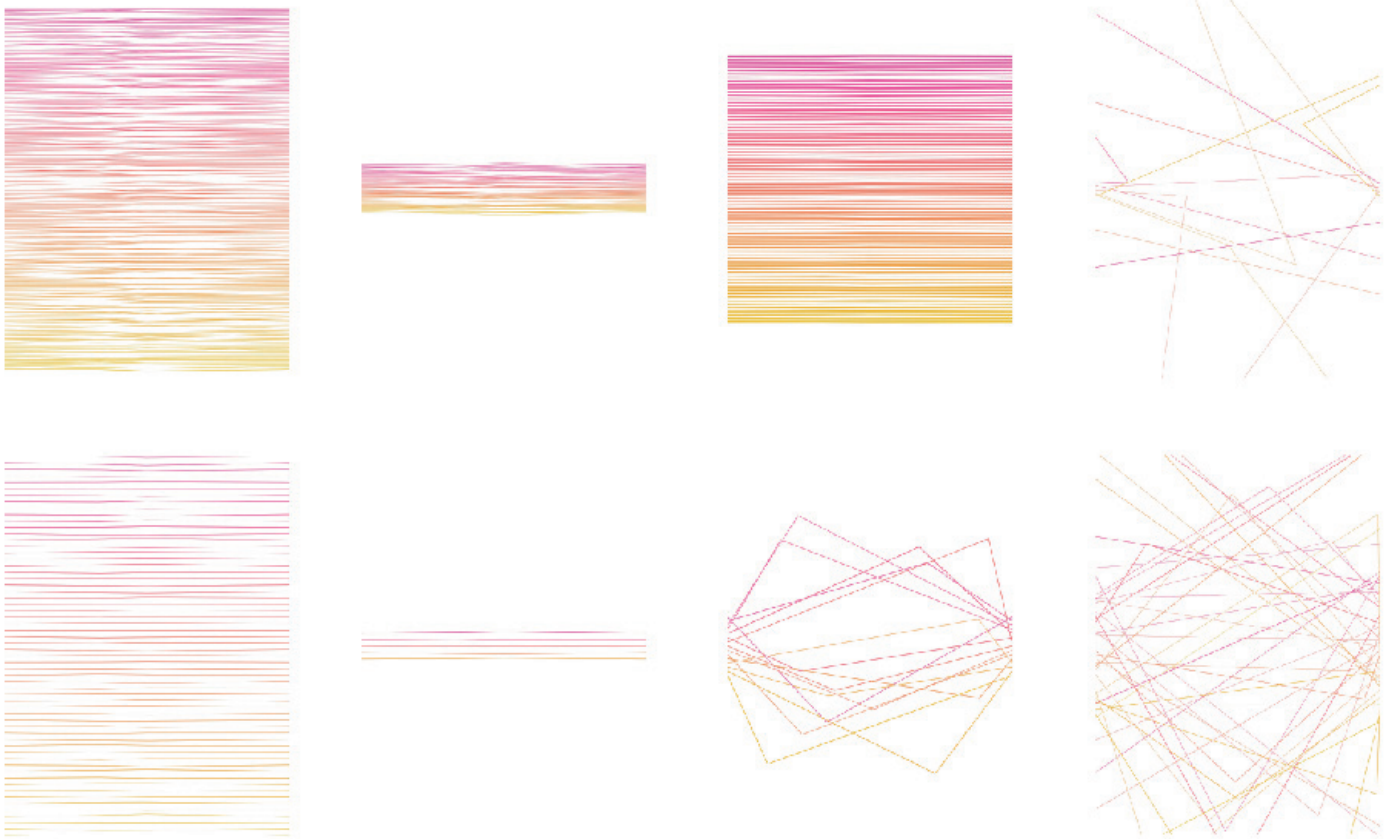
When writing an algorithm that employs a generative technique, artists usually rely on random and chaotic systems for implementing the so-called autonomy *Galanter* defined as being a key concept for generative art. There are also artists and programmers that currently explore more complex systems such as genetic algorithms, swarming behavior, parallel computational agents, neural networks, cellular automata, *L-systems*, dynamical mechanics, fractals, a-life and reaction-diffusion systems to come up with potentially more interesting aesthetical results.



Manfred Mohr. Lady Quark. 1972.

Manfred Mohr's Lady Quark (on the previous page) works as an example to illustrate how random systems can be used on generative algorithms. When producing the artwork, *Mohr* employed an algorithm that first chooses equally spaced points around each square's circumference and then randomly chooses a second set of corresponding points positioned between two smaller squares centered within the space. The algorithm finishes the picture by slowly linearly transforming the first set of fixed points towards the second set of randomly chosen points until a fixed number of iterations is reached.

Another computer generative art example, is the self created series called *Color Trails* (2017). The series of artworks employs a vocabulary of lines, breakpoints and vibrant colors to result in geometrical patterns. All of the pictures created for the series were generated by the same algorithm. The set of instructions has been built so that many different results could be obtained by changing initial parameters and relying on the randomness implemented.



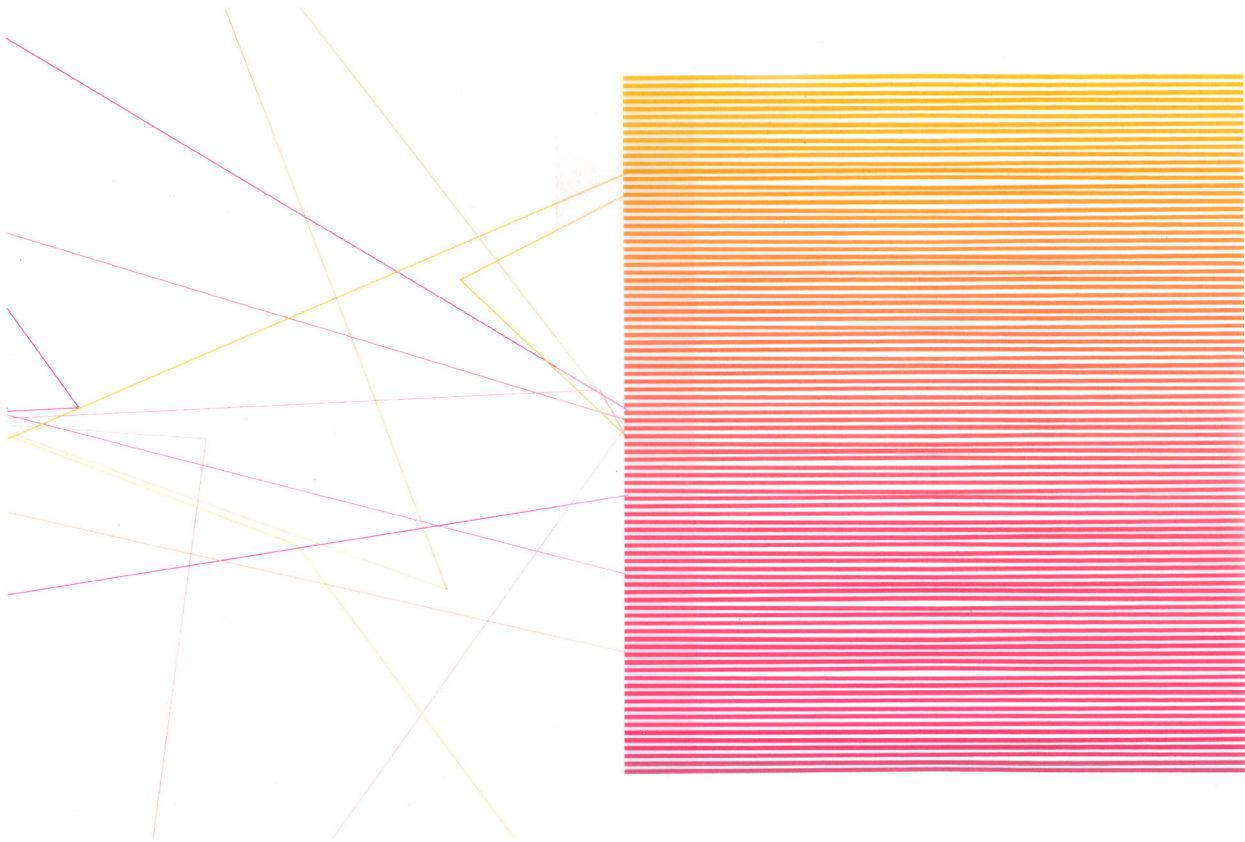
Caio Barrocal. *Color Trails* generated patterns. 2017.

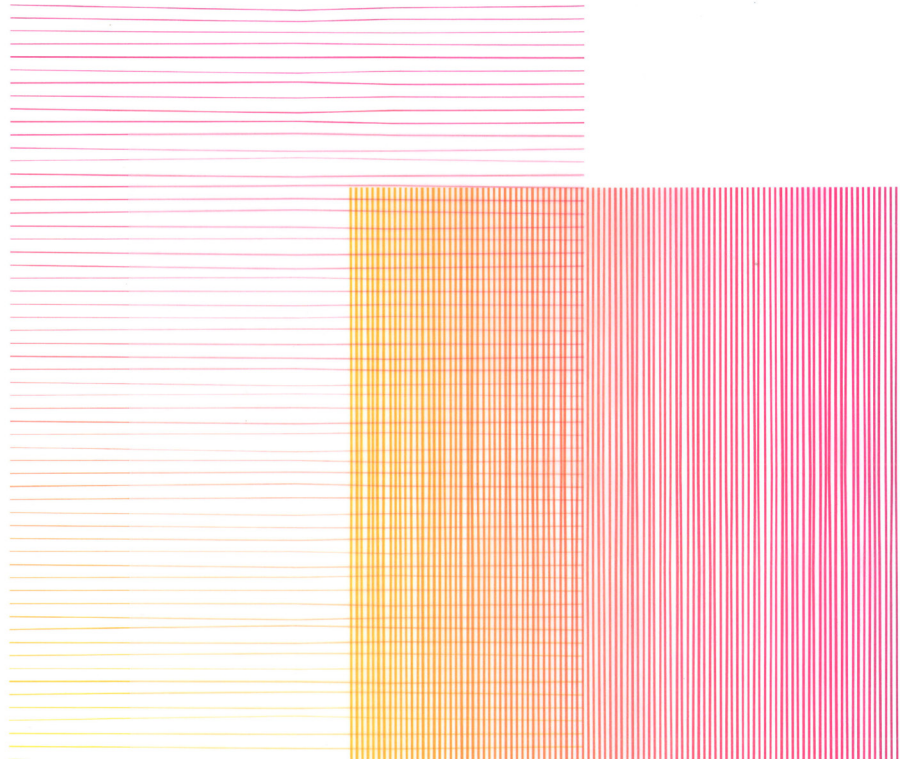
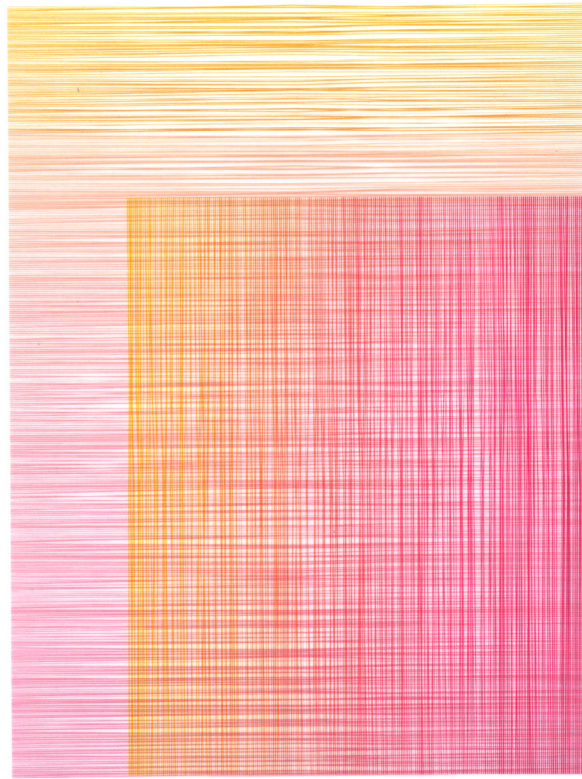
The algorithm employed on color trails is explained by the pseudocode below:

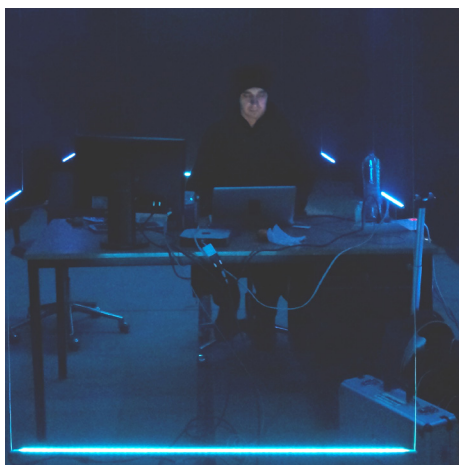
```
1  define number of lines, number of line split points, vertical distance between  
   the beginning of each line, the weight and the noise.  
2  define origin and final color  
3  for each line (from 1 to number of lines):  
4      calculate the color to be used (step on the gradient)  
5      x0 = 0; y0 = previous line + space between  
6      for each line split point in a line:  
7          calculates future pair x1,y1  
8          x1 = width/number of split points + random(-(noise),noise)  
9          y1 = y0 + random(-(noise),noise)  
10     draw line between (x0,y0) and (x1,y1)  
11     x0 = x1; y0 = y1
```

The pink-colored random function guaranteed the autonomy required for this set of instructions (or system) to be considered a generative one. The color trails series also helps clarifying the roles assumed by the artist and the machine: even though the randomness is responsible for the final results and the very diverse nature that characterized the artwork, an important role is still played by the artist, who must build the non-autonomous parts of the algorithm and determine which aesthetic attributes such as colors, sizes and positions would suit better the overall proposed concept. Furthermore, it's still responsibility of the artist to elaborate the concept and design the system so that it respects the intended motivations.

Still referring to the same example and also to demonstrate the many possible applications of generative art, a further artistic work has been done with the pictures from the color trails series by printing each one on tracing paper and physically combining them to create interesting compositions. The idea was to explore possible materialized outcomes for the imagery computationally generated on previous steps. The color trails compositions have been scanned and resulted in other images.







Indeed, besides visual art pieces, generative art techniques have been used to enhance or diversify many contemporary activities. In computer graphics, generative techniques such as *Perlin* noise and *L-systems* are used to synthesize smoke, fire, hair imagery and virtual plant life, something that ends up being efficient for creating “very repetitive” graphic elements and also that demands fewer computational power than using a traditional modeling technique (Galanter, 2003).

As a last example for the many possible applications of generative art and also the last one of this chapter, it’s interesting to discuss the role the artistic practice has within the *demoscene*, an international artistic subculture interested in exploring computational power to create music clips and audiovisual experiences.



Demoscene artists employ many generative algorithms to produce short audiovisual presentations and nightclub projections that may even be synchronized with the music a *DJ* is currently playing. Actually, nowadays it’s common for nightclubs to employ *VJs*, professionals that present algorithmically and non-algorithmically generated projections and videos while the *DJ* plays the music setlist, both composing a single very immersive experience. (Galanter, 2003).

Maurizio Martinucci, also known as *TeZ*, is an Italian artist and independent researcher who currently lives in Amsterdam and focuses primarily on generative compositions of image and sound for live performances and installations. *TeZ* builds custom software and hardware as a way to explore perceptual effects and create experiences that give life to the relationship between light, sound and space.



In 2016, *Martinucci* was asked to create the visual effects for a concert of the band *Clock DVA*, an English band formed in 1978 that is one of the pioneers of industrial, cyberpunk, and electronic sound in general. It is somewhat natural that the experimental music scene is interested in generative art and independent audiovisual productions. Both practices come together to propose innovative and thought-provoking experiences that still have difficulty finding space in the mainstream universe. For this specific show, *TeZ* employed generative algorithms to render instigating shapes and patterns in real-time during the whole event.

Top:
The Italian artist Maurizio Martinucci. 2016.

Above:
Photos of the Clock DVA show. 2016.
Photos by Marilia Fotopoulou.

As the computer evolved and society itself acquired more digital literacy, many people benefited from digital solutions for pushing their own creativity. Whether if applied into visual arts, animation, cinema, sculptures, projections or sound performances, generative art practices can be considered a very contemporary attempt of employing machine expressiveness into media and arts.

Since the sixties, computer art went through many already introduced changes and resignifications. These transformations conducted the movement to the very present moment of exploring and understanding how technology can influence the experiences people have. Sharing the same principles and prescriptions of algorithmic and generative art, many creative coding solutions have been developed to give life to innovative devices and installations that seek to amaze, introduce other forms of interaction and also to investigate the expressive potential of the computer as a medium of communication and creation.

CURRENT AESTHETICS AND MOTIVATIONS

“Like literature or painting, software can be appreciated from the inside out (basing an interpretation on a program’s syntax, or internal architecture) or from the outside in (reading the function of a software application against a broader social context)” (*Blais; Ippolito, 2006*).

As computing techniques became more accessible, many different people (from non-technical to extremely technical ones) began to employ software in their tasks and with the most diverse intentions.

Past are the days computer code was exclusively employed by programmers and engineers; today we can witness the maturing of the first generation of artists and designers who employ computers to bring their experiences to life. Those previously bounded to screens and portraits experiences (websites, user interfaces and visual art pieces) now also found place within the physical world and closer to the user. The evolution of technology allowed artists and designers to build, with the help of electronics and mechanical systems, complex objects and spaces that communicate an overall concept. Alongside screen-based works, a lot of effort is being put for creating innovative physical experiences that amuse the spectator (*Klanten; Ehmann; Hanschke, 2011*).

