Interaktivní instalace
Interactive installation

Aliaksandra Laurova

Master’s thesis
2018

Tomas Bata University in Zlín
Faculty of Multimedia Communications

Univerzita Tomáše Bati ve Zlíně
Fakulta multimediálních komunikací
Ateliér Digitální design
akademický rok: 2017/2018

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Aliaksandra Laurova**
Osobní číslo:	**K15368**
Studijní program:	**N8206 Výtvarná umění**
Studijní obor:	**Multimédia a design – Digitální design**
Forma studia:	**prezenční**
Téma práce:	**Interaktivní instalace**

Zásady pro vypracování:

1. Rešerše k tématu práce
2. Analýza pro zpracování tématu
3. Variantní návrhy řešení
4. Postup zpracování vybrané varianty řešení

a) teoretická část v rozsahu 30 – 35 normostran textu
b) prototyp nebo funkční model nebo fyzický model v měřítku 1:1, 1:2, 1:3, 1:5, 1:10 podle charakteru projektu a konzultace s vedoucím práce
c) grafická prezentace v rozsahu minimálně 3,5 m^2

Na samostatném nosiči CD-ROM odevzdejte v minimálním počtu 10 kusů obrazovou dokumentaci praktické části závěrečné práce pro využití v publikacích FMK. Formát pro bitmapové podklady: JPEG, barevný prostor RGB, rozlišení 300 dpi, 250 mm delší strana. Formáty pro vektory: AI, EPS, PDF. Loga a texty v křivkách. V samostatném textovém souboru uveďte jméno a příjmení, login do Portálu UTB, obor (ateliér), typ práce, přesný název práce v češtině i v angličtině, rok obhajoby, osobní mail, osobní web, telefon. Přiložte svou osobní fotografii v tiskovém rozlišení.

Rozsah diplomové práce:	**viz. Zásady pro vypracování**
Rozsah příloh:	**viz. Zásady pro vypracování**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

1. Pavel A. Orlov. Programming for Artists. Moscow: Avatar, 2015 – 247p. ISBN 978-5-903781-16-4;
2. Penny de Byl. Creating Procedural Artworks with Processing, a Holistic Guide. United States: CreateSpace Independent Publishing Platform, 1 edition, 2017 – 386p. ISBN-10: 153286180X. ISBN-13: 978-1532861802;
3. Joshua Noble. Programming Interactivity. Beijing |Farnham: O'Reilly Media; 2 edition, 2012 – 728p. ISBN-10: 144931144X. ISBN-13: 978-1449311445;
4. Greg Borenstein. Making Things See: 3D vision with Kinect, Processing, Arduino, and MakerBot. Beijing |Farnham: Maker Media, Inc; 1 edition, 2012 – 440p. ISBN-10: 1449307078. ISBN-13: 978-1449307073;
5. Andrew Richardson. Data-driven Graphic Design: Creative Coding for Visual Communication. London: Bloomsbury Publishing, 2016 – 196p. ISBN-10: 1472578309. ISBN-13: 978-1472578303;
6. Matt Pearson. Generative Art. Shelter Island: Manning Publications; 1 edition, 2011 – 240p. ISBN-10: 193518262. ISBN-13: 978-1935182627.

Vedoucí diplomové práce:	**MgA. Bohuslav Stránský, Ph.D.**
	Ateliér Digitální design
Datum zadání diplomové práce:	**1. prosince 2017**
Termín odevzdání diplomové práce:	**11. května 2018**

Ve Zlíně dne 1. prosince 2017

[illegible]

doc. Mgr. Irena Armutidisová
děkanka

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta multimediálních komunikací
L.S.
1

[illegible]

MgA. Bohuslav Stránský, Ph.D.
vedoucí ateliéru

ABSTRAKT

Kreativní kódování a výpočetní umění jsou relativně nová forma umělecké tvorby, která nabírá popularitu společně se stabilním vzrůstem nových technologií, které podporují tuto formu umělecké tvorby. Cílem tohoto projektu je prozkoumat možnosti programování v rámci vizuálních médií. Specifičtěji řečeno, možnosti kódování za účelem tvorby uměleckého díla. První část diplomové práce se zabývá fenoménem kreativního kódování, jeho definicí, historií, různými způsoby užití a nástroji pro jeho tvorbu. Druhá část popisuje proces a vývoj vlastní interaktivní instalace založené na předchozí analýze. Tento typ instalací se v dnešní době stává relativně populárními, tím pádem jednou z výzev je stvořit unikátní přístup k této tvorbě. Projekt je psán v programovacím jazyce Processing. Instalace se skládá z několika grafických objektů, měnících se a dá se říct rostoucích obrázků a animací, vygenerovaných kódem, které jsou promítány na zeď. Obrázky jsou v interakci s divákem, mění své tvary nebo rychlost pohybu, na základě změn v pohybu provedených divákem. Princip této interaktivity je ztvárněn pomocí technologií s detektory pohybu. Finální instalace se bude nacházet na tmavém místě, kde každý návštěvník bude moci navázat kontakt s obrázky, měnit jejich vzhled a možné pořadí projekce. Celkový dojem bude podpořen zvoleným audio materiálem.

Klíčová slova: kreativní kódování, algoritmické umění, výpočetní umění, programování, interaktivní instalace, Processing, detekce pohybu.

ABSTRACT

Creative coding, algorithmic and computational art is a relatively new form of artistic creation that is gaining traction with a steady flow of new technologies to support it. This project shifts attention to programming as an artistic tool and a modern form of expression. The first part of the thesis is dedicated to the phenomenon of creative coding, its definition, history, various fields of usage and tools of its creation. The second part describes the process of development an original interactive installation based on the previous analysis. The final application is written in the Processing programming language; it displays a series of changing images that are being projected on a wall. Interactivity is supported by a motion detecting system, that reacts to visitor's presence in a room, and due to that movement brings some changes to the images.

Keywords: creative coding, algorithmic art, computational art, programming, interactive installation, Processing, motion detection.

ACKNOWLEDGEMENTS

My gratitude to the diploma project supervisor MgA. Bohuslav Stránský, Ph.D, for helping to develop the idea, supporting the project and for letting me work within my own tempo. I want to thank all the friends and family for being there for me.

“The biggest emotion in creation is the bridge to optimism” – Brian May

CONTENTS

INTRODUCTION

It's a modern society's tradition to split the flow of ideas into two opposed streams: the mechanical and the organic. It is believed that mathematics and programming belong only to the cold world of logic and precision, while the field of art is ruled by emotions and sentiments. The goal of this work is two combine these concepts, to use code and algorithms as another artistic tool, as much as a pencil or watercolors, in order to express yourself. Being relatively new in this area, I am fascinated by the way a few strings of code can be converted into a living, moving image, how with a help of cameras and sensors this image can interact with any physical object in a room, how it can change and develop. This project is my attempt to show the beauty of code and a wide range of possibilities for the artists that it brings, to pay attention to a visual programming, a rapidly evolving tool in artistic performance, electronic music production, VJ culture and interactive installations.

This project is a big personal challenge; working with the material I've never faced before, learning new techniques, developing ideas is something an education is all about for me.

I. THEORY

1 THE BASIS OF CREATIVE CODING

1.1 Defining terms

"You might not think that programmers are artists, but programming is such an extremely creative process. It's logic-based creativity." - *John Romero*

Once in a while through the history of human creativity, a new form of art is being born. With a development of urban planning and agriculture, architecture and a landscape design were mastered. The history of traditional visual arts goes back to the ancient caves, such as Chauvet in France, which's paintings date back to approximately 30,000 years before present. Presumably, first musical instruments were crafted around that time, as well as the dances that went with music. Myths and rituals were eventually followed by verses and theater acts. Videos, movies and photography joined the family of art relatively recently.

Computer art is the next newcomer, which is indeed so new, that no criteria have been designated on whether it is worth to be taken seriously, if it can be put in the same row as traditional forms of art. [1] When people think of programming they tend to associate it with applications, databases, software development, but not with the expressive media. While talking about a computer art an average person most likely will mention such things as Photoshop-created images, computer animations, 3D models. Despite that, dozens of new forms of performances, installations, visualizations are being produced with a help of code. Constant development of digital technologies brings a wide variety of new techniques and procedures. Visual forms are being created through a computational logic that is run by a specific set of rules depending on a choice of a code language and a programming platform. A list of new terminology is gaining its popularity, such as:

- creative coding;
- code art;
- generative art;
- computational art;
- interactive art;
- algorithmic art;
- procedural art, etc.