Among the digital art community, there is currently a great desire to develop interactive experiences by building code and structures that allow users to participate, and even modify the final artistic objects. However, it’s still difficult to combine the immaterial code and material mechatronics to create a consistent dialogue and concept for users to engage in. Partly due to the fact that today’s users are more digitally literate and, therefore, more demanding with the experiences they receive, and also because building the necessary technical infrastructure is a complex activity.

This chapter discusses the role of coding in creating contemporary artistic experiences with different types of user interaction. When designing an artistic experience, one should keep in mind which type of interaction is most compatible with the proposed concept: one that enchants only by visual and sound; something that proposes that the user interact through touch, speech or movement; or even one that relies on connecting to the user through internet so that it influences the final result. It’s also useful to say that the very multidisciplinary nature of the activities discussed makes room for fruitful collaborations between designers, artists and computer scientists and engineers.

Coding for Visual Engagement

Even though today it's somehow easy to recognize software as a material for building visual interfaces and objects, sometimes it seems unlikely that the very rigid and formal nature of the algorithms can originate immersive and amusing visual spectacles. The previous chapters of this project discussed a pioneering scenario and evolution path that, actually, established a movement that keeps achieving even more amazing aesthetic results.

A first application and, also one already introduced on the previous chapter, is coding for creating visual and audiovisual products. Actually, an important motivator for artists and programmers to explore code for generating this kind of artistic objects is the very emblematic nature that algorithms have obtained in popular culture as technology and computing techniques became more accessible and compatible with the most diverse activities.

Besides only emulating traditional media and seeing the computer as a digital version of painting, drawing, magazines, cinema, 3D-modeling and so on; having a greater familiarity with computing through programming techniques allows one to create new medias with rules and dynamics more naturally suitable for specific tasks. The *Harvard* chemist and physicist *Eric Heller (1946)*, for instance, says computers have allowed him to use physics as a brush. He creates digital art pieces by painting with electrons moving over potential landscapes, quantum waves trapped between walls, chaotic dynamics and with colliding molecules (*Heller, 2006*).

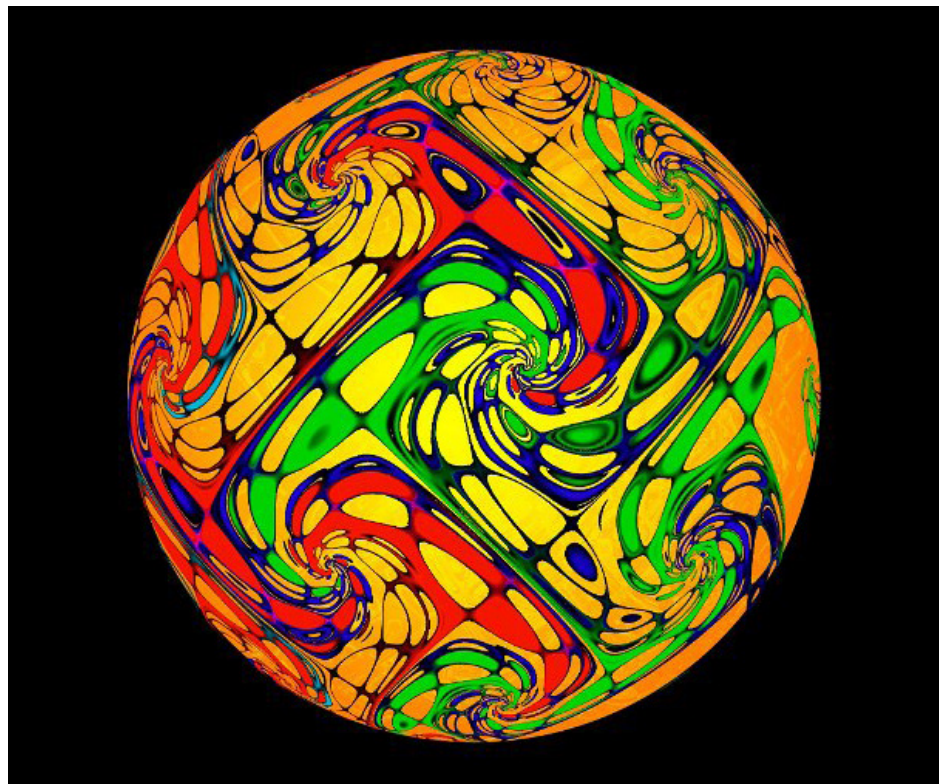
Heller claims to follow a process that unites both of his personalities (the scientific and artistic ones) when producing a visual artwork. He achieves this by writing algorithms based totally on the rigorous applications of the physical models he studies as a researcher and then, manipulating the resulting colors using *Photoshop* to produce “museum-quality” prints that would suit better buyer's preferences. The physicist-artist also argues that art can indeed lead to science, a claim he justifies by pointing out that chaos theory, for example, attracted much more attention and investment because of the impressive computationally generated images obtained, such as fractals (*Blais; Ippolito, 2006*).

Even though the final aesthetic product often overshadows the process responsible for creating the artwork, part of the value computer art pieces have lies in the very relationship between the set of instructions established (the system or algorithm) and the visual complexity and satisfaction of the result. However, it's valuable to say that there isn't a defined level for this relationship to exist; while some artists cherish simplicity in the rules created, others seek increasingly complex procedures in order to attribute to their pieces a more experimental and innovative character.

Some artists also argue that the process established should be made very clear for the general audience as an attempt to make the product-process relationship more palpable and to promote a greater understanding of the work as a whole. Elucidating the set of rules used is not always an easy task, though; while some works rely on more simple rules and algorithms that possibly would be more easily understood, a hypothetical viewer would have to be very skilled in physics theory in order to fully understand *Heller's* artwork. From the moment of planning to actually building the work, it's important to have this natural tension in mind.



Left:
Eric Heller. Exponential. Undated.



Right:
Eric Heller. Dissipation. Undated.

Within the audiovisual world, many generative artists produce computer-generated objects that come to life through movies and mapped projections, a technique that utilises unusual surfaces and locations such as buildings and landscapes as a setting for animation.

When creating this sort of projections one commonly rely on software interfaces such as custom ones developed with the *Processing* language or commercial ones to map light onto any surface other than the usual flat-white ones. This process ends up turning objects of any three-dimensional shape into interactive displays that can be used for many different motivations such as advertising, live concerts, theater, gaming and decoration.

During the opening ceremony of the 2008 *Olympic Games*, an astonishing performance has been prepared by the German creative media production company *Congaz*. The challenge was to create a serene and dreamlike spectacle that represented nature and the elements of fire, water and light.

To accomplish this task, *Congaz* prepared a series of aesthetically instigating digital animation pieces and employed 500 support actors responsible for moving 268 gauze screens around the arena. These moving screens then received the prepared animations by being illuminated by 22 video projectors distributed around the arena; thus also resulting in the largest video projection screen in the world.

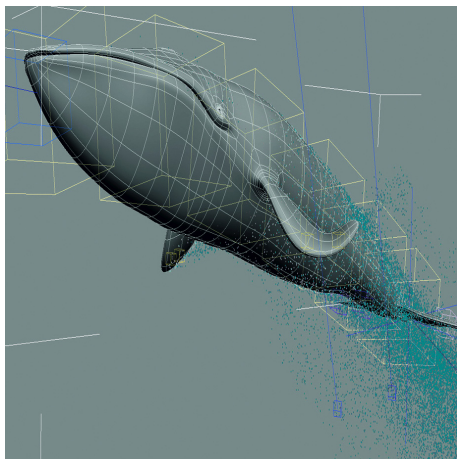
A still from Congaz's recorded performance. 2008.





A still from Congaz's recorded performance. 2008.

Besides coding for creating software interfaces responsible for mapping the projections, most part of the objects exhibited were computationally generated, such as the real-size blue whales displayed on the top of the arena and the green fighters reproducing *Tai-Chi* movements that were created through data from sensors.



Left and Right:
Congaz. A still from the Making Of movie. 2008.



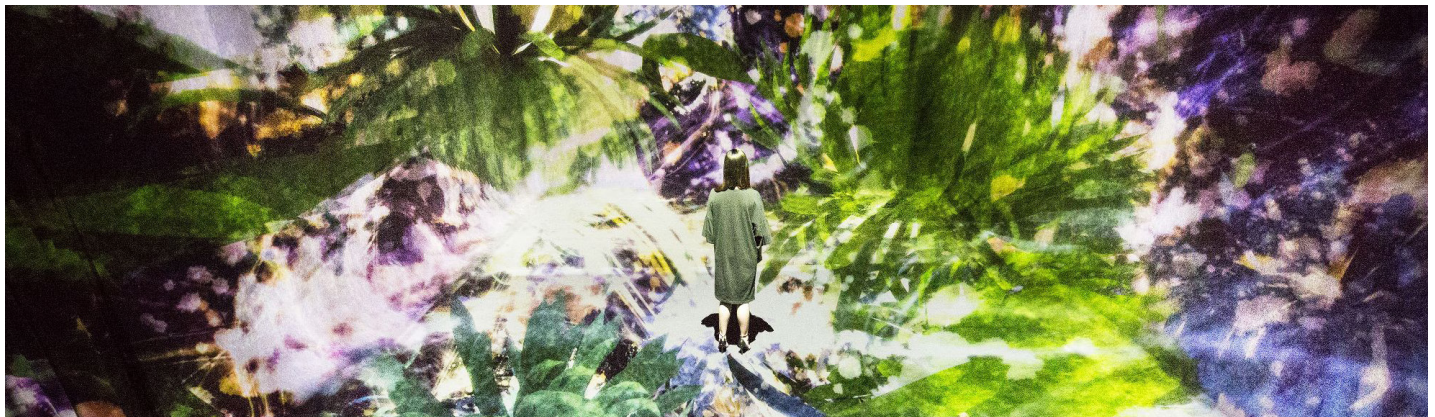
Another example of computer art piece for optical seduction is an interactive installation created by the Japanese digital art collective and company *teamLab* in 2017. The installation named "*Crows are Chased and the Chasing Crows are Destined to be Chased as well, Transcending Space*" invites viewers to freely explore a space that completely surrounds them with projections on the walls, floor and ceiling.



Previous page and bottom:
Team Lab. Crows are Chased and the Chasing
Crows are Destined to be Chased as well,
Transcending Space. 2017.
Experience photographed.

The projections show real-time rendered crows that fly around the room generating visual patterns and light-trails. Since the space is totally employed as the display and the group built it so the boundaries between the floor and the walls disappeared, the final experience is a very immersive and contemplative one. An important role is also played by the soundtrack, that enhances the overall experience.

Generative art techniques are used to determine many elements and parameters of the audiovisual product such as the trajectory and speed of the crows; colors and density of the light rays; and the position of elements. A computer program is used to implement these generative techniques and to render the whole installation in real-time so that previous states are never repeated and every single experience is literally unique.



Coding for Engagement through Actions

Another cluster of activities that promotes fruitful explorations in generative and digital art is the design of experiences that invite users to actively interact through touch, speech or movement. This sort of interactive experience can benefit from generative techniques, for example, by relying on systems in which the autonomy is obtained by reacting to the very actions users perform when interacting.

When designing an experience for user to actively interact, one should keep in mind the many possible technology solutions for translating users' actions into stimuli for the designed system. While screen-based and touching experiences are more intuitive and natural due to the familiarity promoted by smartphones and tablets, there are currently many devices that work as sensors for capturing movement, noise, distance, humor, stress level and other attributes that can be used as input data for interactive systems.

The very popularity of *Kinect*, a motion sensor designed for the *XBox* console, for example, showcased the potential this kind of sensor had to enhance interactive experiences in a more natural way, as it didn't require joysticks or cables. Currently, many interactive installations rely on similar technologies to promote a more palpable and immersive experience once the lack of cables and attached devices facilitates the establishment of a more natural dialogue between the user and the artwork.

In 2015, the American interactive art studio *Red Paper Heart* based in New York, created an art installation inspired by the *HBO* series, *Game of Thrones*. The installation relied on motion sensors and generative art techniques to convey the sword training journey of one of the main characters.



Red Paper Heart. *Game of Thrones* Sword Experience. 2015.
Photos by David Terranova.



The *Red Paper Heart* team captured the swings of the swords by embedding them with wireless gyroscope and accelerometer sensors that could measure the trajectory and velocity of each swing. Data from the swings was used as input for generative art algorithms that translated the very physics of the movements into an artful self portrait.

The autonomy of the generative system employed was achieved by designing algorithms that could take advantage of the most diverse possible movements of the user and use them to generate unique artistic objects. Once hit, targets were dismantled into artistic elements that kept scaling, rotating and moving in the specific way the user swung the sword.

For creating the whole visual experience, the *Red Paper Heart* team developed a custom graphics engine written in *Cinder* (a C++ framework) that was responsible for blending both 2D and 3D objects inspired by the *Game of Thrones* universe. By masking the generated objects with the user's own profile photo, fans could reveal a portrait of themselves as the targets were hit and the artistic objects filled the artwork.



Top and right:
Red Paper Heart. Game of Thrones Sword
Experience. 2015.
Photos by David Terranova.

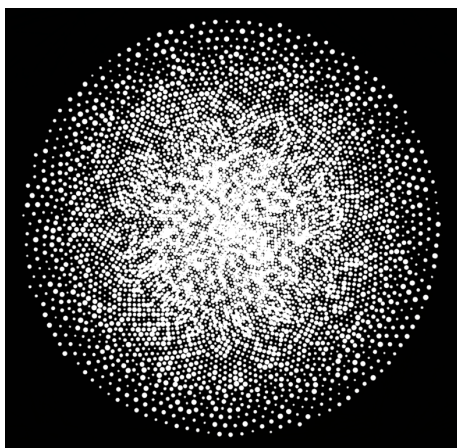


The Aether installation. 2014.
Created by Thomas Sanchez and Gilberto Castro.

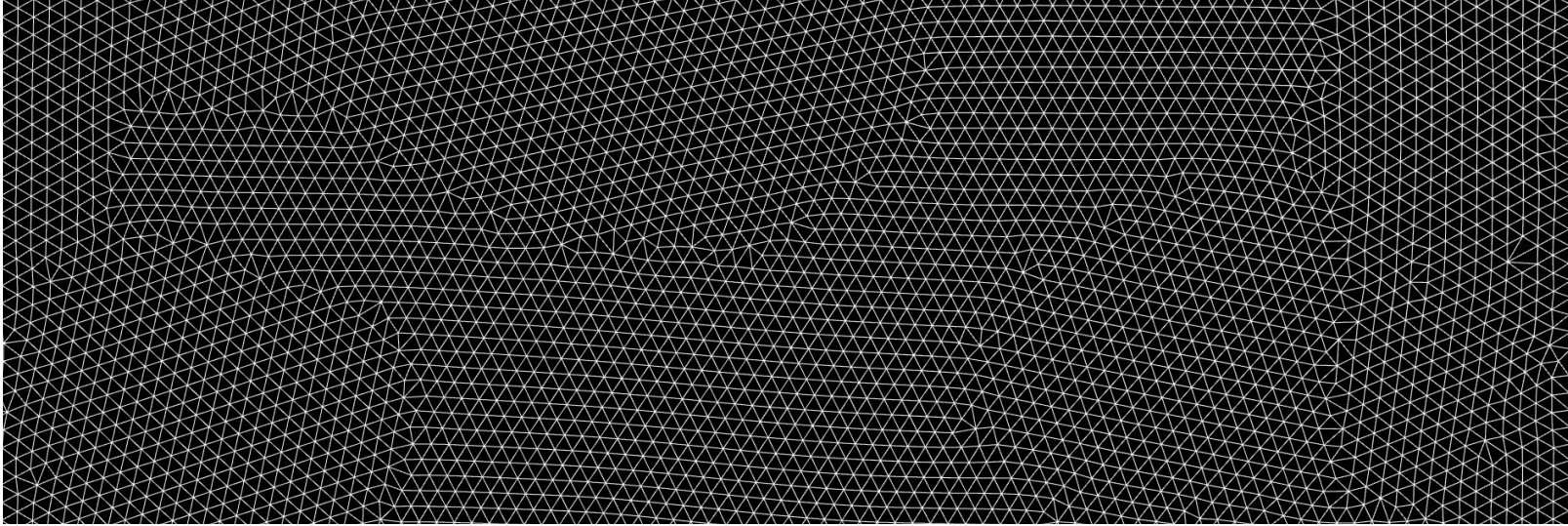
Another example of experience that relied on an interaction through actions is a project developed by the artists *Thomas Sanchez* and *Gilberto Castro* in 2014. *Aether* was designed to work as a multi touch wall that invited the user to run their digits through a “cosmic pool”.

Historically, “aether” was the name given to the material that was believed to fill the universe beyond Earth; the concept was used by theorists to explain several natural phenomena, such as the traveling of light and gravity until the 19th century. Inspired by the mythological fifth element, *Sanchez* and *Castro* conceived an installation that invited users to a cosmological experience by running fingers through large format screens powered by generative software. *Aether’s* interaction dynamics consisted of letting users manipulate source footage of the universe taken from the *Hubble Space Telescope* that were used as input for generative algorithms that created instigating shapes and colors.

A simulation showing the particle system with an attraction force on the center.



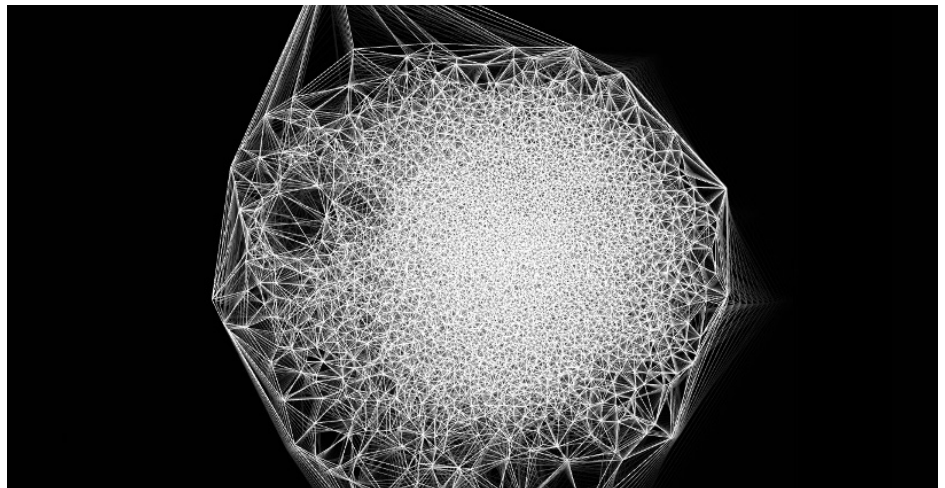
The application developed for bringing *Aether* to life was written in *Cinder* and consists of a dynamic particle system with simple interaction forces, such as repulsion and attraction, rendered in real time. Concerned with the performance of the application when interacting with the user, the artists decided not only to use *Cinder*, which is substantially faster than similar tools such as *openFrameworks* and *Processing*, but also chose to use the *Euler* method to solve the positions of the particles in a shorter time. Even though there is some loss of numerical precision when compared to other methods such as the *Runge Kutta* one, the *Euler* method runs faster and, therefore, is more suitable for real-time interactions. After developing the dynamic particle system, the artists relied on the *Delaunay* triangulation algorithm to map the space and calculate the triangulation of almost 3000 points in real time. This procedure resulted in a mesh of points that behave following the physics simulation of the particles and that could be disturbed by users’ touches.



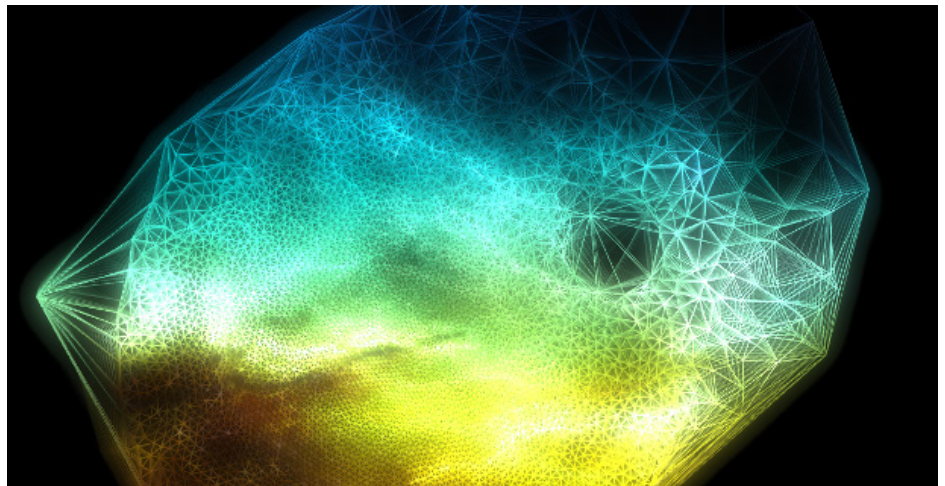
Aesther's triangulated mesh.

In the last steps, the artists used elements of the pictures from the *Hubble Telescope* as textures to be applied on the mesh of points obtained previously. By using a *Cinder* extension called *Tri Mesh*, they could easily map each point of the mesh to a point of the picture, thus creating an association between the particles and the texture that resulted in a moving object and also one that could be distorted as the user interacted with the installation.

Tringulated mesh after implementing repulsion forces through mouse click or touch as to obtain responsiveness.



Responsive triangulated mesh after applying texture.



The *Aether* installation impresses, not only for the beautiful aesthetic result, but also for the care that its creators had with the performance of software and hardware.

Below and bottom:
The *Aether* installation.



Coding for Engagement through Involvement



VISIBLE. Paris Expo Porte de Versailles' light installation. 2014.

Besides conceiving experiences by only reacting to users' actions exactly on a given period of time, designers and artists are also currently exploring the creation of installations and objects that keep evolving, living and learning from each interaction. Often, besides relying on many people interacting at the same time to compose a single great experience, some of these artistic devices even store previous states to be transformed by users' actions as to create a dynamic and coherent spectacle.

When designing an artistic object that perceives the environment and people around, artists often employ software for real-time processing and many kinds of sensors for collecting data and using it as input for generating the experience. Actually, it's common for these installations to be enabled by software and machinery that run continuously during all the time planned for the exhibition.

A good example of experience that learns for reacting is the art installation developed by the Slovenian studio *VISIBLE* for the French exhibition center *Paris Expo Porte de Versailles* in 2014. By that time, the studio was asked to design some installation that could track cars on a highway and use this data to generate light patterns in real-time.



To develop the proposed experience, the studio decided to use images from a camera as input for the software that controlled the light blobs. By first making experiments and recording hours of videos from highways near the site where the studio is located, the development team could obtain thousands of images of vehicles that, then, were used as a training set for Machine Learning classification algorithms.

Once the system was trained and was able to track the elements it needed to track (vehicles and their position on the road), the studio built the structure in Paris. To properly set the visuals, the group used light blobs that gently undulated as the traffic varied: when the density of the traffic increased the blobs became brighter and livelier. There were also two different colors the light blobs could take on: orange for representing vehicles heading south and magenta for the opposite side.



Top:
Camera used by the VISIBLE team to obtain traffic data.

Above:
VISIBLE. Lighting that reacts to Traffic installation, 2014.

UVA's Array installation at the courtyard of the Chuya Nakahara Memorial Museum in Southern Japan. 2014.



Array, an art installations created by the British group UVA (*United Visual Artists*) in 2008 is another example of experience that relies on the audience to be conceived. The installation was set on the courtyard of the *Chuya Nakahara Memorial Museum* in Japan and consists of a field of light columns that invites people to explore and interact.

By employing a hidden network of ultrasonic sensors, the creators could conceive a mysterious experience that responded to the movements of the viewers across the light field. The columns created light and sound effects, gently shifting and shining, according to people's position within the courtyard and their trajectory as they explored it.



As to give life to a “shy spirit” living within the light forest, a single red light would emerge from time to time among the white light columns, gently move, interact with the viewers and disappear. The whole ambience was created solely based on electronic devices, and the autonomy of this generative system was obtained by reacting to viewers’ movements.

Top and left:
UVA’s Array installation at the courtyard of the
Chuya Nakahara Memorial Museum in Southern
Japan. 2014.

As computing and technology evolved, many abstractions and high-level solutions have appeared as to make it easier for even non-technical people to design and build their own experiences. Something that is pushing the discipline's traditional boundaries and patterns and consequently bringing computer science many diverse visions and motivations.

Rather than showing specific examples and clusters of interactive experiences, this chapter sought to explore the current art-technology scenario as a fruitful medium for multidisciplinary collaborations; and with many instigating applications and results. All around the world, studios, agencies, consultancy firms and innovative companies are looking for people capable of using technology to bring innovative experiences to life. Related positions commonly require some knowledge in virtual and augmented reality, design, web development technologies, basic electronics and most famous creative coding frameworks such as *Processing*, *openFrameworks* and *Cinder*.

CURRENT TOOLS FOR COMPUTER ART

While extremely scarce back then, there are now plenty of good and different options for those looking for creating art with the computer. While some of these solutions are more focused on highly technical and computer graphics professionals, others are developed with comprehension and ease of use in mind. *Processing* is a good starting point for both artists/designers interested in learning programming skills and computer scientists who are starting to explore software within the visual arts.

This section discusses the three most famous languages or frameworks for generative art and also examples of art installations built with each of them.

Processing

The *Processing* project was created in 2001 by *Casey Reas* and *Ben Fry* while both were still undergraduate students at the *MIT Media Lab* (more specifically, within *John Maeda's Aesthetics and Computation* research group). At first, *Processing* was born to become a software sketchbook, a teaching tool for people to practice programming fundamentals within a visual context. Later, and as the community started growing, the language has received more and more improvements and evolved into a professional alternative for proprietary software tools, such as *Adobe's Photoshop* and *Illustrator*.

```
sketch_180806b
float pX;

void setup() {
  size(510, 510);
  background(255);
  noFill();
  stroke(0);
  pX = width/2;
}

void draw() {
  background(255);
  ellipse(pX, height/2, 30, 30);
  pX = pX + 1;
}
```



The example above shows a circle moving to the right, 1 pixel per frame; the draw function written below renders a white background and then renders the same circle 1 pixel to the right in relation to the frame before.

The tool is an extension of the *Java* language, with additional simplifications such as additional classes and built-in mathematical operations and functions; and an *IDE* that works as a sketchbook and makes it easier to run a program. When programming in *Processing*, one can keep in mind the same syntax and object-oriented logic from *Java*. Actually, knowing *Java* can help programmers building more complex things that would exceed *Processing's* native capabilities. Since the tool was created to serve as a first language within a visual context, it's simple and straightforward to obtain relevant shapes, the syntax is easy for people who never programmed and even easier for those with programming experience in other languages.

Due to the more visually-oriented nature of *Processing*, the most useful functions are the *setup* and *draw* functions. The *setup* function will run before every other one and "prepare" the sketch with default attributes such as size, background color, renderer, color mode and frame rate; and the *draw* function will keep the piece of code *Processing* will run in every frame exhibited.

Even though it's simple to start programming in *Processing*, the software can deliver high-level and versatile outputs. There are lots of artists, visual designers, architects and computer scientists using *Processing* to create some really impressive and different works, such as projections for dance and music performances; images for music videos and film; images for posters, magazines, and books; and interactive installations in galleries, museums, and on the streets. Some works were featured at museums and art centres like the *Museum of Modern Art* in New York, the *Victoria and Albert Museum* in London and the *Centre Georges Pompidou* in Paris.

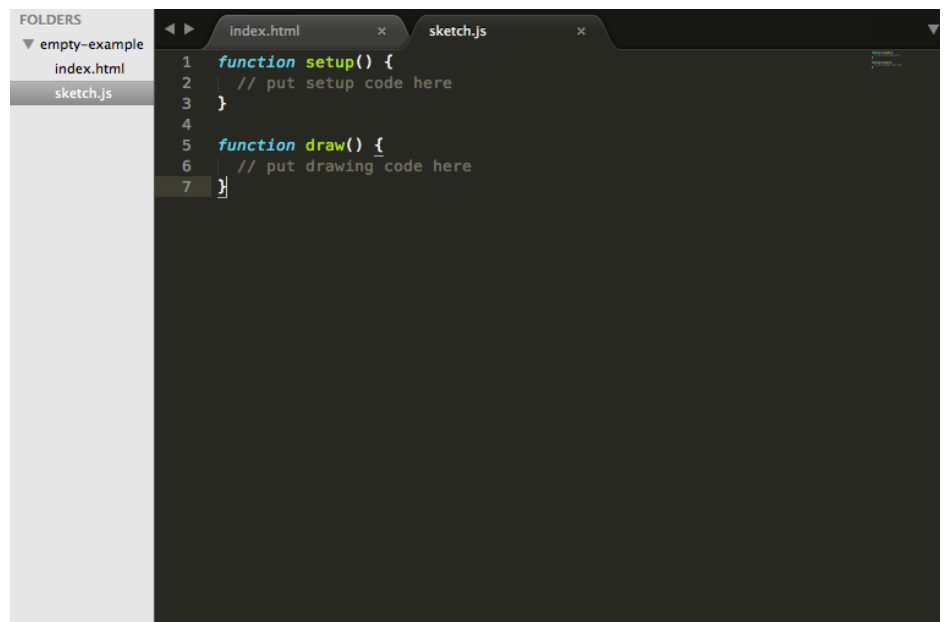
P5.js and Processing.py

As an attempt to translate the same visually-oriented *Processing*'s nature to the web, *P5.js* was created. The *Javascript* library implements the same software sketchbook metaphor and makes it easy to interact with *HTML* objects and other web technologies through its libraries.

The programming logic is almost the same as the original *Processing*, except for some specific syntax changes. People with experience with the original version will find it easy to migrate and start exploring web technologies within a visual context.

Processing also has a plug-in for *Python* programmers that implements the same setup and draw logic into *Python* syntax.

Example of p5.js basic syntax and logic.



```
FOLDERS
▼ empty-example
  index.html
  sketch.js

1  function setup() {
2    // put setup code here
3  }
4
5  function draw() {
6    // put drawing code here
7  }
```


Processing case: Plausible, Possible, Potential, by Miguel Nóbrega



Miguel Nóbrega. *Plausible, possible, potential*. 2015.

Nóbrega is a Brazilian designer and visual artist who uses generative algorithms to explore the relationship between art and technology. He's received his Bachelor Degree from *ESDI, Rio de Janeiro* and also holds an MFA from *UCLA Design Media Arts* in Los Angeles.

Plausible, possible, potential is a series of visual art works produced by *Nóbrega* in 2015 in which he employs generative algorithms to illustrate architectural structures and spaces that cannot actually exist. The artist used *Processing* to obtain the shapes and also to control a plotter machine that draws them using different watercolor pens attached to it.

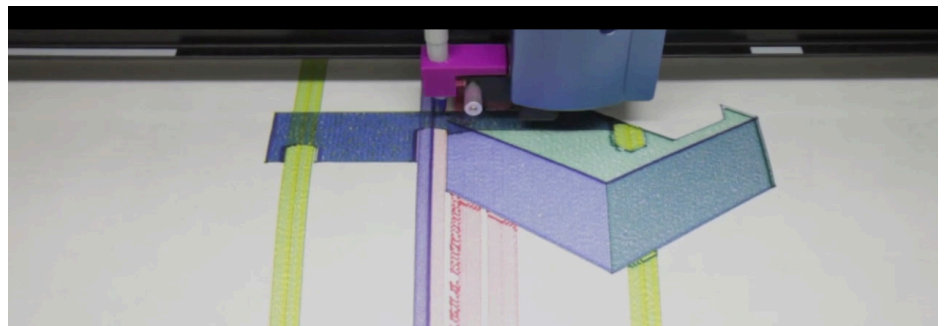


Photo of the CNC machine *Nóbrega* used to create the series.

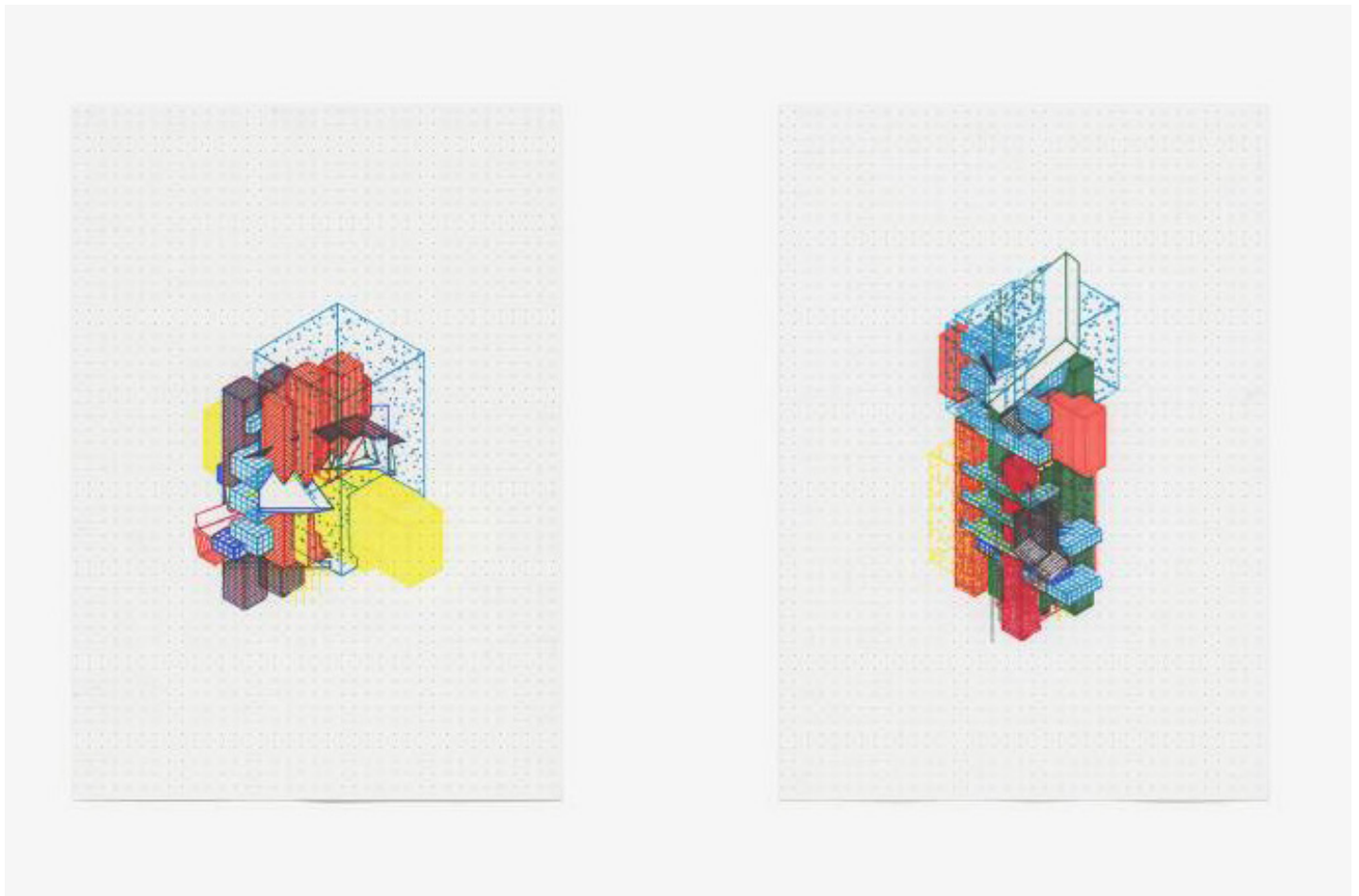


Top and bottom:
Miguel Nóbrega. Plausible, possible, potential.
2015.