These terms themselves might appear somewhat foreign to the world-art mainstream, and before getting into further analysis, it will be useful to explain them. Although, trying to give each one a precise definition reminds of the parable of an elephant and blind men: a group of blind men is inspecting an elephant for the first time. One of them, with the hand on the trunk, said: "This being is like a thick snake". The second, whose hand was on animal's leg, proclaimed it to look like a tree-trunk. For another one, whose hand reached elephant's ear, it appeared to be some kind of fan. The man who felt its side said: "elephant is a wall". The blind man feeling its tail compared it with a rope. The last one exploring elephant's tusk described it as a smooth and hard spear. [2]

Indeed, while being used for many different fields of activity, definitions might wary like blind men's descriptions. Here are some of them:

Mitchell and Bown define *creative coding* as a process, based on a phenomenon of discovery, that includes research methods, repetition and reflection, and that uses code as the main source for creating works of art. [3]

"*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." [4]

"*Generative software art*, as it is usually understood today, is artwork which uses mathematical algorithms to automatically or semi-automatically generate expressions in more conventional artistic forms. For example, a generative program might produce poems, or images, or melodies, or animated visuals. Usually, the objective of such a program is to create different results each time it is executed. And generally, it is hoped that these results have aesthetic merit in their own right, and that they are distinguishable from each other, in interesting ways. Some generative art operates completely autonomously, while some generative artworks also incorporate inputs from a user, or from the environment." [5]

"*Interactive art* is any type of art that involves the viewer in the creative process. Interactive art attempts to challenge the traditional boundary between artist and "audience". It may use a physical medium, as in the case of installation art, or it may be purely digital and Internet-based. Interactive art often uses computing power to govern responses to viewer actions." [6]

Taking into account all the definitions, it is safe to say that creative coding is:

- a collaboration between an artist and a computer;

- a procedural way of creating an expressive media;
- a meeting point between an organic and mechanic;
- a new artistic approach to programming that's gaining its popularity.

Basically, creative coding is a type of procedure that uses programming for expressive purpose rather than functional. Creative coding enables to mix code and art, to bring a human being into a program, to convert live motion into strings of zeros and ones.

Creative coding is neither art nor programming, in their conventional sense. The truth is, it's both and neither of these things. Programming is a bridge between a human being and the machine, it has its strict logic, a set of algorithms to use and a great number of rules to follow, while the process of art creation is mostly based on emotions and feelings where the result can be interpreted in many ways based on the observer's sense of perception. Projects that are born with the creative coding is a meeting point between the two. [7] It's a discipline of converting cold structured processes and turning them into unpredictable and expressional results. Nowadays, more and more artists turn their minds to this new world of creativity, developing their own methods, styles and techniques.

1.2 Creative coding is easy

Oftentimes programming is viewed as a complicated process that not everyone can conquer. Learning programming indeed requires a certain amount of efforts, but so does mastering any other tool or discipline. Code is just another instrument of creation and, as well as an academic drawing, it has a set of standards. These standards and rules might be a little more specific, but not much more complicated.

Learning a programming language shouldn't be viewed as an essential barrier to enter the world of creative coding. Compared to the fields of artistic creation, it has an indisputable advantage: unlike the usual practice of drawing/sculpturing etc. a minimum level of skills doesn't have to be sharpened, most of the procedures and algorithms are built inside the software tools. While it usually takes years to learn to paint, compose music and in general to create something meaningful, a set of programming techniques required to get significant results in creative coding can be learned within just a few days. [7]

It's a common thought that programming is only for people who are good at mathematics and all the technical disciplines. It might be true when it comes to those who create the programming languages themselves, this, indeed, requires a technically oriented mind. But as soon as a language and environment are created it is possible to involve more artistic people into the further development.

Alternative languages, *Processing*, for example, bring a lot of new possibilities for a visual purpose. The other example of an alternative language is *Logo*, which was designed in the 1960s. It's creator, Seymour Papert, planned it as a concept for children; with its help they could program different kind of media, graphic images and animations or even robotic elements.

Another more modern example is the *Max* programming environment. It was created by Miller Puckette in the 1980s and it has significant differences from the other languages. The program code is represented by a set of boxes, not lines of script, and in order to make a program do a certain command a user would have to move and organize these boxes. A lot of audio and visual software was created using this method. Just like that, modern graphic user interfaces open up a door to the world of code for millions of people, alternative programming languages continue to help artists work directly with software, not seeking for a help of technicians. [8]

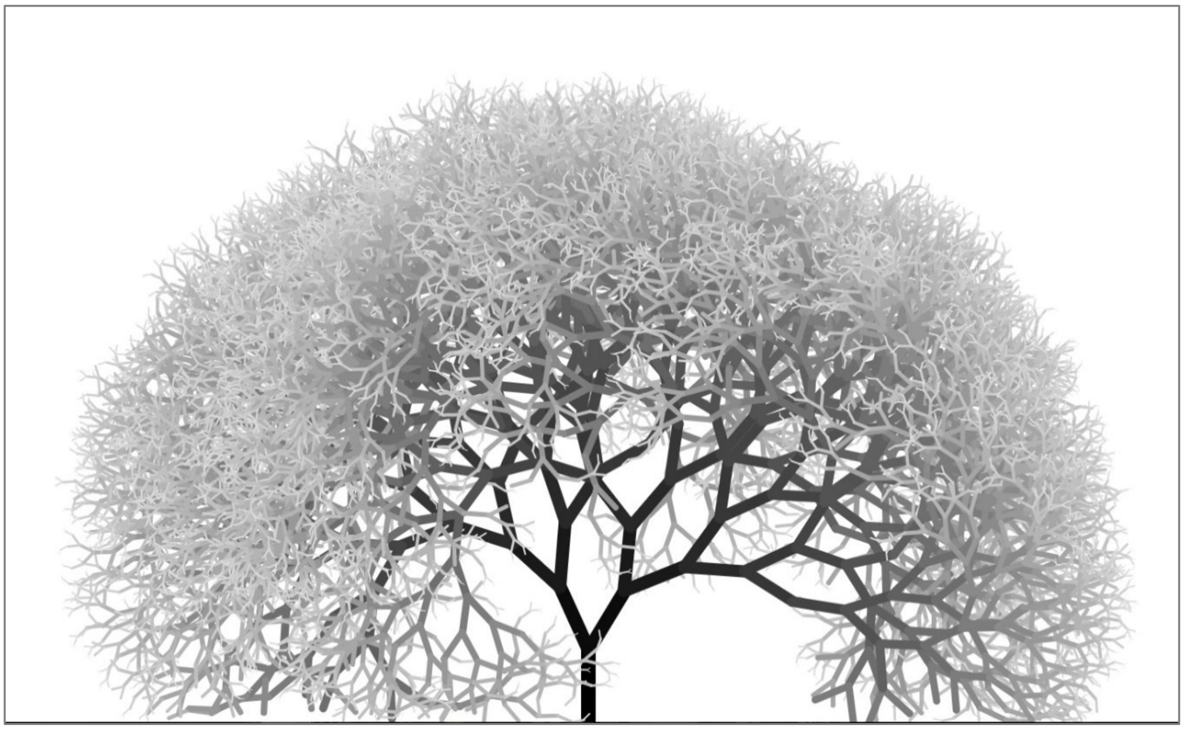


Figure 1. EnderPicture, "Tree fractal, natural", 2018: https://www.openprocessing.org/sketch/504372

The image in the Figure 1 was generated using only 16 lines of code (with the Processing programming language), and every time the page (or program) is being refreshed, it generates a different "tree", never repeating itself. All the code does is creates a vector and splits it into few more after a certain period of time, choosing random directions and decreasing its size and color saturation.

The practice of creative coding has a number of features similar to the characteristics of any other creative process: oftentimes an artist begins to create, having only a rough idea in mind, not a detailed task. Experiments are a part of any working process; combining different commands and procedures might lead to an unexpected result for a coder himself. Thus, one of the key features of creative programming is the absence of a formalized technical assignment.