Each artwork is generated by randomly instantiating initial properties every time the code is run, so that, every execution results in a different draw that falls into one of the three different themes: *Possible Structures*, *Potential Sculptures* and *Plausible Spaces*. In some way, all of them remind one of the explorations of space and illusions that *Escher* employed in his works and also to the use of perspective for the representation of three-dimensional spaces.

Algorithmically, *Nóbrega* achieved his final results by creating objects that stored each specific drawing pattern and also by calculating how these potential objects could interact with each other in order to create an overall fake isometric perspective. In other words, he achieved the best logic for objects to be drawn so that they would result in an impossible structure.

Part of the interesting and instigating aesthetic result was also obtained by relying on the very nature of watercolor pens. Each illustration that composes the *Plausible*, *possible*, *potential* series is 2 meters tall and, since it required a lot of ink, approximately 10 watercolor pens were used to come up with one picture. Part of the immersive effect was obtained by keep drawing until the pen's ink was finished, thus resulting in a pattern by varying the intensity of the color.



OpenFrameworks and C++

Processing may be a really versatile and easy to learn language and platform, however, some more complex applications and projects will naturally demand a higher performance. Due to its very close-to-machine-code nature, when working with 3D-graphics, interactivity and/or rich animation, C++ tools tend to run faster than *Processing*; even though it may get harder for one to build the code and maintain it.

When working purely with C++ to come up with art and design pieces, programmers commonly employ traditional libraries such as *OpenGL*, *GLEW* and *cairo* for graphics; *FreeType* for fonts; and *rtAudio* and *OpenAL* for audio input and output. As an attempt to wrap together these and other several commonly used library, *openFrameworks* was developed: an open source C++ toolkit to assist the creative process.

Just like *Processing*, *openFrameworks* was developed with ease of use in mind and can be applied to several different art outcomes such as physical installations, visual art, animation, interactive projects with sensors and so on. It's also a very good way to start programming in C++, since the toolkit works as an intermediate layer with between the real low level C/C++ programming style and a high level approach obtained with *openFrameworks*.

The toolkit is also very flexible and would possibly please programmers with very different styles since it works with either a higher level logic obtained with C++, or a very low level oldschool C code. *OpenFrameworks* also makes it easier to develop graphic user interfaces and interactive environments.

OpenFrameworks also works with a *setup-and-draw* structure similar to *Processing*. Most functionality in the toolkit works using the pattern *setup*, *update* and *draw*: the *setup* method is called once at the beginning of the application while the *update* and *draw* methods are called infinitely, one after another in that order, until the end of execution. The *update* method is meant to be used for updating the state of the system and making the changes required for the next frame to be drawn. After this update step, *openFrameworks* calls the *draw* method to, properly, draw to the screen.

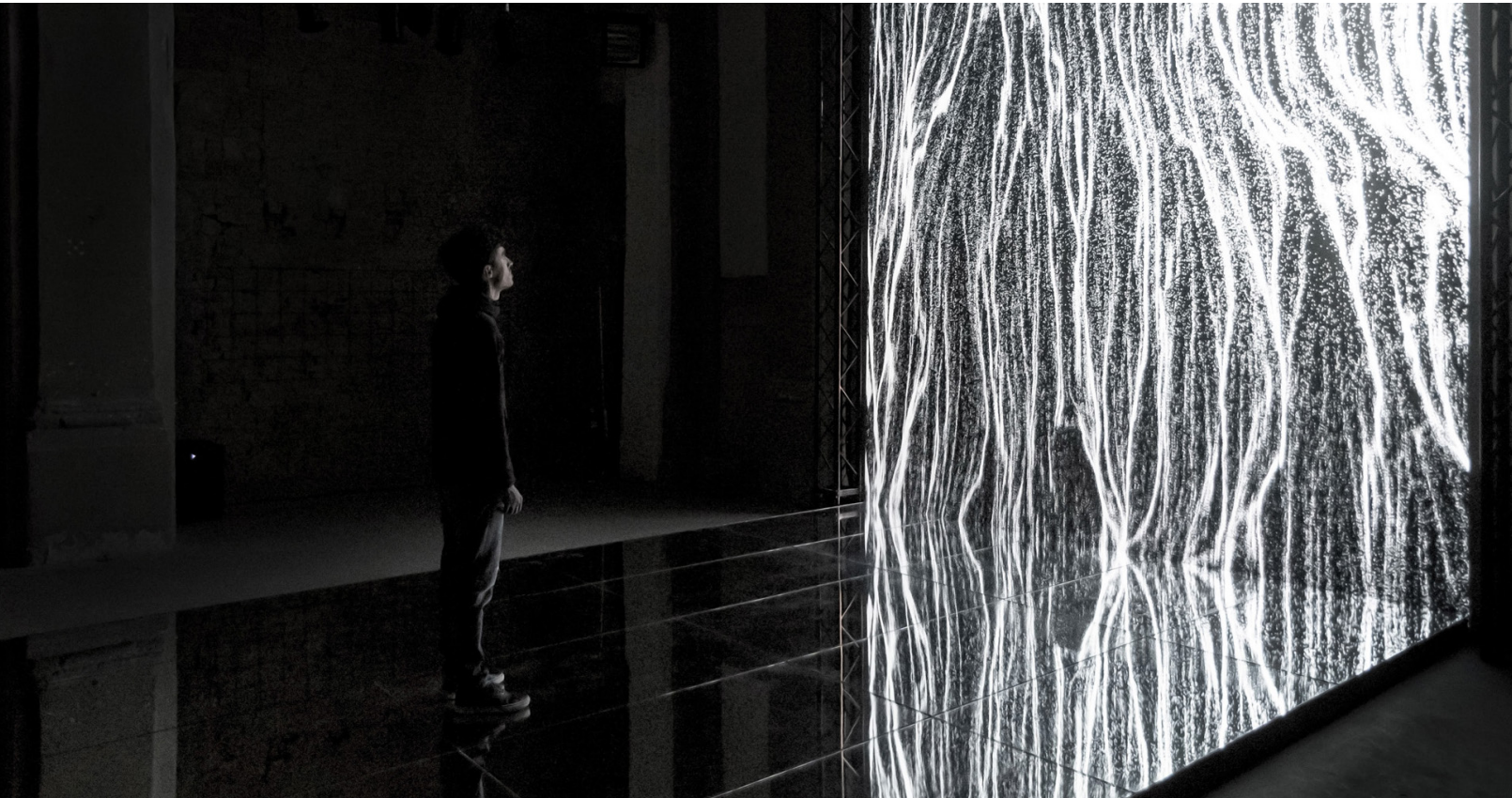
The tool can be installed in many operating systems and has a very active community that helped and continues to help the tool to get better. The project's website also has a learning section with very good documented steps: <https://openframeworks.cc/learning/>, although it requires at least some knowledge in C++ programming.

```
ofApp.h
float x;

ofApp.cpp
void ofApp::setup(){
    x = 0;
}
void ofApp::update(){
    x++;
}
void ofApp::draw(){
    ofDrawCircle(x,120,30);
}
```

Example of openFrameworks' basic syntax and logic.

OpenFrameworks case: MULTIVERSE, by fuse*



fuse*. MULTIVERSE. 2018.
Photo by Emmanuele Coltellacci.

MULTIVERSE is an audiovisual installation created by *the fuse** group in 2018. The project's creators define their work as an audiovisual installation that explores the evolution of infinite possible universes through the use of generative graphics and sounds that exploit the theorization of the existence of the so-called multiverse: a system composed of an infinite number of universes that coexist parallel outside our space-time (*Fuse, 2018*).

In order to build a coherent narrative context with the installation, the group based their work on the multiverse theory developed by the American physicist *Lee Smolin* in 1992. In his work, *Smolin* discusses a “Cosmological Natural Selection” in which all the existent universes were actually born from the collapse of previous universes and their specific laws. These collapses would result in universes ruled by different laws compared to its “parents” and, therefore, one can see the existence of Earth and human life as partly casual and, at the same time, a result of this evolutionary process.



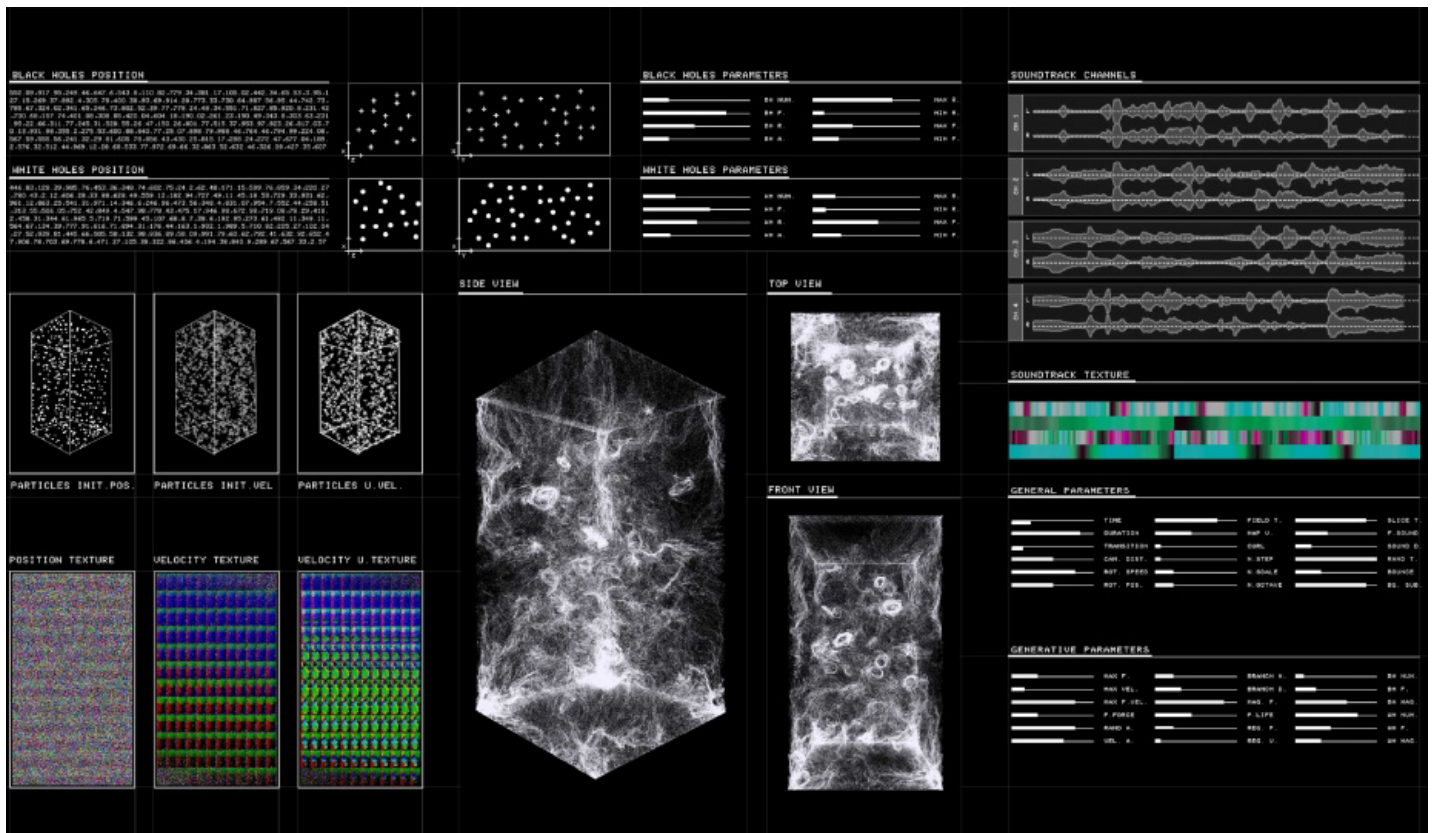
When projecting the *MULTIVERSE* installation, the group aimed to create a very intimate and immersive experience for the viewer while maintaining the hierarchies between the human and the vast and incomprehensible phenomena happening in front of him. The feeling of hierarchy could be obtained by establishing a proportion in which the spectator perceived himself as smaller than the events taking place.

An application developed with *openFrameworks* acts as the “the creator” of the scenes that are displayed. These scenes also interact with the digital audio tools *Ableton Live* and *Max/MSP* in order to produce the soundtrack through a generative sound system.

Below:
Examples of the simulated particles generated by fuse's application.

Bottom:
Interface of the software developed for generating the scenes.

Algorithmically, the application produces simulated particles that follow global motion rules that keep changing slightly and autonomously when the “collapses” occur, also a generative technique. At the same time, the program uses a *voxel* space as the data structure that stores system states and the interaction between those particles.



Cinder

Cinder is another C++ tool mostly used for aesthetic results created by the artist and computer scientist *Andrew Bell* in 2010. Just like *openFrameworks*, the *Cinder* library wraps up other commonly used libraries for creative coding and makes it easier for developers to build graphic results with its built-in functions. It can be installed on *Windows*, *Linux*, *Mac* and *iOs* operating systems and is generally used in non-browser environments.

Cinder is a powerful and production-proven tool suitable for both computer graphics professionals and artists, although it may be more complex compared to *Processing* and *openFrameworks*. Actually, the main difference between *Cinder* and other C++ frameworks is that it uses more system-specific libraries for achieving a better performance. The tool also stands out when building heavily abstracted projects such as art installations, marketing campaigns and advanced animation projects.

Just like *openFrameworks*, *Cinder* also follows the *setup*, *update* and *draw* logic: the *setup* method is called once at the beginning of the application while the *update* and *draw* methods are called infinitely, one after another in that order, until the end of execution.

Cinder case: Red Bull Music Academy ‘A Night of Spiritual Jazz’ Installation by The Mill



The Mill. A Night of Spiritual Jazz Installation.
2016.
Photo by Drew Gurian.

The Mill is a visual effects and content creation English studio that is most famous for collaborating on digital and design projects for advertising, gaming and musical spectacles. The studio was founded in 1990 as the first visual effects company in Europe to use exclusively digital methods.



Top:
The Mill crew working during the event.

Bottom:
The Mill. A Night of Spiritual Jazz Installation,
2016.
Photo by Drew Gurian.

In 2016, *The Mill* was called by *Red Bull* to create a visual installation to collaborate with live performances of three different jazz artists for the “*Night of Spiritual Jazz*”. The group developed a system, *Cosmograph*, that generated wormholes and patterns inspired by the psychedelic and improvisational vocabulary of *Spiritual Jazz*.

For creating shapes, the visual effects team tapped the audio from instruments such as the lead saxophone, the drums and the piano as a way to collect data and use it to affect visual variables uniquely, thus resulting in an improvised visual performance. Then, the custom software *Cosmograph*, built in *Cinder*, was responsible for producing the mystical geometries exhibited by processing the signal received from the bands’ instruments and using them as input for generative algorithms.

The project was led by the executive creative director *Rama Allen* who said that it was a great challenge to create real-time visual effects that suited well the music being played. In his own words, “it was a collaborative improvisation with the musicians where we were working to stay in ‘harmony’ with them the whole time. A challenging task with such complicated music and such wildly talented and experimental improvisational musicians.”



A very important aspect of contemporary creative coding tools is the level of abstraction they deliver. The many efforts made by the communities of *Processing*, *openFrameworks*, *Cinder* and of many other solutions culminated in a range of versatile and powerful tools that may suit the most diverse motivations.

It's also useful to point out that, similarly to any other computing technology, creative coding is very associated with the most relevant media options available at a given moment. The efforts of enabling *Processing* to run on an internet reality through *P5.js* and also the creation of a version, for example, demonstrate the commitment of a community that continues to work hard for delivering relevant solutions and to enable their own innovative projects.

By implementing built-in functions and programming dynamics that make it easier for even “non-technical” people to bring their ideas to life, these tools also allow developers, artists and designers to focus more on the experience than on technical issues. This ease of use also helps establishing a more collaborative and creative working dynamic as people can quickly prototype and test their creations.

INTERVIEW WITH JARBAS JÁCOME



Jarbas Jácome.
Photo by Fernando Vivas.

Jácome is a professional and academic researcher of computer art, computer music, electronic visual music, interactive art and experimental games since 2001. He also developed, with *ViMus - C++*, *OpenGL*, *OpenAL* and *Kinect*, computer art installations for exhibitions of the great Brazilian composers *Tom Zé* and *Gilberto Gil*.

Since 2011, he teaches computer programming for visual art students at *UFRB*, *Cachoeira*, *Bahia* and has been developing, since 2003, the *ViMus*, a free software for real-time integrated audiovisual processing with *C++*, *OpenGL*, *OpenCV*, *Boost* and *Multithreading*.

The following interview was originally conducted in Portuguese and later freely translated into English.

1) As a computer scientist, how was your transition to the field of art-technology?

Briefly, this transition wasn't actually a transition because since I was a child I've lived with both art and technology. Since very young, I've had a huge involvement with music, which naturally always kept me close to art. On the other hand, at an early age I was already interested in computing: when I was 13, I remember watching a program that explained how computer graphics had been used to create the dinosaurs of *Jurassic Park* and also how they had created that tornado for the movie *Twister*. Watching those documentaries impressed me, and since then I have decided that I wanted to study computing to explore its applications into arts and into the creation of other universes.