To start working with code art no serious knowledge of programming is needed, it could even be the other way round – more experienced programmers that are used to the typical computational logic might need to let some of it go and re-educate themselves. Code art is a more chaotic practice, it is something people normally don't expect or welcome from computers and technical devices. "Order and chaos may be mutually exclusive, but that doesn't mean they can't work together." [7]

1.3 Chaos and order, art and programming

"Chaos in the midst of chaos isn't funny, but chaos in the midst of order is." - Steve Martin

Creative coding does not equal to usual programming practice. Modern computer programmers tend to turn their back on the world of nature and expression. Their applications and algorithms are meant to serve a specific purpose, be useful and productive. The chaos of the natural world isn't welcome in this kingdom of logic; you'd expect a computer to execute the exact command it was given. However, the same does not apply for a creative coder, whose work is meant to be based on logical procedures but resulted in a creative way that can have various interpretations.

Chaos and order, nature and technologies, imagination and logic aren't necessarily in the opposite corners of a ring, they work together, they are codependent. One couldn't exist without the other, our own existence is not possible within the rates of just one of these dimensions. Life is only born in harmony of order and entropy, in a meeting point between the twirling chaotic environment of nature and cold, structured, simple field of order. Any living being carries both of these traits, and there is a never-ending fluctuation between them. [7]

While being unpredictive, expression and emotions are not something wanted in traditional practice of strict programming, which is why more experienced coders may not take creative aspects seriously and find such ideas somewhat unpalatable. Fortunately, no one is asking to take sides. The goal of creative coding, if any goal need to be defined at all, is to create something beautiful, to express an idea. The whole process starts with taking the mechanic – all the logical sets of command and algorithms - and turning it into something organic. Code is another tool of creation; many live paintings, interactive installations and art objects are born this way.

One of the modern examples of such a collaboration is "Spaghetti Coder" - a multimedia project started in January 2015 by Toni Mitjanit - a creative coder in the area of generative art, in order to explore new boundaries in audiovisual expression. Toni uses creative coding practice to produce interactive animations, moving unique graphics and audio projects.

From the interview with Toni Mitjanit: "I've always been amazed by the beauty of nature and its wonderful patterns: symmetries, spirals, meanders, waves, cracks or stripes. At the very beginning I started creating artworks with basic geometry and fractals, but later I discovered the possibilities of using randomness, physics, autonomous systems, data or interaction to get more expressive and meaningful artwork. My goal as a generative artist and experimental animator is to create autonomous systems that make the essence of nature's beauty emerge by modeling not only its appearance but its behavior. Mathematics, physics and computation are essential tools to implement a set of rules that an autonomous system must follow to simulate nature's appearance and behavior." [9]

Depending on the nature of the artwork that is being developed at the moment, different kind of tools and procedures are used: software programs, mathematical algorithms, including trigonometry, differential equations, vector spaces and calculus, matrices, graph theory, etc. Visual products (images, animations, videos) are being generated with such programming languages as Java, C, C++, Processing, openFrameworks and GLSL. To compose the soundtracks and audio effects Toni uses SuperCollider and Chuck programs.

Images in the Figures 2 and 3 are examples of the arts made by Toni. The first one, "FIREFLY TRAIL (III)", is generated with a help of random walkers, noise and physics in order to paint a lighten firefly trail. The second one, “PRIME MOVER (II)”, is a drawing with agents that move using an angular movement. Their birth position is based on concentric circles and their opacity is reduced exponentially.

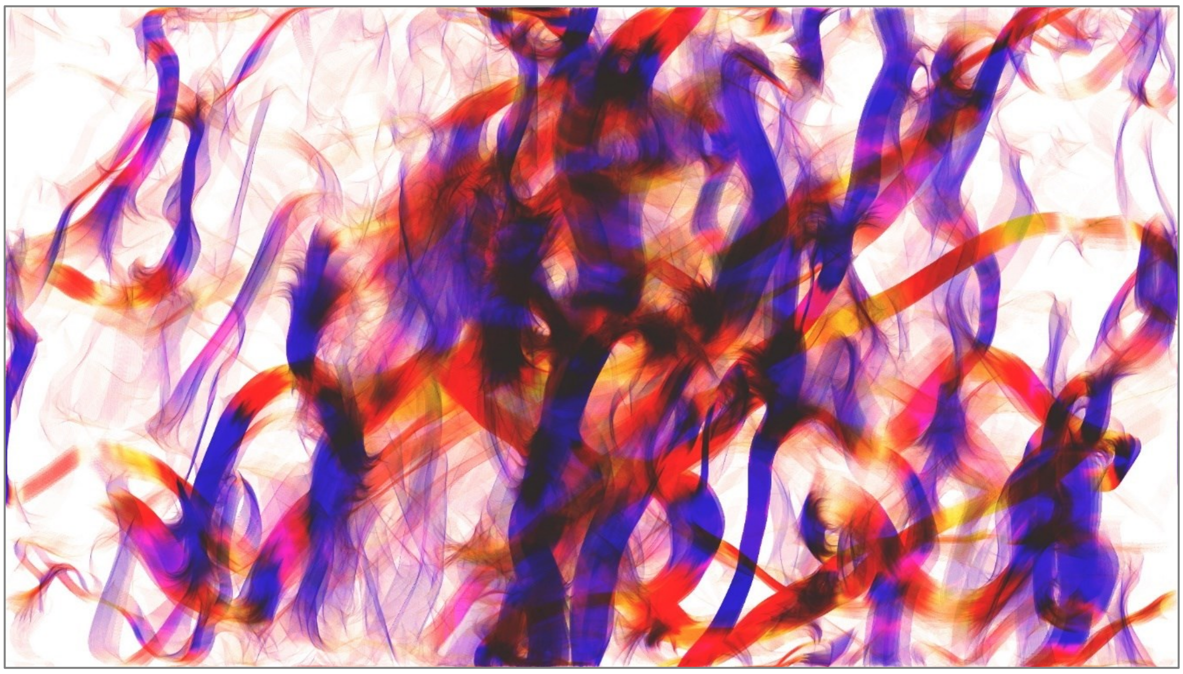


Figure 2. The “Spaghetti coder” project, “FIREFLY TRAIL (III)”, 2016: https://coderspaghetti.wordpress.com/2016/07/27/firefly-trail-iii/

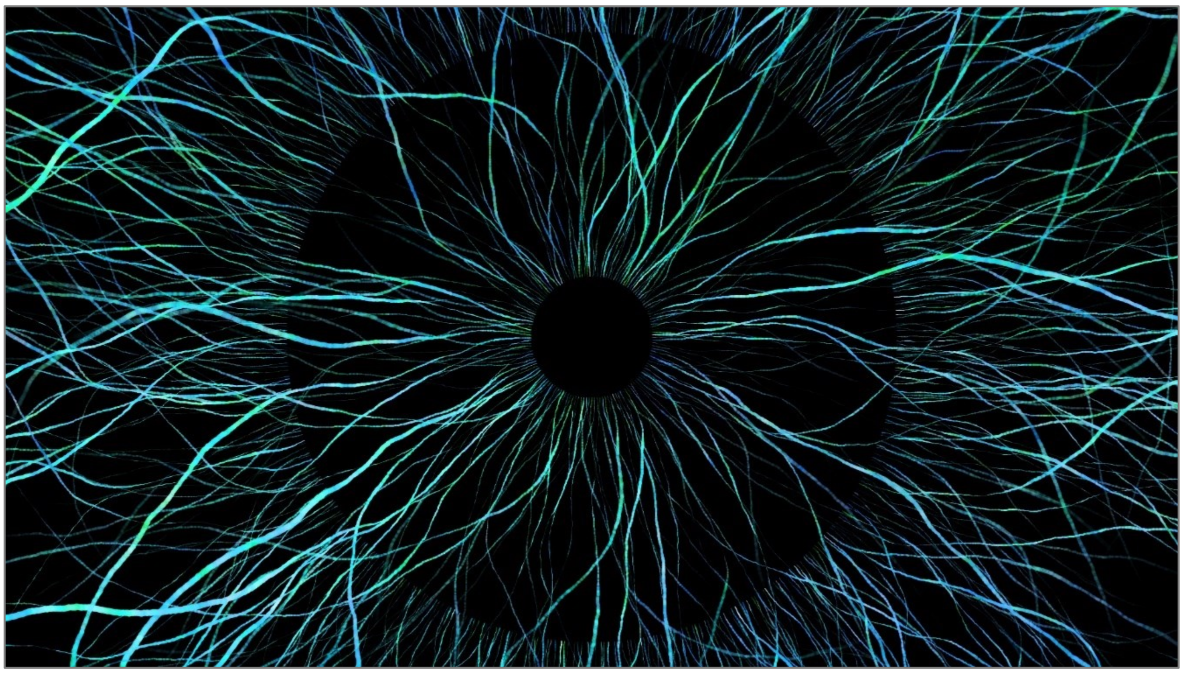


Figure 3. The “Spaghetti coder” project, “PRIME MOVER (II)”, 2016: https://coderspaghetti.wordpress.com/2016/02/03/prime-mover-ii/

2 CREATIVE CODING IN PRACTICE

2.1 The pioneers of code art

A creative process of any kind does not exist in a vacuum - it reflects the world around, the aesthetic aspects of the day, the technological progress and the historical background. Today we witness the heyday of the technological era – digital space provides us with new ways to work, communicate and to create.