In the late 90s, when I was preparing for college, I've chosen to apply for the *UFPE (Universidade Federal de Pernambuco)* because of the music scene that was taking shape in the city of Recife. When I started studying computer science, I still had in mind that I wanted to work with a more artistic application, and that was the moment I've discovered musical computing and ended up deciding that it was going to be my focus on both my Bachelor's and Master degrees.

More academically speaking, some experiences I've had also made me decide to follow the art-technology field. In 2002, I started experimenting with real-time sound and image processing and that really interested me; and a fun fact is that I've found out this interest at a moment I was very disheartened by the graduation for feeling that computer science wasn't a good match for me

and for having the desire to become a musician.

At the height of this “depression” with my graduation, I was attending an elective of musical computing for which I had to develop a waveform viewer. Although computationally I found the visualizer project a triviality, as I was doing it I began to realize that by using concepts that we saw in a computer science course, it was possible to create some crazy things that few people did here in Brazil.

More institutionally speaking, one thing that pushed me into this area was to have been encouraged by Professor *Geber Ramalho* to apply a project for the art-technology “*Rumos Arte Cibernética*” contest announcement hold by *Itaú Cultural* in 2006. Having won this award gave me a lot of visibility within the artistic community and made me begin to create a contact network in the area.

2) If you could put it in words, how would you describe your creative process (from initial planning to development)?

The process is not always the same, but in general there are some things that always happen. For instance, I can say that I tend to start working with an idea that emerges and that I was not necessarily consciously seeking. Since I’ve already had other sources of income, I could establish my own routine and respect my time, rather than becoming a “creating machine”. In other words, I could be carried away by extreme spontaneity.

Talking about the creative process itself, I always plan the experience first, whether if it’s interactive or one that proposes that people only watch or listen to something. I conceive the scene on my mind, as if I closed my eyes and imagined how it is supposed to happen.

At the same time, I start to do some programming experiments and also employ the technical challenges I encounter as guides: iterating through this cycle of coding and evaluating how the current technical implementation is creating the final experience, until it reaches a satisfactory one.

I also often hear criticism from friends and colleagues. For instance, when I was designing my 2006 work “*Crepúsculo dos Ídolos*” (*Twilight of the Idols*), at first I thought about an interaction that allowed viewers to scream and, through this scream, distort an image on a television. Then, a more experienced friend of mine advised me to keep working on the original idea as to improve the experience delivered by the artistic object proposed. It was after receiving his advice that I ended up reaching the final version in which the TV’s image is replaced by the own image of the person screaming.



Jarbas Jácome. *Crepúsculo dos Ídolos*. 2006.
Photo took during the 2006 FILE (Electronic Language International Festival) event.

3) Which languages or tools were the most significant for you during your trajectory?

The first tools I came into contact with in life were the *MS-DOS* and some simple diagram programs such as *FlowChart*. I only came to learn how to program in 1996 when I was already 13 or 14 years old and got to know *Windows QBasic*. With *QBasic*, it was possible to study programming through a kit that accompanied the operating system; something that was nice because it worked just like some kind of *Processing* with which we could learn by following examples and easily implementing new objects. I also started to improve my programming techniques when I started to play, in *QBasic*, with some physics concepts I was currently learning at school.

In the late 90s, when internet became a hit, I bought a *Javascript* book and also took some *HTML* classes. One thing I found to be very nice about *Javascript* at the time was that I could apply the same logic as *QBasic* into *HTML* elements. Nowadays, I also understand that in fact, I was very privileged at that time because few parents could actually afford these programming courses for their children. I also worked with web development eventually for making some money.

In 2000 I joined the computer science course and was part of one of the last classes that attended to introductory disciplines in *C* instead of *Java*. I was very lucky for it because we were forced to learn pointers, which is very nice for providing consciousness of how to operate directly the memory of the computer. Something even better is that this greater knowledge of the functioning of the computer is some kind of hype for contemporary computer graphics as *Vulcan* is taking *OpenGL's* place.

From that moment on, I started to use *C++*, mainly because my whole college class entered the trendy *Java* and object-oriented paradigm. I found *C++* to be very good because it coupled performance with the possibilities of abstraction through objects and also because it was a common language choice to work with *OpenGL*. Since I wanted to work with computer graphics, I ended up assuming *C++* and *OpenGL* as main tools for me.

More ahead, when I was getting my Master degree in musical computing, I was forced to learn *Pure Data*, a language I came to see that had a very great power of expression and that was amazing. No wonder it's currently the default language in *MIT* for anyone who wants to learn musical computing, for example.

Just like I was forced to learn *Pure Data* for working with musical computing, I was forced to learn *Processing* so I could teach art-technology topics. I've already wanted to teach using *C++*, but I think *Processing* shortens the path and makes it easier to learn, since you only have to press play in the *Processing* IDE in order to have a program running.



A spectator interacting with Jácome's "Vitalino".
2008.
Photo took during the 2008 FILE (Electronic
Language International Festival) event.

Finally, for art today, I develop my works in C++. And during my PhD in 2019, I will continue to develop *ViMus*, a software that I have been building since graduation, to teach visual arts.

4) Which languages or tools are currently the most relevant for the art-technology community?

When working, I often prototype using *Processing* and implement using C++. I also employ *Processing* when teaching and I often ask my students to develop their projects in *Processing* or *Pure Data*. I think the general community ends up adopting these choices.

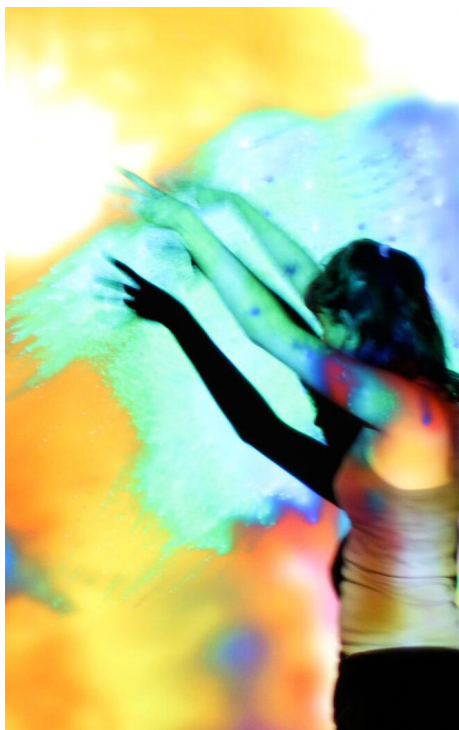
5) Could you comment, in terms of technology and algorithms, on some of your favorite projects or some that have been more challenging?

"*Vitalino*" (2011) was one my favorite projects because it was the one that took me further (Taiwan), where I taught an *openFrameworks* workshop. It's a work I appreciate to see running and also one that shows how coding is a great poetry opportunity.

With *Vitalino*, the visitor could make a digital sculpture by moving his or her fingers and having these movements captured by two webcams placed perpendicularly on a structure with controlled lighting. The project consists of a cube subdivided in several other smaller cubes (*voxels*), and is practically composed of 0 or 1: 0 if there is a little cube there and 1 for the opposite. Then, the visitor could tear these small cubes apart with the hand and obtain, through the projection, a sculpture by subtraction, the result of the modeling the person is doing.

In the honor of *Vitalino* (a craftsman from Pernambuco, Brazil), the name of the variable that holds the three-dimensional matrix we used to store the sculpture information is "clay", because it's just like as if this clay was a place on memory that the person is using to model the sculpture displayed. And I think this is how we can attract people from the field of computing: showing them the poetry that exists in the things they develop.

One of the most challenging projects I've ever done was the "*Net*" for *Gilberto Gil* (a Brazilian composer), this work was exhibited during the "*Gil70*" in 2013 and had been shown in 4 different cities. One point I find interesting when working with art-technology is that things that are frustrating for artists are not necessarily the same things that are frustrating for computing people, and vice versa. Although I feel enchanted by the technical challenge, it's sometimes hard for me to get involved in some project that is technically challenging, but that ends up resulting in a not so interesting aesthetic outcome. I think the "*Net*" was the most difficult project I've ever done because it basically



Top:
Ernesto Klar. *Convergenze Parallele*. 2007.
Photo took during the 2007 FILE (Electronic
Language International Festival) event.

Above:
Memo Akten. *Body Paint*. 2010.
Allowing people to paint a canvas with their body.

consisted of many analytical geometry operations that had been very time consuming for me.

When working on a more recent project I've developed for *Tom Zé* (another Brazilian composer), the challenge was to achieve a good performance. I've had to manipulate a *Kinect* library so it could run on multi-core and perform better.

6) Are there some specific art-technology works that you appreciate a lot?

First I'll mention *Aaron Koblin*, who created the *Johnny Cash* video clip. He distributed the frames of the video through the internet for people to draw whatever they wanted on them. I found this work very interesting because of that.

Some other artist's work and also the one that pushed me into art-technology was a work created by *Ernesto Klar*, a Venezuelan professor currently at *Parsons*, New York. This work was a kind of augmented dust with which people could interact. There was a high resolution camera that captured this dust's movement after it was blown by the spectator, and from this movement the computer created a projection as if it was a visual dust. I first saw this work at *FILE São Paulo (Festival de Linguagem Eletrônica)* in 2007.

I'll also mention *Mary Ellen Bute*, a woman that created, in the 1940s, some things people create today in *Processing*. She was one of the pioneers in *Visual Music* of the twentieth century, but ended up dying poor in New York.

7) Which artists or programmer inspire or have inspired you the most?

I liked that in your question you wrote "artists or programmers" because by doing this you include those artists who are more programmers than artists, and I think that I'm more programmer than artist.

A guy that could be some sort of guru for me is *Miller Puckette*, the creator of *Pure Data*; and *Linus Torvalds*, the creator of *Linux*; both are people that inspire me a lot!

Specifically from this art-technology area, a person that inspires me a lot is *Memo Akten*, who has several awards and created a lot of works employing artificial intelligence, electronics and computer vision. *Akten* is from the *demoscene* community, and a lot of people from the field of computing ended up falling into art because of the *demoscene*.

I could also mention *Juliana Cerqueira*, who creates art by hacking software and also another artist I've met called *Rhazes Spell*.



Mary Ellen Bute. One of the pioneers of visual music.

8) What would be your advices for a computer scientist who is interested in art and would like to follow the art-technology path?

I think a good first tip is playing both games: to understand how people from the computing field legitimate things and also how artists do this; also to understand the specificities of the two areas and use one thing to attract the other. In general, people from the arts are quite “techie”, there is a tendency to admire and idolize technology and that could result in many interesting projects; on the other hand, computing people tend to be a bit arrogant because they see themselves as the people who run the planet today. These two groups end up isolating themselves in their own beliefs and methods and end up missing opportunities to work together.

A more technical advice is to focus on the same thing for a long time: spend months programming and experimenting with the same concept until reaching a satisfactory result. Many times I have alternated my musician’s life with nights spent in labs working on a single project. I give this advice, mainly because if you want to enter this artistic world, you will have to show through your work that you have dedicated many hours to that project, artistically evaluating the effect of every single detail.

It’s also necessary to learn how to navigate in various scenarios and contexts: to get in touch with the art-technology academic community, which is a group of people that get together, create their own niche and strive for it to be increasingly legitimized by the contemporary art community; promote yourself to industry personnel, get investments and show the symbolic value of instigating people’s emotions and how that can change their relationship with brands; there are also many banks that invest heavily in this, such as *Itaú* (a Brazilian bank) that holds several events and awards; and the people from telecommunication companies that also hold some art-technology events. The advice is to recognize that they are very specific groups and that we have to learn to negotiate with them, speak their language and show that we can grow with each other.

To finish this section: I spent very individually a great part of my trajectory, thinking about my works and consolidating them. In the last few years, I’ve started to teach a lot of workshops and spread the knowledge, very rewarding moments that only happened because of the time I spent creating my own identity and positioning myself within art-technology.



Rhazes Spell. THINK. 2011.
Photo by Mirada Studios.

SELF CREATED EXPERIMENTS

The following experiments were developed with the intention of illustrating and practicing the concepts and techniques studied during the creation of this undergraduate thesis. All of them are generative and algorithmic art pieces created entirely with *Processing*.

Cloudy Sky Simulation





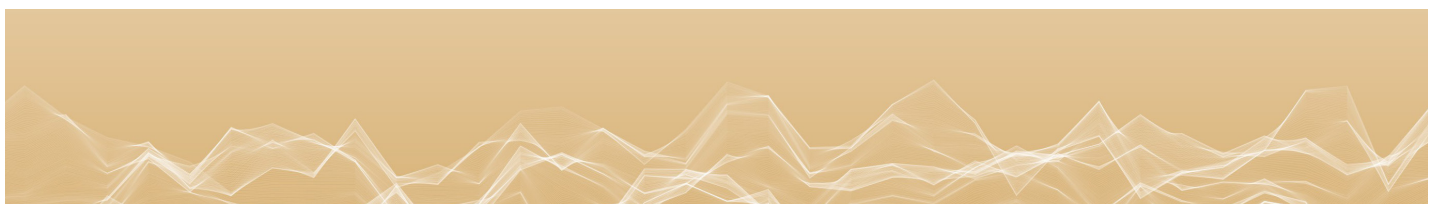
A study in pixel arrays and layer blend modes for generating a vivid cloudy sky simulation.

The clouds are created by randomly defining an initial frame in which pixels are colored in many shades of gray (picture below). The animation is obtained by sequentially updating the location of these already colored pixels. In other words, a pixel array stores the location of each pixel to be updated as frames are displayed.

Initially, points are colored so a “gray sky” is obtained. Then, to obtain the final result, a colored layer is painted over the whole image but with its blend mode set to “Screen”. During the animation, the color layer keeps changing according to the HSB color model as to obtain different but harmonic sky tones.



Sea Waves or Mountains





Exploring the overlap of several lines, *Sea Waves or Mountains* creates two different ambiences by relying on a color change and on different initial parameters of the same algorithm.

Lines are created by establishing a series of connection points that fill the whole screen width. The very aesthetic nature of the results are obtained by determining the X coordinates of a line's set of connection points according to a fixed interval but randomly determining the Y values of each one of them.

By drawing each line with a varying opacity, a more fluid effect was obtained.





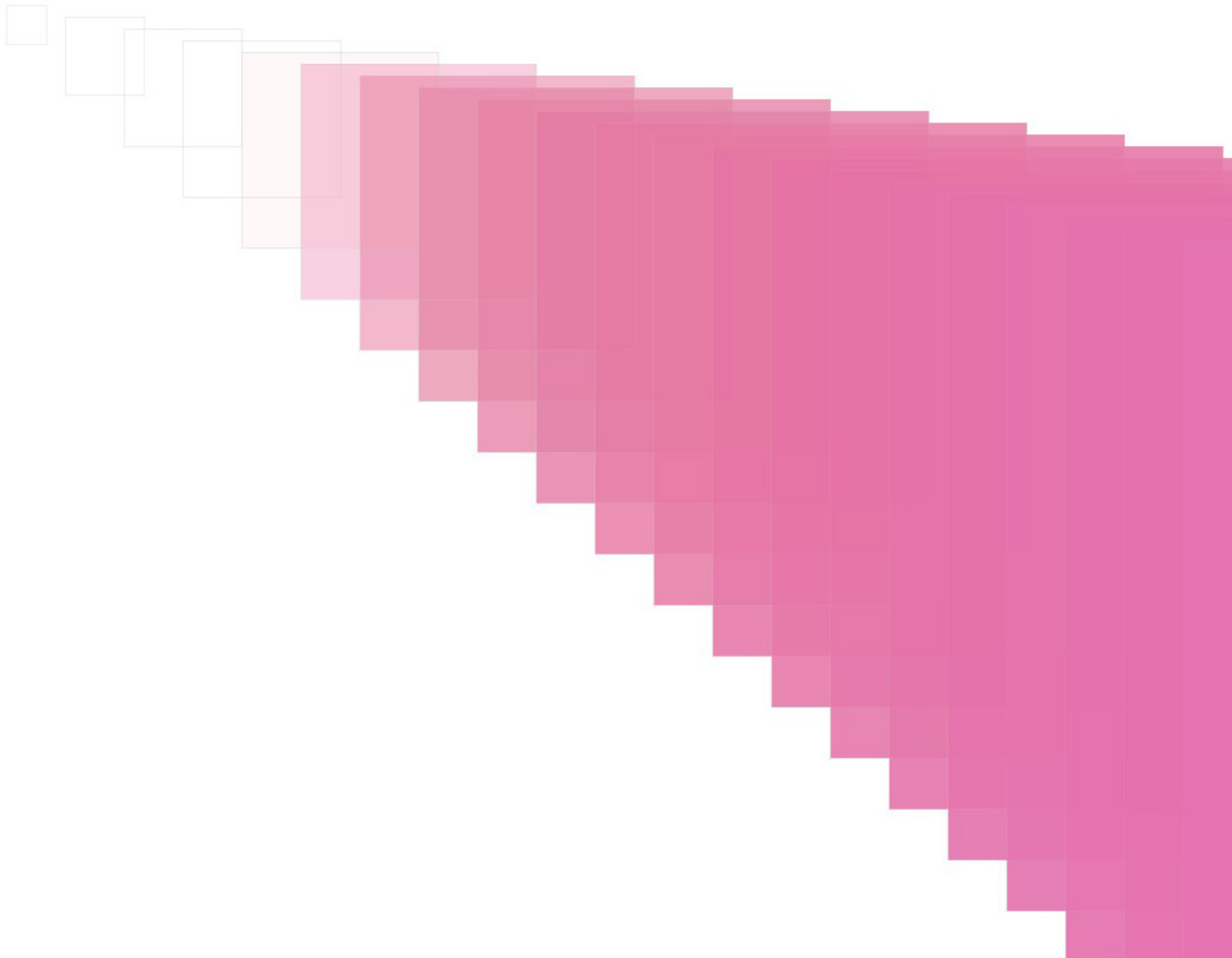
Perspective Studies



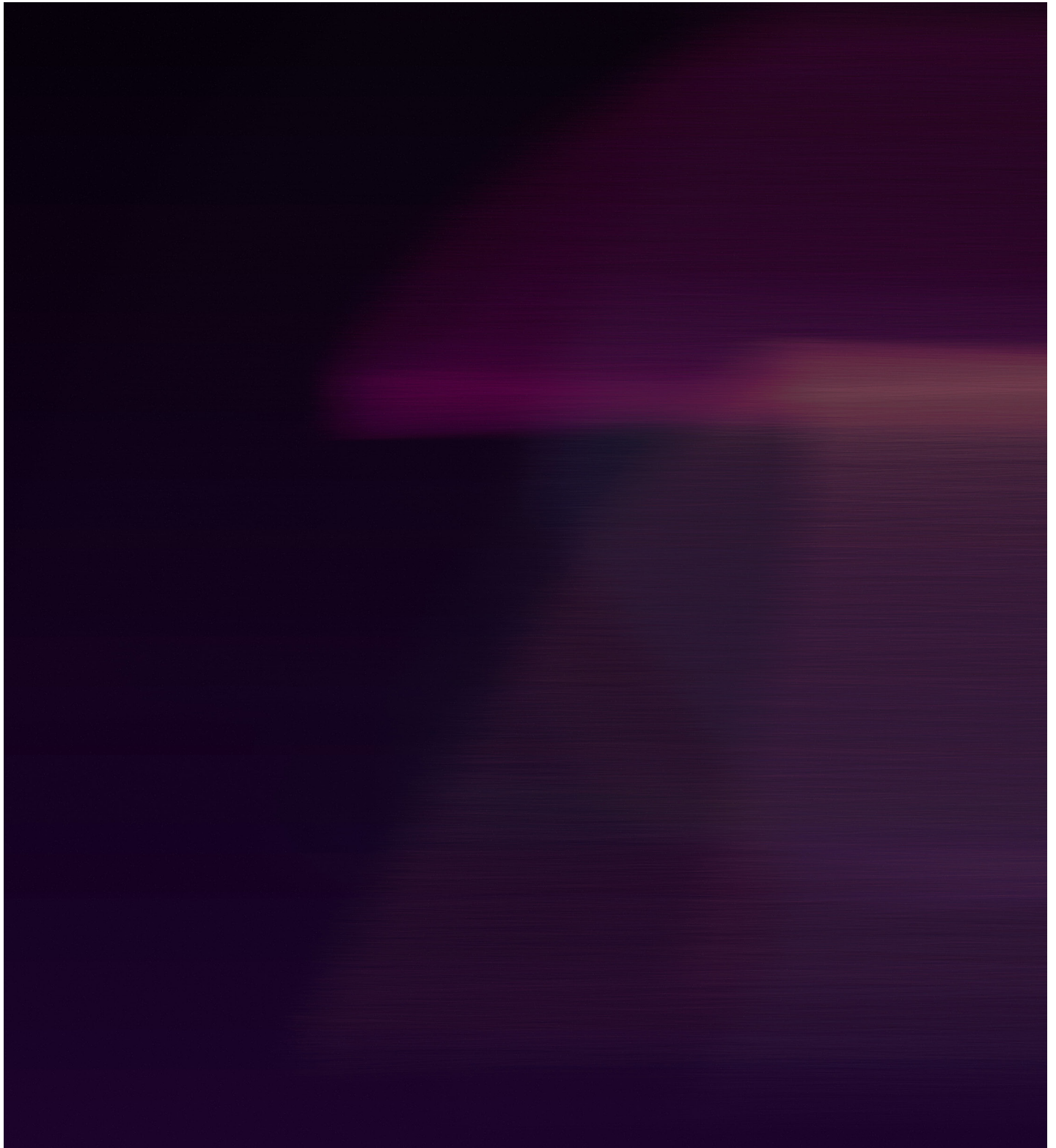
The *Perspective Studies* algorithm relies on parametric equations of line segments between start and vanishing points, both pre-specified.

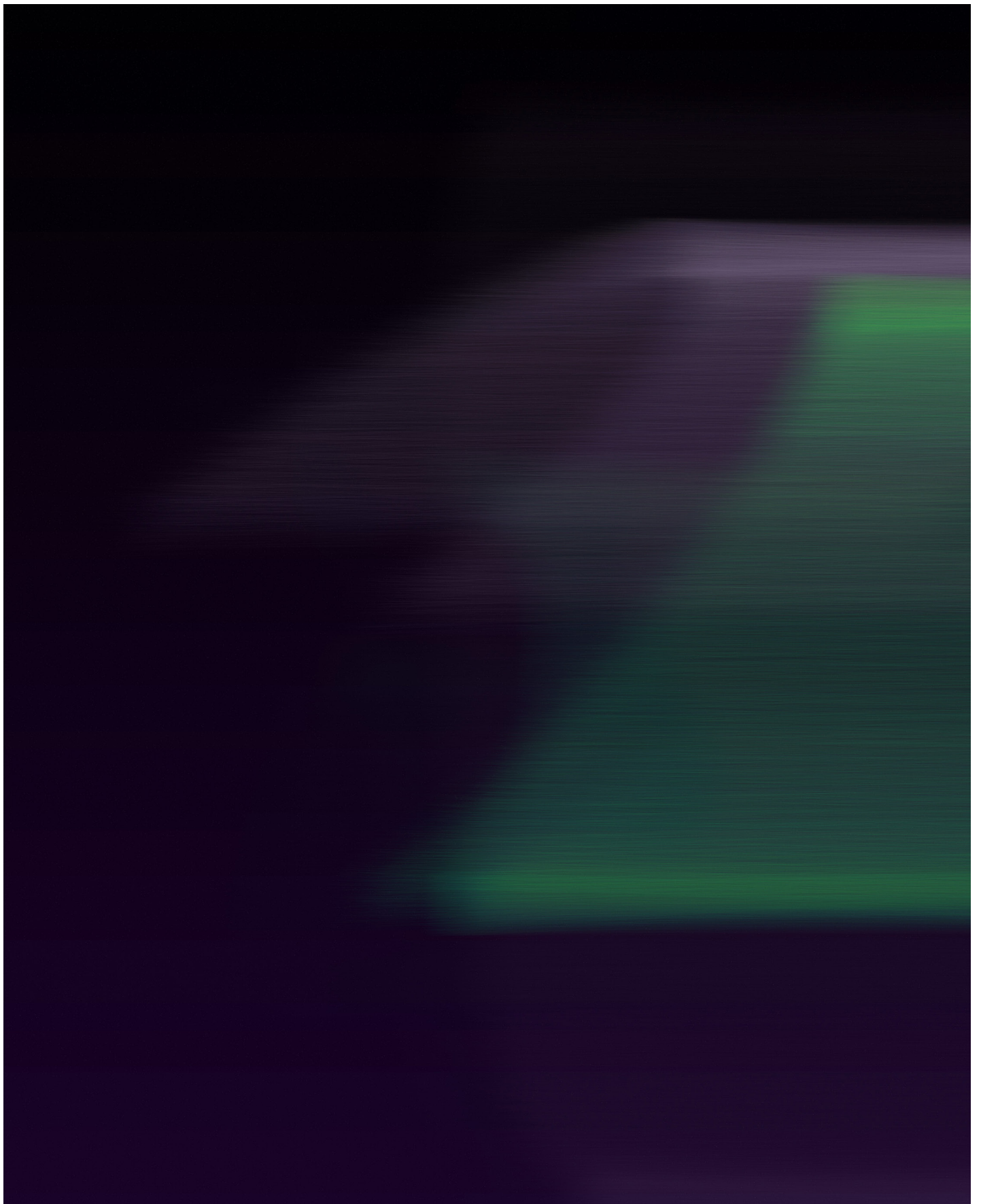
As an attempt to exercise the perspective concepts discussed on the first chapter of this project, *Perspective Studies* allows users to define a squared shape and also a desired vanishing point and let the algorithm draw the correct perspective.

By drawing a series of colored sub-squares resized to fit the correct perspective, the final result is a vibrant perspective study. The user can also define how many squares will compose the final shape, thus controlling how condensate the final object will be.



Northern Lights





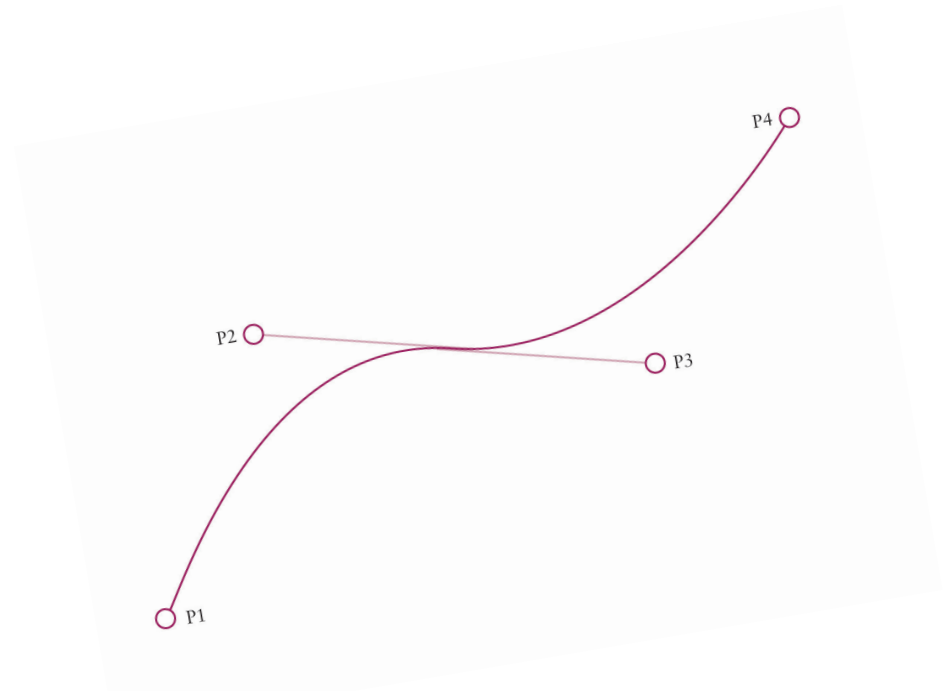
An algorithm that relies on Bézier curves for creating shapes inspired by the northern lights phenomenon.

Bézier curves are parametric curves often used in computer graphics related applications. Each curve is created by establishing a finite set of control points that determine its curvatures.

In *Northern Lights*, the algorithm first randomly determines a given number of control points and then employs the *De Casteljau's Algorithm*, a recursive method to evaluate Bézier curves polynomials, for effectively drawing each curve.

The very characteristic nature of the series is obtained by using the *De Casteljau's Algorithm* for drawing lines instead of simply drawing the points of a Bézier curve path. It's also necessary to point out that a great part of the effect obtained is due to the color palette chosen and to the determination of other aesthetic attributes such as opacity and saturation.

The picture below is an example of a cubic Bézier curve formed by its control points $P1$, $P2$, $P3$ and $P4$.



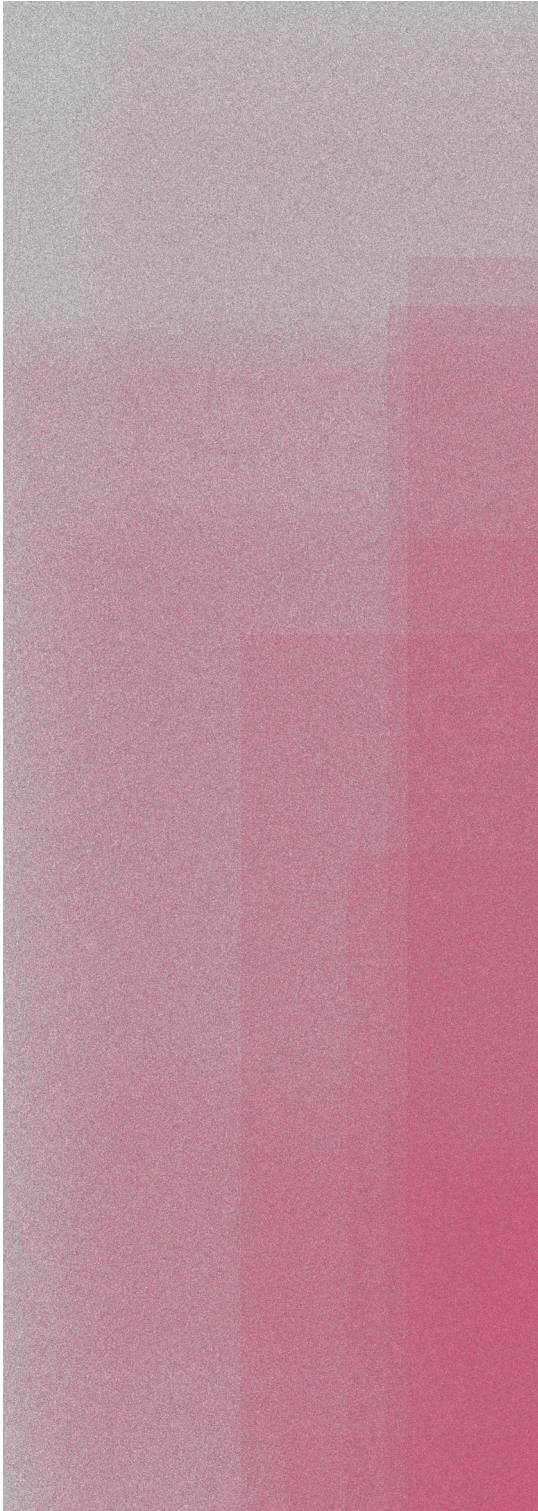
Colored Superficies

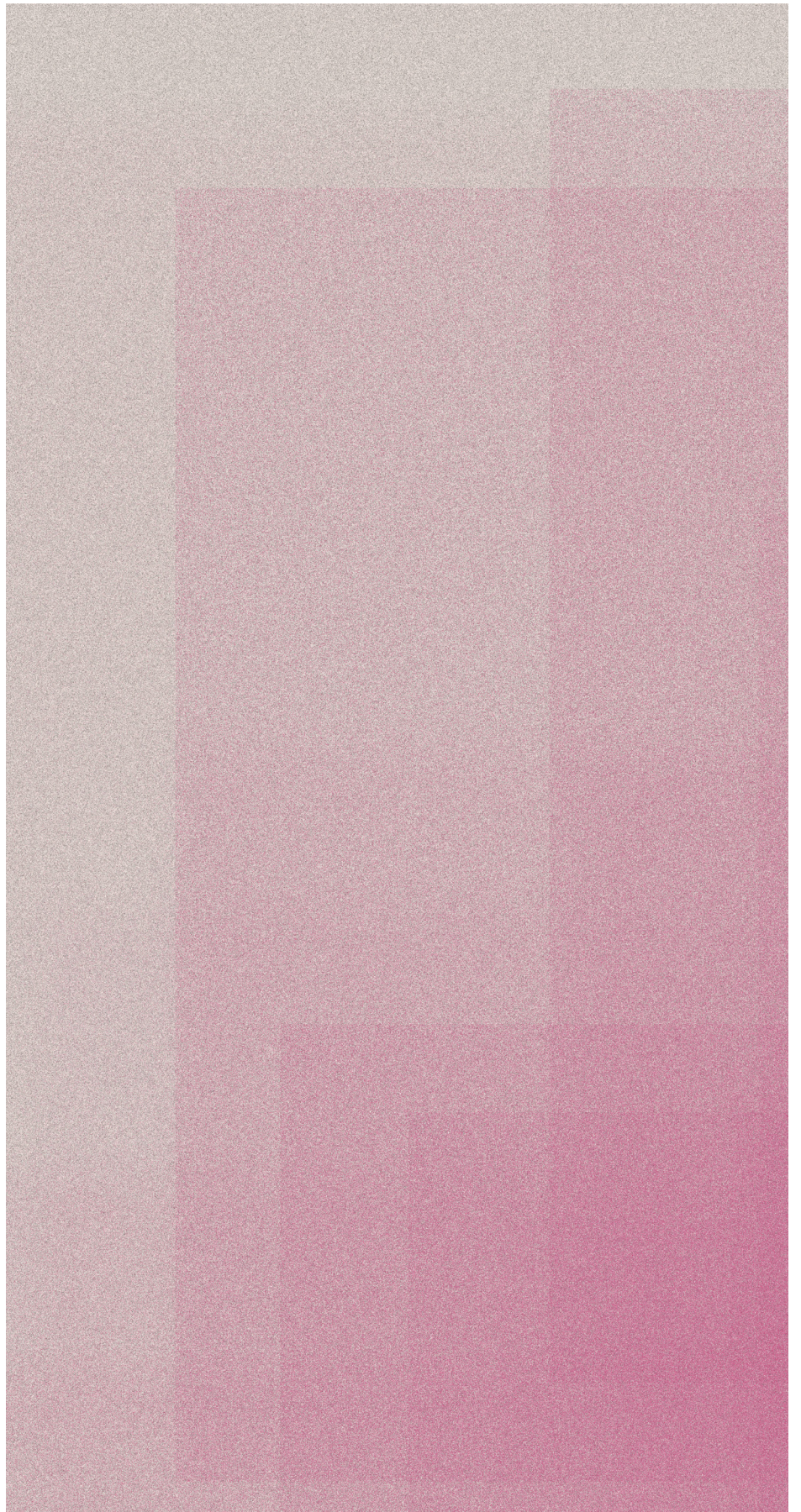
Creating patterns through “attraction areas”.