Everything has its start, so does the programming art. Soon after its invention computer has been involved into the creativity process. In the late 50s, few years after the first commercial computer (The UNIVAC) was developed, artists began their experiments using it as a new tool of creation. Of course, being extremely expensive and exclusive, computers were inaccessible to the majority of the population. Jasia Reichardt, a British art critic, curator, and writer, recalls that only a few places could provide artists with the access to the university computers: "one can assume that there are probably no more than about 1,000 people in the world working with computer graphics for purposes other than the practical ones." [10]

Up from these early computing days the story of live programming goes on, new forms of art are being invented: from interactive installations to live art performances. No matter what the output is, there is always a person behind the high-tech curtain. Numerous artists and coders left their trace through the history, the following names are just few examples of the many.

2.1.1 Harold Cohen, 1928 - 2016

Harold Cohen, a British-born artist, is considered to be a true pioneer of computer and algorithmic art. He was also an engineer, whose approach helped the progress of the first generation of computer-produced art. Cohen is most famous for developing the *AARON* system - one of the longest-running, continually maintained artificial intelligence systems in history.

Cohen was a painter that grew weary with the traditional ways of art and switched his attention to the potential of programming-based experience. By the beginning of the 70s he spent two years at the Artificial Intelligence Laboratory of Stanford University as a guest expert. That's when his work became focused mainly on the integration of the artificial intelligence with the artist's practice. Cohen's initial question was: "What are the minimum conditions under which a set of marks functions as an image?". To find an answer he developed an expert drawing system, a software based on a set of algorithms and patterns of the C programming language, which's main goal is to simulate certain traits of human creativity and art. The system was named *AARON* and has been in continual development since 1973. [11]

AARON is capable of creating original artworks on its own and its style has been changing through the years. The first set of works represented simple lines and shapes, then adding more complex elements, the perspective of a foreground and background; eventually AARON learned to recognize when a picture has reached its final stage, creating rather naturalistic and organic representational images. Through all his life Mr. Cohen have been working on improving AARON's style. Starting from the abstraction in the 70s, then changing the direction toward the field of realism, adding more details, generating rocks, then plants, then people. In the 1990s AARON learned to apply colors to the images. Two examples are represented in the Figure 4. In the 2000s its style returned back to abstraction. AARON developed from using special plotter devices to draw an image on canvas to a set of completely digital paintings.

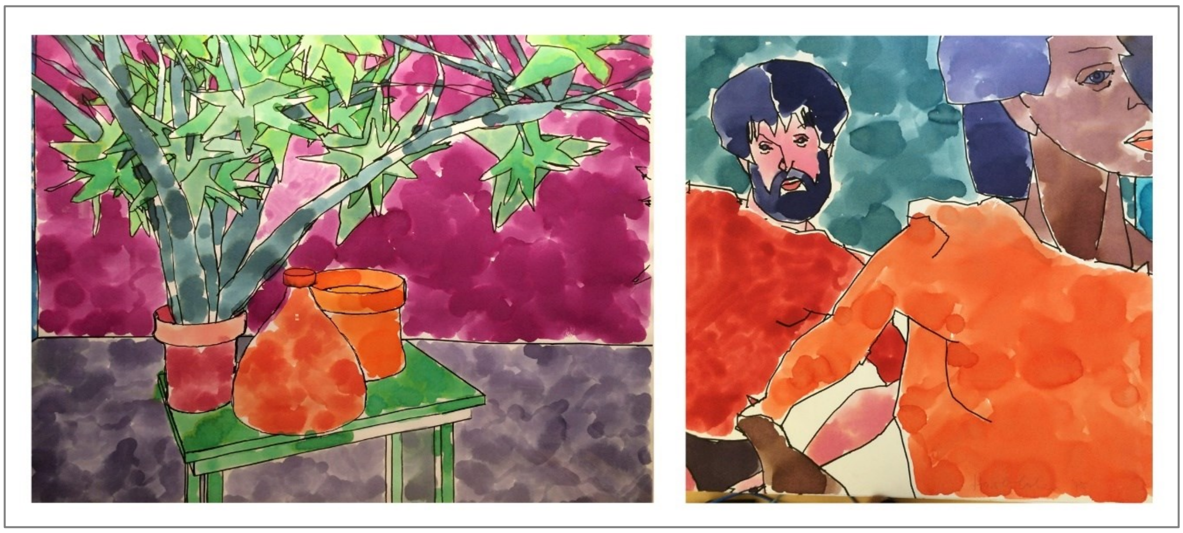


Figure 4. Images generated by AARON at the Computer Museum, Boston, MA, 1995: http://www.computerhistory.org/atchm/harold-cohen-and-aaron-a-40-year-collaboration/

Harold Cohen’s career lasted for more than 60 years, his works are held in major museums all around the world, such as the Victoria & Albert Museum in London, the Stedelijk Museum in Amsterdam, the Tate Gallery. Frieder Nake, a legendary computer artist, commenting on Cohen’s death, said: “Harold was the lonesome rock of rule-based algorithmic art. Nobody in the world did anything in fine art as courageous, as daring, and as successful as he. His approach was unique in all its facets. His rich work stands out unparalleled.” [12]

2.1.2 Roman Verostko, b. 1929

Roman Verostko is an American artist, who uses code to generate images, known as algorithmic art. He created his own software that produces drawings based on a transformation of forms. Verostko’s “visual poetry” is a result of numerous calligraphic scripts that run with the coded procedures. One example of his work is represented in the Figure 5.

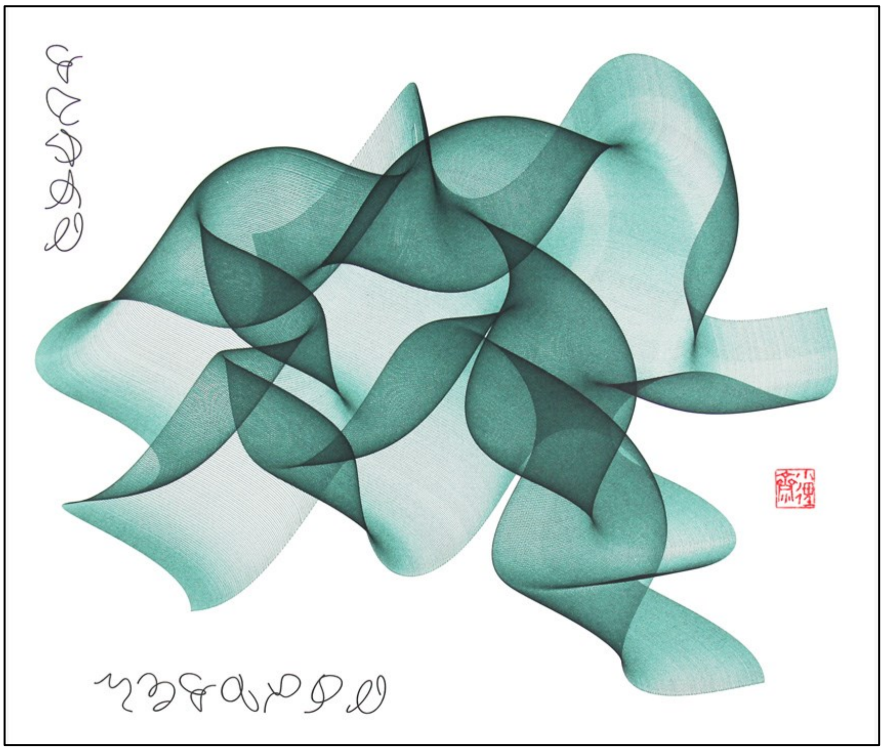


Figure 5. R. Verostko, “Green Cloud (the drawing)", 2011: http://www.algorists.org/algorists/roman/green-cloud-w2.jpg

Verostko has a long, multifaceted career, starting as a painter in his early life, through the studies as a Benedictine monk, to the computer art practice. As a painter, Verostko was inspired by such masters as Malevich, Kandinsky and Mondrian in his struggle to discover the balance of opposing forces of nature within a single drawing.