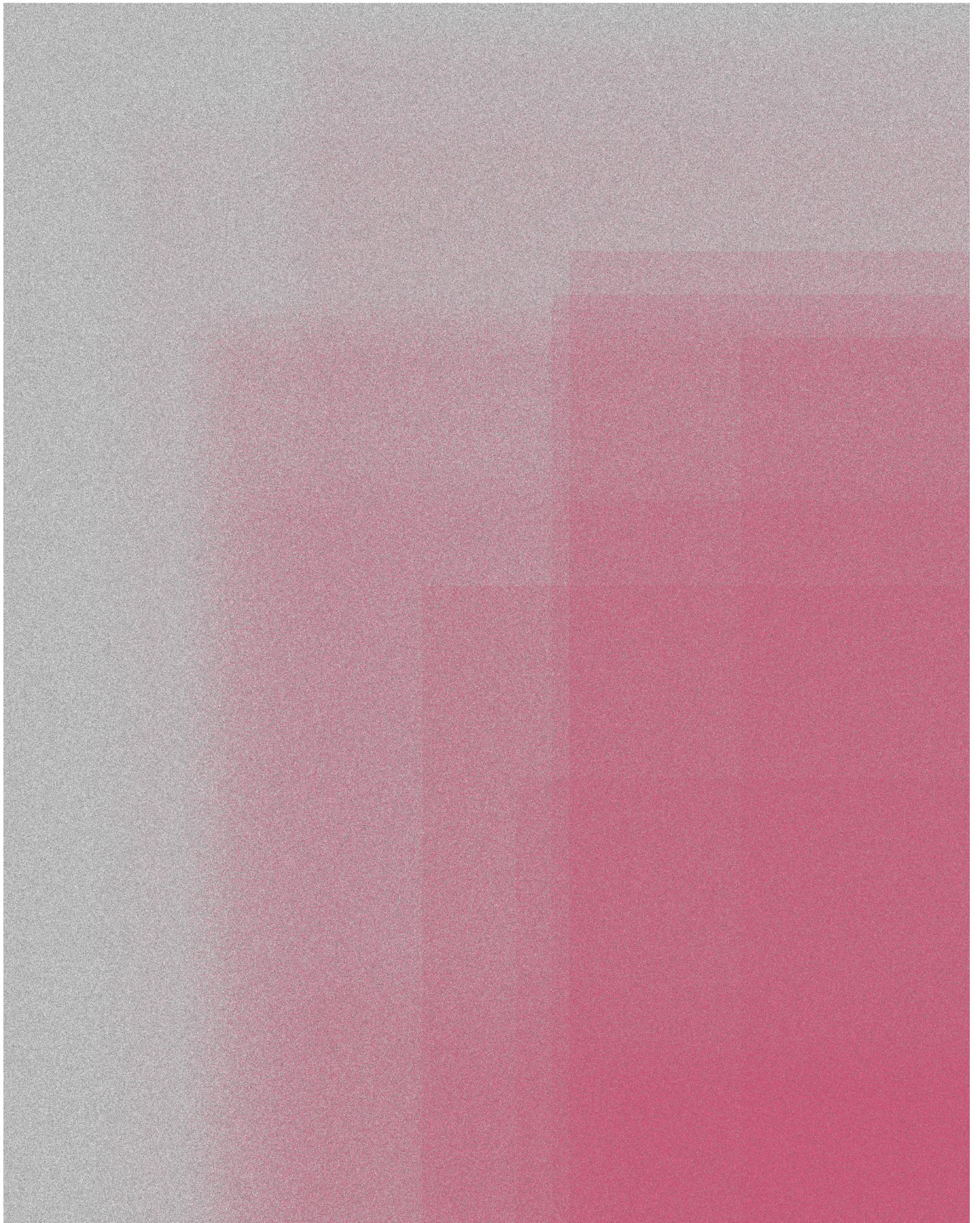
After being set with a given number of attraction areas, an attraction radius and other parameters, the algorithm iteratively draws a huge number of points that, actually, have a bigger probability of being drawn on these specific zones.

Part of the “layered effect” is obtained by randomly choosing the opacity a point will have according to the attraction area in which it’s drawn. The other part is obtained by randomly choosing some of the attraction areas to be drawn according to a given noise, thus resulting in a faded aspect.

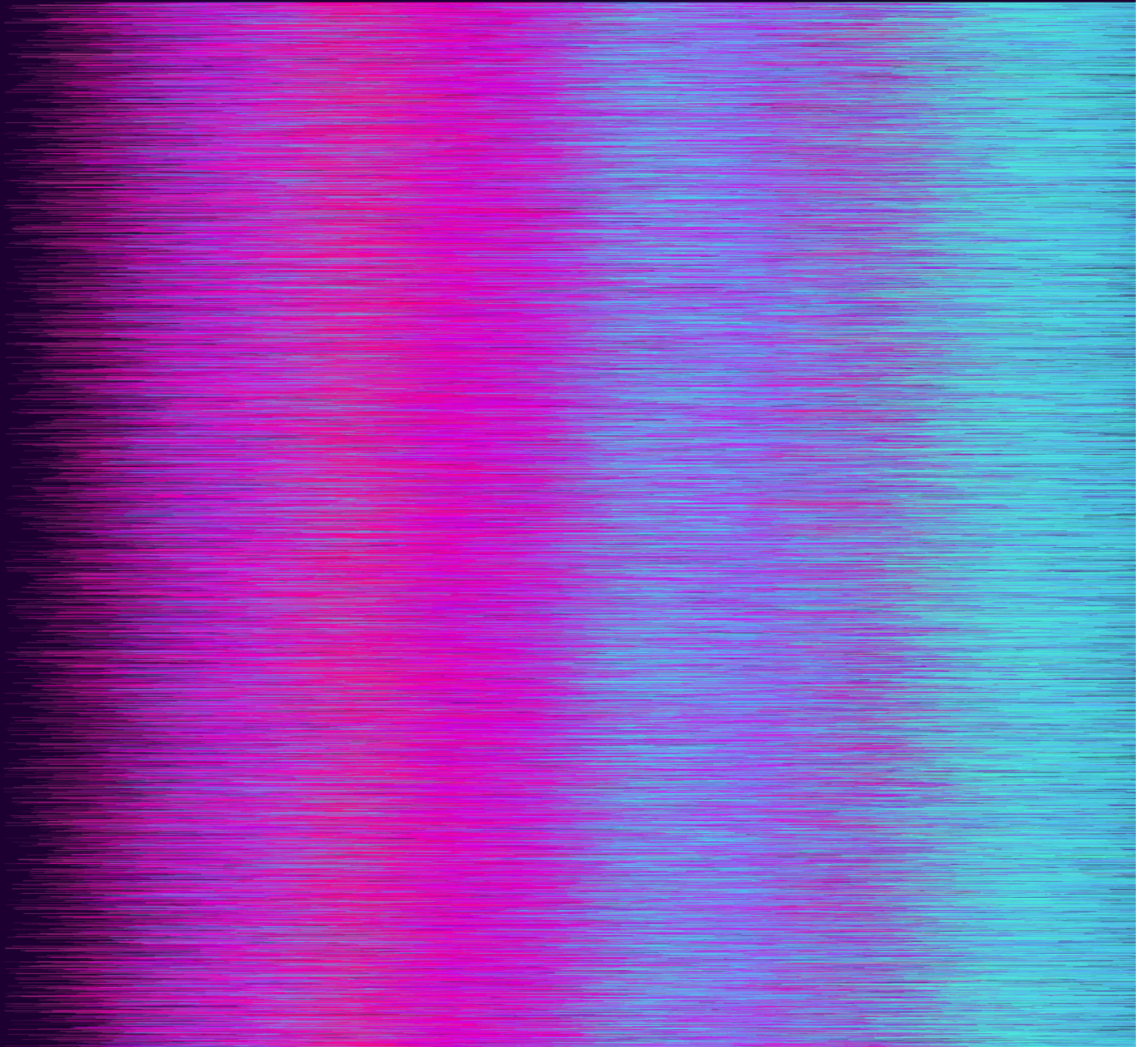
An also important parameter of the algorithm is the attraction rate, a value between 0 and 1 that determines how many points must be attracted, thus drawn on an attraction area, and how many must be freely drawn in any location of the screen. An attribute that allows the algorithm to control the level of contrast of a final composition.

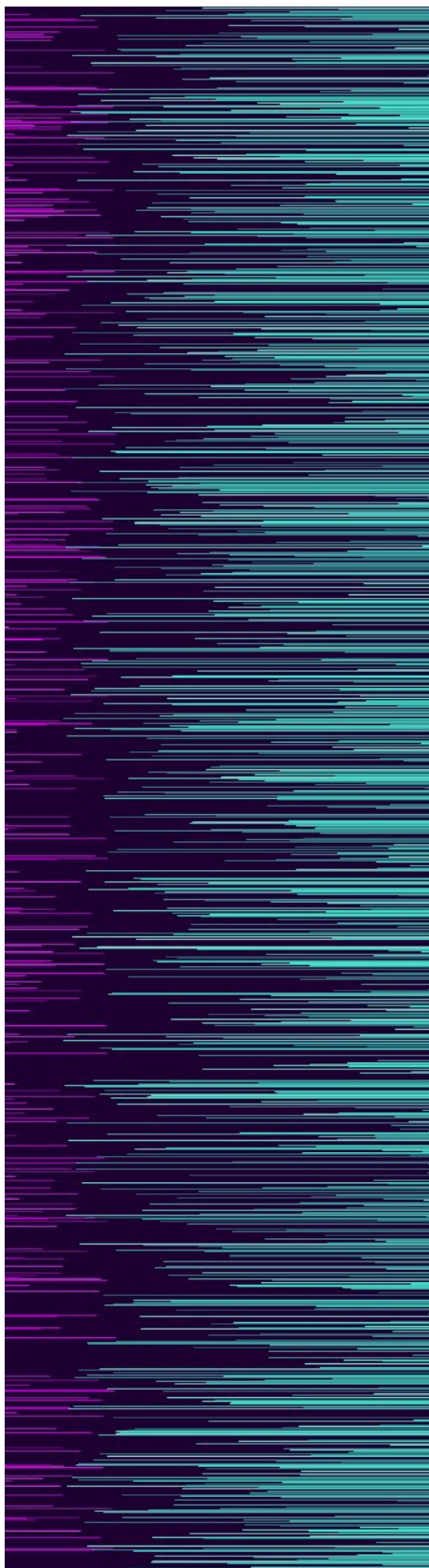






Horizontal Energy

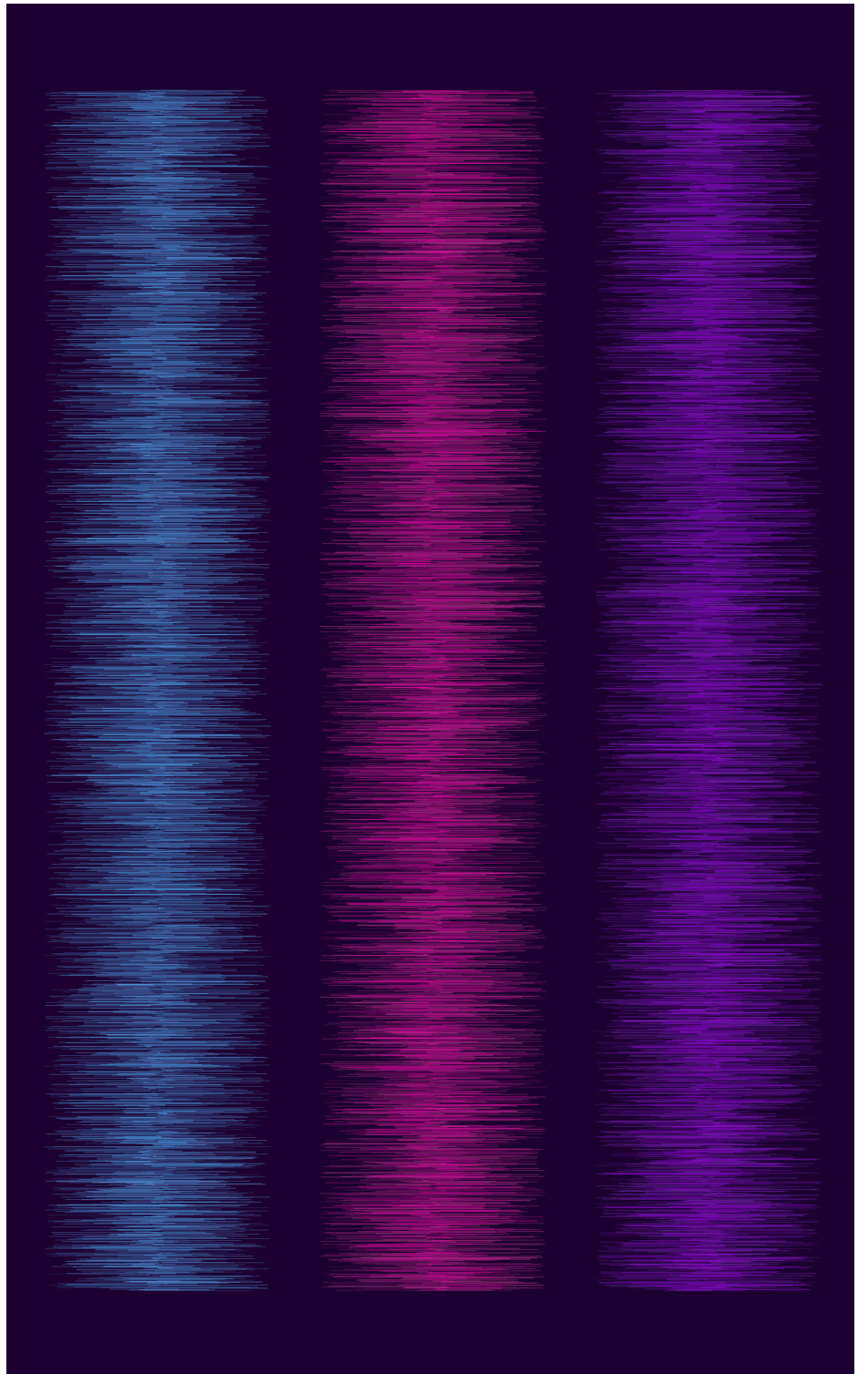




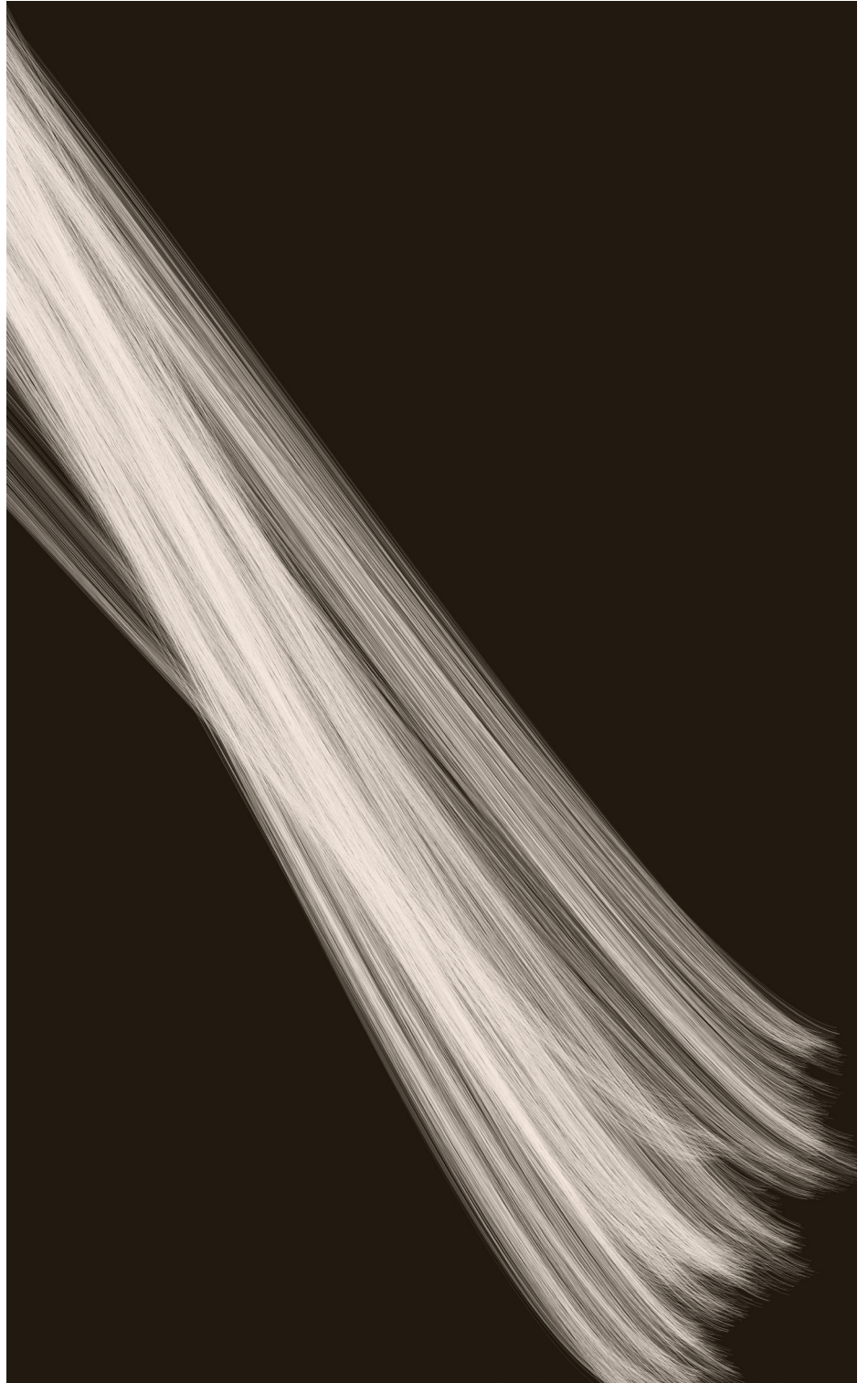
Using the overlap of many colorful and translucent lines and a specific color palette for creating a vibrant composition.