During his career, Verostko created many outstanding projects, such as "*Magic Hand of Chance*" in 1982 - an interactive computer program that produces endless series of light effects and videos that do not repeat themselves. This program was written on a first-generation IBM computer in the *BASIC* programming language.

Verostko also created "*Hodos*" - a software/hardware system. The software generates digital images which are being drawn on paper using brushes mounted on a pen plotter's drawing arm. This system is able to create different kind of images, based on a personal style of an artist and his interest in some specific theme.

Verostko is also one of the founders of "the Algorists" group. Alongside with Jean-Pierre Hébert and Ken Musgrave they created an informal "club" for those artists, who use algorithms and math logic for the purpose of self-expression and art creation.

His works can be found in numerous museums and educational centers, such as: Minneapolis Institute of Arts, Albert Museum, Saint Vincent College, Spalding University in Louisville and Frey Science and Engineering Center at the University of St. Thomas in St. Paul. His images have been featured in many exhibitions around the world, including Berlin, New York, Rome, Istanbul, Tokyo and others. His work was recognized by prestigious awards, like Prix Ars Electronica (honorary mention, 1993), the Golden Plotter (first prize, 1994) and Gladbeck (Germany). [10]

Verostko, describing his art, said: "My art reflects coded procedures driving the technologies that shape our culture. Each drawing visualizes the code by which it was generated. With surprising grace and beauty, these visualizations invite us to ponder the power of form-generating code with its seemingly stark logic. By doing so, this art celebrates the mysterious nature of coded procedures that underlie the shape of our evolving selves." [13]

2.1.3 John Maeda, b. 1966

John Maeda is an American designer and technologist, one of the most well-known digital art pioneers in the world. Maeda's studies, techniques and innovations have made a major impact on a sphere of digital design, and especially his work influenced the new generation of code artists.

Being an artist himself, in his early art projects Maeda combines traditional art techniques with creative coding, using digital media as a tool of expression and creation. It resulted in a set of live interactive images and videos; created with digital instruments they were later printed out on canvases and placed in different exhibitions. One example of such a work is a set of pictures called “Space diagrams” (Figure 6), where Maeda applies two-dimensional shapes to a plane, changing their proportions and a distance from a center point.

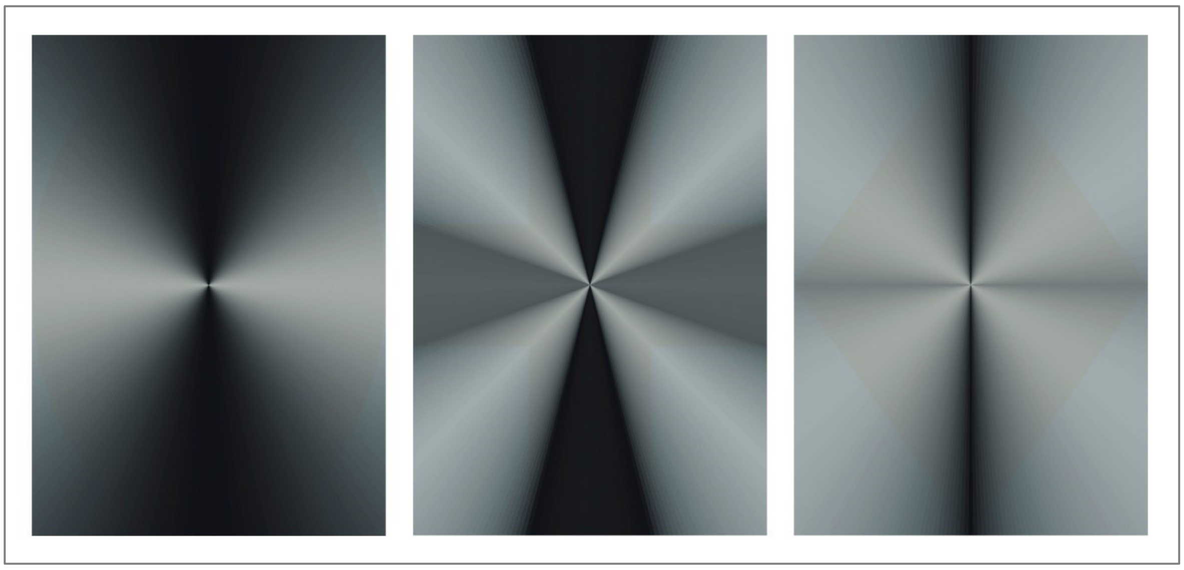


Figure 6. J. Maeda. Space diagrams, 1995: https://maedastudio.com/space-diagrams-1995/

During the period of 1996-2003 Maeda was working as a director of “Aesthetics + Computation Group” (ACG) at Massachusetts Institute of Technology (MIT), that concentrated on developing new methods of creativity using computer applications and creative code methods. Later Maeda changed the concept of the research, transforming ACG into Physical Language Workshop (PLW), a new group of researches, whose main goal is to design software/hardware tools for creating digital media in a network environment. At the same time Maeda codirects another media lab called “SIMPLICITY”, which aim is to redefine daily user’s experience with modern technologies.

Maeda wrote a lot of books and guides for everyone who is interested in digital design. He was included in Esquire’s 1999 list of the 21 most important people of the 21st century due to his vast range of works, extraordinary talent and critical approach to the implementation of technologies and the development of digital sphere in user's every day's life. [10]

2.1.4 Casey Reas, b. 1970

Casey Reas is an American artist and designer. His innovative approach to the use of software results in many artworks that show the beauty of code and algorithmic procedures. Moreover, alongside with Ben Fry, Reas is famous for creating a popular open source programming language and environment for the visual arts called "*Processing*", that was launched in 2001

Reas is working as a professor and lecturer at the University of California, Los Angeles. He got his bachelor's degree in Media art and Science from the College of Design, Architecture, Art, and Planning at the University of Cincinnati, and then the master's degree from the MIT.

As an introduction to their book about Processing, written with Ben Fry, Reas says, that while being a graduate student at the MIT Media Lab, he met a specific group of individuals interested in combining their knowledges from different fields of study. In general, these knowledges were related to computer technologies and programming and people had some experience in a number of disciplines, such as architecture, mathematics, physics, art, design and music. Only few software environments, that combined both: a sophisticated programming language and an opportunity for creating visual media, were available during that time. To meet all the needs and use all the potential provided by the digital technologies, these people started to work on their own software programs. The development of these tools and their experience with numerous projects led to an uprising of a new culture that combined the visual media artworks with computer science possibilities. The years at the Media Lab and collaborations with other technologists, the need of a new approach to this information for a non-technical people became a basis and motivation for the development of *Processing*. [8]

Reas uses code as his primary medium. However, the final result might vary from interactive installations and online web-projects (Figure 7) to printed pieces. His works have been featured in many exhibitions at art galleries, museums and educational institutions all around the Europe, United States and Asia. Some of the recent expositions were placed at the Art Institute of Chicago and the San Francisco Museum of Modern Art. Rease's projects have been awarded by the New World Symphony in Miami and the Whitney Museum of American Art. His prints are placed in a number of private and public collections, such as the Victoria and Albert Museum and the Centre Georges Pompidou. [14]

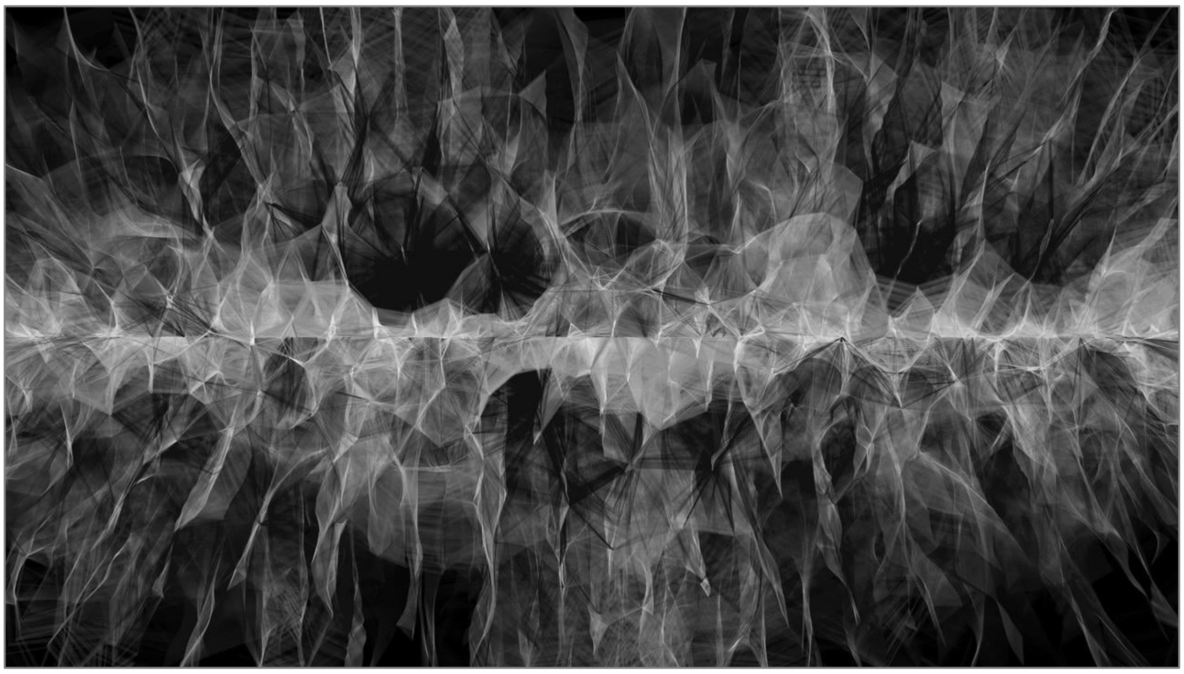


Figure 7. Casey Reas, Process 13 (Software 2), 2011: https://vimeo.com/22063474

2.1.5 Jared Tarbell, b. 1973

Jared Tarbell is another code artist from the USA, New Mexico. His love of computational media began with the introduction to his first personal computer in 1987. Ever since then Jared has been developing his skills of programming and creating a unique approach to its application.

Tarbell got his bachelor's degree in Science from the New Mexico State University. His major interest is concentrated around the visualization of mathematical procedures and the transformation of the complex code algorithms into works of art. [10]

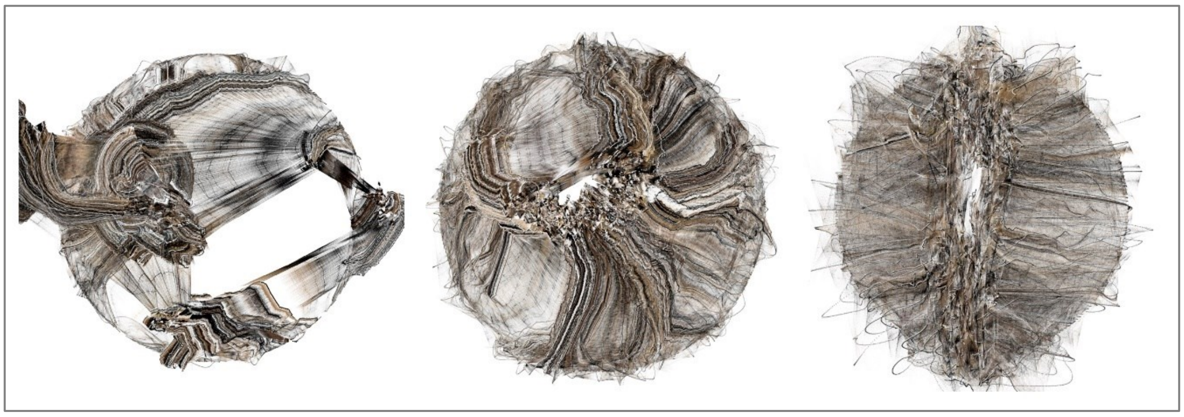


Figure 8. Jared Tarbell, Happy place, 2004:
http://www.complexification.net/gallery/machines/happyPlace/

Tarbell's work is a result of his studies of complex mathematical structures and its graphic representation. He always bases his projects on integrations between analytic and aesthetic concepts. The result reflex his deep interest in a field of visualization and generative art. Many of his early works are created using ActionScript - an object-oriented programming language. With the development of *Processing* Tarbell switched to that environment. Constant development of its capabilities allows Tarbell to produce even more complex, organic and expressive arts. One of these projects is shown in the Figure 8.

Tarbell inspires and supports a lot of coders with his commitment to an open source development; many of his projects can be found online with a full description and an access to the source code. He makes his contribution to the world of creative coding by cooperating with other authors on different programming guidebooks (like of ED book, New Masters of Flash, Volume 3), taking part as a lecturer on numerous international conferences and by writing online articles describing his field of study. [10]

2.1.6 Ben Fry, b. 1975

Benjamin Fry is an American expert in data visualization. With his co-worker, Jared Tarbell, Fry represents a modern generation of coders and artists that seeks for inspiration in the computational processes and generative structures. He's also a co-creator of the Processing language and software.

His passion to computers and programming began with the concepts of household electronics that he used to study. As soon as he gained all the available knowledge in that field, Fry moved to software. Computer technologies brought a number of questions and challenges, Fry started to teach himself programming, mostly by studying someone else's code and modifying it according to his needs. With time it became easier to write his own material from a scratch. Eventually, this resulted in Fry's fluency and comfort in coding, that is evident in his works, which can be viewed as a poetic and advanced representation of computational process. Many of his modern projects deal with visualization of massive data sets and dynamic sources of information. [8]

Fry has been working on creating tools for visualizing the genetic data and human genome for the Eli & Edythe Broad Institute of MIT & Harvard. His personal work also deals with this study. For example, his well-known project "Valence" (Figure 9), that is based on a set of software sketches, which explore the structures and connections inside large sets of data. "Valence" has two versions: the "light" one – built using the Processing language, and more complex based on C++, Perl, and OpenGL.

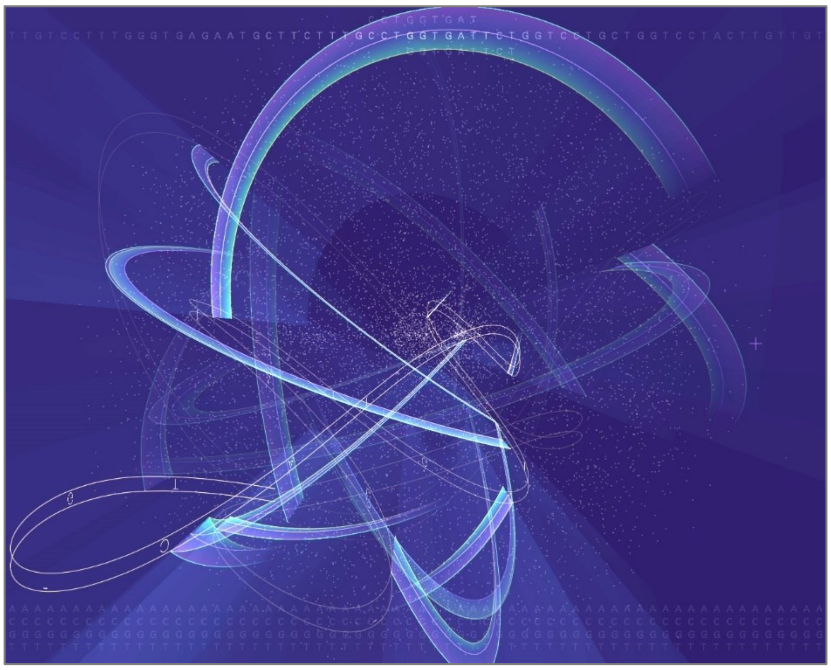


Figure 9. Ben Fry, Valence, 2002: http://benfry.com/genomevalence/

Fry's and Reas's project – Processing, received a number of awards. In 2006 it got a New Media Fellowship from the Rockefeller Foundation in order to keep its development. Processing was featured in the 2006 Cooper-Hewitt Design Triennial. A number of Processing guidebooks were published.

Fry’s works were represented at different art festivals, conferences, exhibitions and museums, such as Whitney Biennial, Cooper Hewitt Design Triennial, Ars Electronica in Linz and Museum of Modern Art in New York.

2.2 Modern generative art

Despite having a history, digital technologies are still very young, the computer is only in its first century. All the programming tools give coders and artists the ability to change the way our media looks and behaves. The newest software gives a power of creation to anyone who is willing to learn. Thus, thousands of people try to test their skills in the world of creative coding. Numerous books, guides and articles are available. People share their experience on web portals, online galleries and forums. Special festivals, such as “EueO” and “Resonance” are being organized to gather creative coders from all around the world, demonstrate their works and discuss new ideas.