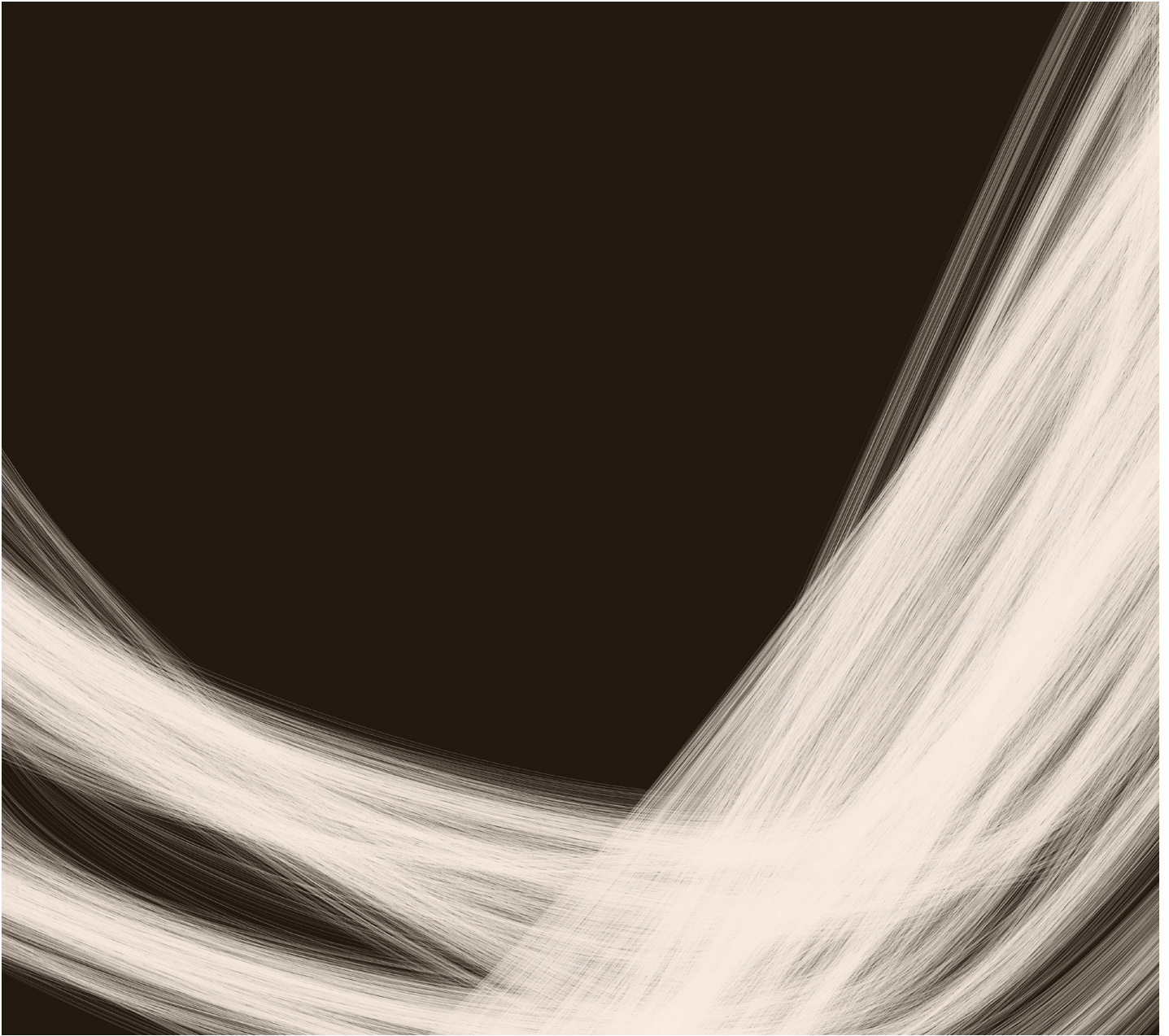
In *Horizontal Energy* the algorithm goes through the entire screen width, according to a pre-specified interval, and draws a given number of lines. The very nature of the artwork is obtained by changing the stroke color as each width interval is reached and also by playing with shape parameters such as the length of the lines and their opacity.

An important role is also played by the energetic color palette chosen for the artwork. The red, violet, pink and blue tones create the desired vivid aspect and give the whole composition a final touch.



Curved Seduction



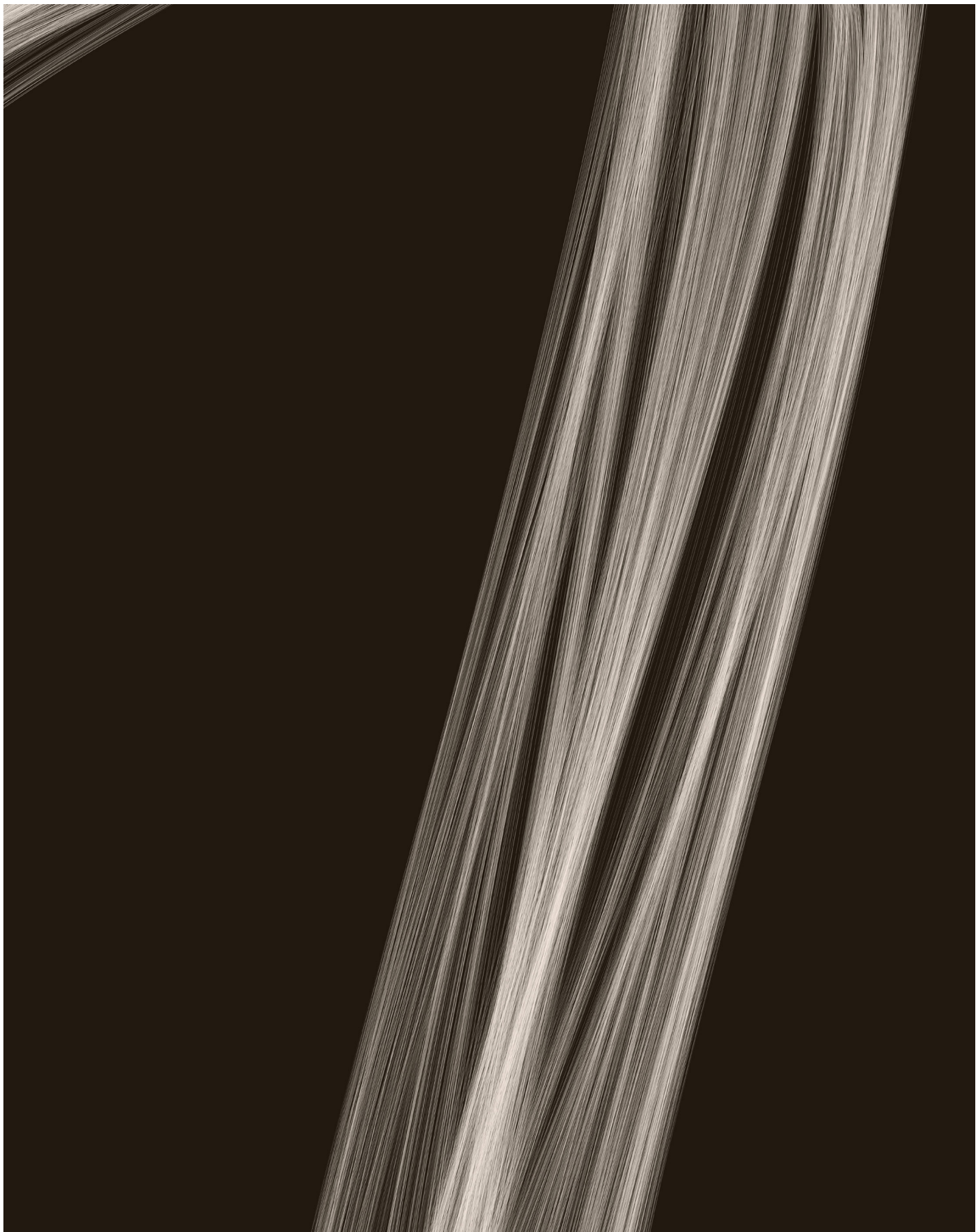


Curved Seduction plays with curve control points and properly chosen aesthetic attributes for creating visual appealing shapes.

The algorithm creates the shapes by overlapping a huge number of translucent curves. First, the algorithm chooses the number of control points that will define the current curve and then randomly determines each one of these points.

The very characteristic nature of the artwork is obtained by iteratively drawing curves but slightly varying their previously defined control points, thus resulting in similar shapes. Every time a new curve is drawn, the opacity is also randomly defined, contributing for the obtention of the “silk” effect.





Bibliography

_Fishwick, Paul A. *Aesthetic Computing*. Cambridge: Mit Press, 2006.

_Santaella, Lucia. *Arte e Tecnologia. Modus operandi universal. A relevância da arte-ciência na contemporaneidade*. Brasília: Instituto de Arte da Universidade de Brasília, 2012.

_Popper, Frank. *From Technological to Virtual Art*. Cambridge: Mit Press, 2007.

_Jankel, Annabel; Morton, Rocky. *Creative Computer Graphics*. Cambridge: Cambridge University Press, 1984.

_Taylor, Grant. *When the Machine Made Art: The Troubled History of Computer Art*. London: Bloomsbury Academic, 2014.

_Pearson, Matt. *Generative Art*. Manning, 2011.

_Reichardt, Jasia. *Cybernetic Serendipity : The Computer and the Arts : A Studio International Special Issue*. London: ICA, 1968.

_Blais, Joline, IPPOLITO, Jon. *At the Edge of Art*. Thames & Hudson, 2006.

_Klanten, Robert; Ehmann, Sven; Hanschke, Verena. *A Touch of Code: Interactive Installations and Experiences*. Gestalten, 2011.

_Gray, Alfred; Abbena, Elsa; Salamon, Simon. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. Chapman and Hall/CRC, 2006.

_O'Connor, John; Robertson, Edmund. *Mathematics and Art: perspective*. 2003.

_Galanter, Philip. *What Is Generative Art?: Complexity Theory as a Context for Art Theory*. 2003.

_Esaak, Shelley. *Understanding Proportion in Art*. ThoughtCo, Jun. 22, 2018. thoughtco.com/proportion-definition-in-art-182453.

_Wever, Sarah. *Media Lab Intern Spotlight: Exploring Algorithms in Islamic Art through Augmented Reality*. The MET, Aug. 26, 2014. metmuseum.org/blogs/digital-underground/2014/exploring-algorithms-islamic-art.

_Molnár, Vera. *Atari Archives: Vera Molnár interview*. Tihany, France, Aug 1975. atariarchives.org/artist/sec11.php.

_compArt daDA: the database of Digital Art. Georg Nees. 2018. <http://dada.compart-bremen.de/item/agent/15>.

_compArt daDA: the database of Digital Art. Frieder Nake. 2016. <http://dada.compart-bremen.de/item/agent/68>.

_DAM: Digital Art Museum. Frieder Nake. <http://dam.org/artists/phase-one/frieder-nake>.

_Lillian F. Schwartz: Biography. 2013. <http://lillian.com/biography>.

_Heller, Eric. Eric J Heller Gallery. <http://jalbum.net/en/browse/user/album/1696720>.

_the fuse*. MULTIVERSE. 2018. <https://www.fuseworks.it/en/works/multiverse/>.

Credits

_8: Spike, John T. Masaccio, Rizzoli libri illustrati. Milano. 2002.

_9.t: Musa - Musei Statali Di Arezzo (<http://www.museistataliarezzo.it/museo-san-francesco>).

_11.b: Lines & Marks: Albrecht Dürer (<http://linesandmarks.com/albrecht-durer/>).

_13.t: Official Escher's Website (<https://www.mcescher.com>).

_14: Picture and building steps: School Of Islamic Geometric Design (<http://www.sigd.org/resources/pattern/pattern-1/>).

_15.t and 15.b: Betty Quinn's Official Website (<http://www.bettyquinn.com/islamicalgorithms.html>).

_16.l: Giorgi Dalakov's History Of Computers Website (<http://history-computer.com/ModernComputer/Software/Sketchpad.html>).

_18.l and 18.r: Reichardt, Jasja. Cybernetic Serendipity : The Computer and the Arts : A Studio International Special Issue. London: ICA, 1968.

_19 and 20: The Database of Digital Art: Donald K. Robbins (<http://dada.compart-bremen.de/item/agent/310>).

_21.r and 22: Jankel, Annabel; Morton, Rocky. Creative Computer Graphics. Cambridge: Cambridge University Press, 1984.

_23.t: Eletronic Visualization Laboratory's Website (<https://www.evl.uic.edu/entry.php?id=1113>).

_23.b: Star Wars: Episode IV – A New Hope: Making of the Computer Graphics for Star Wars (<http://www.movieweb.com/movie/star-wars-episode-iv-a-new-hope/making-of-the-computer-graphics-for-star-wars>).

_24 and 25.t: Jankel, Annabel; Morton, Rocky. Creative Computer Graphics. Cambridge: Cambridge University Press, 1984.

_25.b: SIGGRAPH Digital Archive (digitalartarchive.siggraph.org).

_29.b: Free Art Bureau (http://freeartbureau.org/fab_activity/the-computer-from-hell/).

_30.r: Bense, Max; Nees, Georg. rot 19. Computer-Grafik. Stuttgart: Max Bense, Elisabeth Walther. 1965.

_31.t: Professor Horst Konrad Zuse's Website (<http://www.horst-zuse.homepage.t-online.de/index.html>).

_31.r: Victoria and Albert Museum Collection (<http://collections.vam.ac.uk>).

_32: Digital Art Museum - Frieder Nake (<http://dam.org/artists/phase-one/frieder-nake/artworks>).

_33.b: Senior & Shopmaker Gallery (<http://www.seniorthandshopmaker.com/artists/vera-molnar/>).

_34.l: Digital Art Museum - Vera Molnar (<https://digitalartmuseum.org/molnar/144trapezes.html>),

_35.t and 35.b: The Anne and Michael Spalter Digital Art Collection (<http://spalterdigital.com/>).

_36.b, 37.t, 37.b, 38 and 39: Algorithms and pictures from Manfred Mohr's Official Website (<https://www.emohr.com>).

_40 and 41.b: Lillian Schwartz Official Website (<http://lillian.com>).

_42.t and 42.b: Magenta Plains - Lillian Schwartz (<http://magentaplains.com/exhibitions/lillian-schwartz>).

_44: Manfred Mohr's Official Website (<https://www.emohr.com>).

_45, 47 and 48: Caio Barrocal's Behance Profile (<https://www.behance.net/caiobarrocal>).

_49.t: Shape: Maurizio Martinucci (TeZ): Creative investigation in music, art, science (<http://shapeplatform.eu/2016/maurizio-martinucci-tez-creative-investigation-in-music-art-science/>).

_53.l and 53.r: Eric Heller's Gallery Website (<https://ejheller.jalbum.net>).

_54, 55.t, 55.l and 55.r: Congaz - Opening Ceremony Of The Olympic Games 2008, Beijing (<http://www.congaz.de/live-communication/project/opening-ceremony-of-the-olympic-games-2008-beijing/>).

_56, 57.t and 57.b: teamLab - Crows are Chased and the Chasing Crows are Destined to be Chased as well, Transcending Space (https://www.teamlab.art/w/crows_transcending_space/).

_58, 59.t and 59.b: Red Paper Heart - Game Of Thrones Sword Experience (<https://redpa->

perheart.com/work/swordexperience).

_60.t, 60.b, 61.t, 61.b, 62.t and 62.b: Thomas Sanchez Lengeling Portfolio - Aether (<http://codigogenerativo.com/works/aether/>).

_63, 64.t and 64.b: Visible - Paris Expo Porte de Versailles (<https://visible.si/project/paris-expo-porte-de-versailles/>).

_65, 66.t and 66.b: Uva - Array (<https://uva.co.uk/works/array>).

_71.t: Nóbrega's Official Website - Plausible, possible, potential (<http://www.nobrega.info/index/plausible-possible-potential>).

_71.b: Nóbrega's Plausible, possible, potential Video (<https://vimeo.com/143076578>).

_72.t, 72.f and 72.r: Nóbrega's Official Website - Plausible, possible, potential (<http://www.nobrega.info/index/plausible-possible-potential>).

_76.t and 76.b: Fuse* - Multiverse (<https://www.fuseworks.it/en/works/multiverse/>).

_79.t: The Mill - Red Bull Music Academy 'A Night of Spiritual Jazz' Installation (<http://www.themill.com/millchannel/731/red-bull-music-academy-a-night-of-spiritual-jazz-installation#>).

_82: File - Electronic Language International Festival - Jarbas Jácome - Crepúsculo dos Ídolos (<https://file.org.br/artist/jarbas-jacome-2/>).

_84: File - Electronic Language International Festival - Jarbas Jácome - Vitalino (<https://file.org.br/artist/jarbas-jacome/>).

_85.t: File - Electronic Language International Festival - Ernesto Klar (<https://file.org.br/artist/ernesto-klar-2/>).

_85.b: Memo Akten's Official Website - Body Paint (<http://www.memo.tv/portfolio/bodypaint/>).

_86.t: Center for Visual Music's Bute Research Website (<http://www.centerforvisualmusic.org/Bute.htm>).