The variety of techniques, styles and visual results is truly tremendous. Starting from the minimalistic approach (Figure 10) to more complicated one (Figure 11). The first one belongs to Anders Hoff – a generative artist, the creator of the “Inconvergent” project. This project studies ways of turning code lines into aesthetic compositions and structures, inspired by the combination of nature and mathematics. The second picture is a part of a “Spaghetti coder” project, that was mentioned earlier. It represents a chaotic twirling figure that reacts to the audio material and is being distorted based on an FFT audio spectrum.

Nowadays, the use of creative codding can be applied to any subsection of an art practice. This includes music, installations, stage performances, interactive art objects, film industry, advertising, entertaining, educating. Every day brings new technologies that only enrich the potential of the further development of the code art.

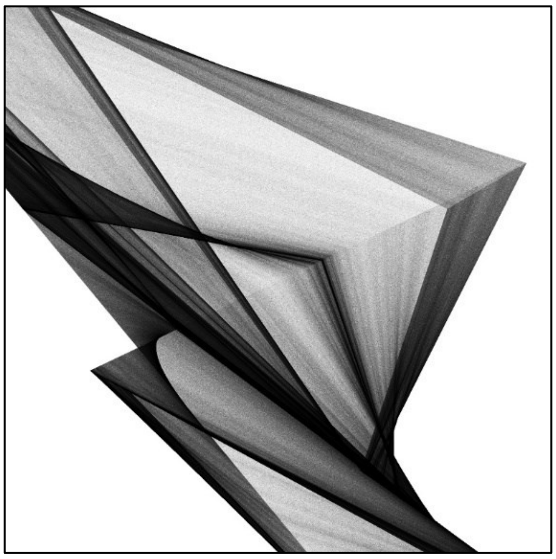


Figure 10. Anders Hoff, “Experiment 4ba9bcf”, 2017: http://inconvergent.net/2017/summary/

Figure 11. “Spaghetti coder” project, “DISORDERED SPACE (III)”, 2016: https://coderspaghetti.wordpress.com/

2.3 Algorithmic Composition

The history of computer usage as systems for composing music goes back at least to the paper by Brooks, Hopkins, Neumann, and Wright in 1957. In that paper, they describe the creation of a statistical system of Markoff chains for analyzing a body of musical scores. With a set of data and statistics, the system could create new scores that mimic the style of the analyzed ones. [15] Later numerous audio synthesizers that could recreate the sound of traditional instruments gained their popularity. Many applications, programming languages and environments were developed, such as ChucK, Cmix, Csound, FAUST, HMSL, Kyma, Laptop orchestra, etc. All of this led to a new term and trend – an algorithmic composition.

Algorithmic composition (or automated composition) can be defined as a process of putting together different algorithms to create a piece of music. This process is mainly automatic, which means the minimization of a human intervention. A composer might develop his own set of algorithms or use the ones provided by a chosen software. The increased role of a computer in all spheres of creativity made this method extremely popular. As a result, a new generation of music was developed.

2.4 Live programming

Live coding (or live programming) is a form of performance and art creativity based on a use of source code during a live performance. With a help of programming a performer might add different visual and audio effects to a stage, including digital images/animations on a screen, light systems. The key thing is that all the effects are changing in a real time, oftentimes by improvisation. This type of performance is very popular in a computer music industry, when a coder with a laptop replaces a DJ and his set.

Thinking about live coding, people might assume that the "live" part refers to a live audience during a performance, while it actually describes all the changes a coder, composer or performer applies to the program. During a live coding performance, the source code is usually displayed on a screen behind the act itself. It might be shown as it is – in strings of commands and algorithms (Figure 12) or it can be transformed into a visual material (images/animations) with a help of special programs.

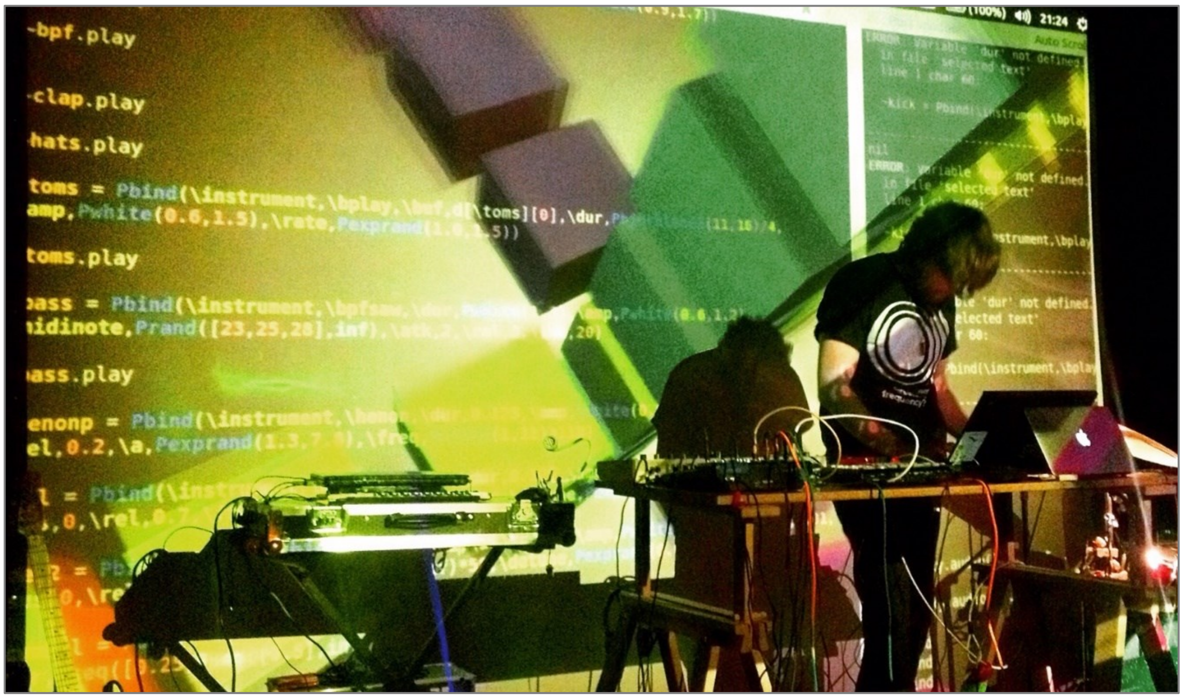


Figure 12. Sean Cotterill, Live Coding, Algorave & 3D Visuals, 2015: http://dm.ncl.ac.uk/seancotterill/2015/11/30/live-coding-algorave-3d-visuals/

The idea of live coding is not new, it goes back to the early days of computing. Programming languages, such as SmallTalk, allowed to immediately reflect any modifications made to a source code, the result would appear in another window. Later this practice was applied to different forms of art: from interactive interpreters to live art performances. Live coding practice influenced a lot of creative situations, such as movie animations, game development, interactive design, music composing.

Computer technologies made a giant leap over the last decade; many visual artists, performers and musicians started applying their engineering skills to create their own live programming systems, environments and languages, such as:

- ChucK;
- COLT;
- Impromptu;
- Fluxus;
- Extempore;
- LiveCode;
- Scratch;
- SuperCollider;
- Sonic Pi;
- TouchDesigner.

One of the challenges that live programming faces is a problem of source code visualization. Over the past two decades a number of possible solutions were introduced. Some programs, such as *CodeCity*, that renders a program and displays the result as a developing metropolis, or *SeeSoft*, that represent all the source code lines using digital graphics, do a fine job of visualizing, but might lack the actual liveness. The problem of visualization still stands, it does not support the main live coding goal yet, but there are many concepts and ideas that can allow it in the nearest future. [16]

2.5 VJing

Another form of creative coding is VJing – a process of mixing different images, videos, animations, graphic effects in a real time during a performance. This practice is mostly used on parties, night clubs, as a part of concert tours, on any kind of live events that include audience, even in churches. Typically, VJing is just a part of the act that supports the main event. It is a relatively new form of performing, that shows some similarity with other kinds of media arts, like videos, films, computer games, etc., but still has its significant differences.

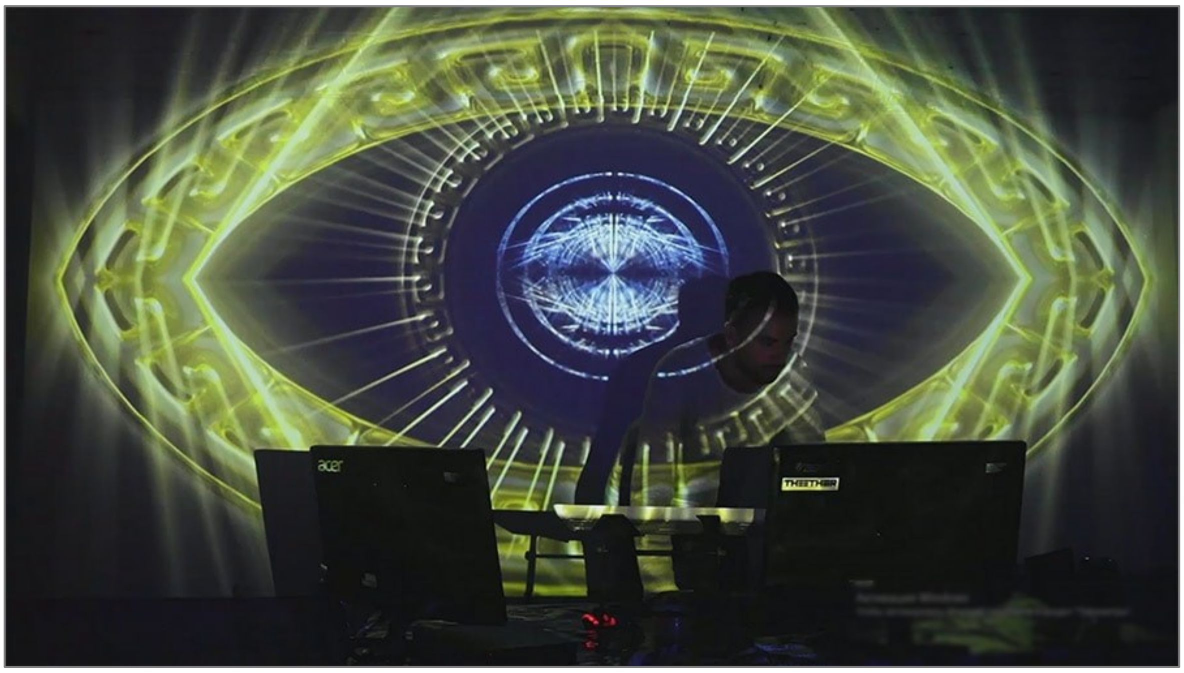


Figure 13. Live VJing by Limeartgroup: https://limeartgroup.com/services/vjing-videomixing/

VJing is always a collaboration of music and visual effects; its main goal is to turn an event into more colorful and entertaining performance act (Figure 13). Modern Video Jockeys (VJ) use a big set of hardware/software equipment, that usually includes laptops, input and output devices, many kinds of MIDI controllers, audio mixers, scratch pads, etc. Normally, such sets offer a huge range of possibilities and, as a result, software becomes too complicated, which can lead to a VJ being fully concentrated on the mixing, playing and creating process rather than on the live audience. Such a performer usually stands behind a table with all his equipment, that, unfortunately, hides most of the onscreen interactions. For audience members not familiar with a creative coding practice it might be unclear that the visual effects are being mixed and created right on a stage. Thus, such a barrier between a VJ and an audience is still to be overcome. [17]

Computer and programming are always main tools in creating materials for VJing performances. Artists use different kinds of programming environments, such as Macromedia Director, Max/MSP, Quartz Composer etc.

2.6 Projection mapping

The practice of projection mapping is relatively new, but it has already become one of the most popular forms of displaying visual contents. Projection mapping uses usual video projectors, but instead of projecting on a screen or flat wall, it maps the digital media on any surface, turning usual objects into interactive displays. In other words, projection mapping is about displaying a visual content on a non-flat surface.

Steps of creating such a mapping include the analysis of a chosen three-dimensional object; a development of a special set of images, animations or videos that will be applied to that object; an application of a special technique that causes optical illusions and precisely aligns them.

The types of projections can vary from simple indoor stage affects to mapping complex architectural landscapes. Typical objects that are used for this kind of performance are: facades of buildings, stage decorations, large 3D objects. In these cases, visual content is usually being displayed on static objects by applying manual alignment between the projected content and the objects. [18]

A projection mapping system can be used in a wide variety of cases, such as:

- product presentation;
- exhibitions;
- advertising;
- live performances;
- entertainment;
- medicine.

Most common projection mapping performances, that focus only on visualization the digital media, use mapping on fixed objects, measuring the surfaces and parameters of each object, and then reproducing these surfaces in a virtual scenario. To achieve an accurate mapping of a scene, this scenario requires the most precise scaling and calibration. Each mapping project can be used only for the original object it was designed for. One of the examples of an industrial projection mapping can be seen in the Figure 14.

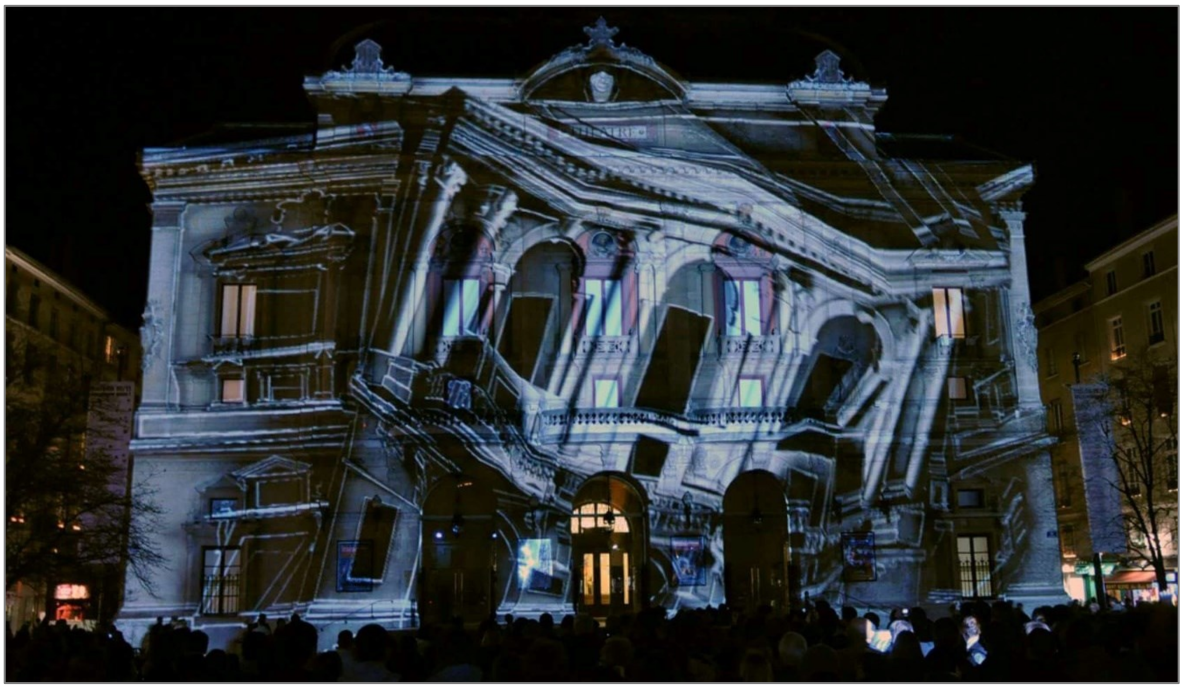


Figure 14. From Gravity to Inception: The Mind-Bending Movie World of Projection Mapping, 2014: bit.ly/1Q78Ank

Another sphere of the projection mapping usage is art performances. Live and often interactive scenes are being applied to the stages in theatres and concert halls. The development of computer technologies allows to use mapping even on moving flexible shapes, such as actor's costume.

A set of special tracking methods was developed to localize actor's position on a stage. Performer's movement modifies the virtual scene by being caught with a help of high-speed cameras, infrared sensors and markers. While being rather independent, these systems still require displays to present data.

One of the recent example of such technologies is "EXISDANCE" – a project that was born from the collaboration between P.I.C.S. and Panasonic groups in 2017. The most modern techniques of real-time tracking and mapping were used to create an outstanding performance that combines visual effects, choreography and martial arts. "EXISDANCE" is an extraordinary performance that expands the potential of a live stage projection mapping. (Figures 15 and 16)

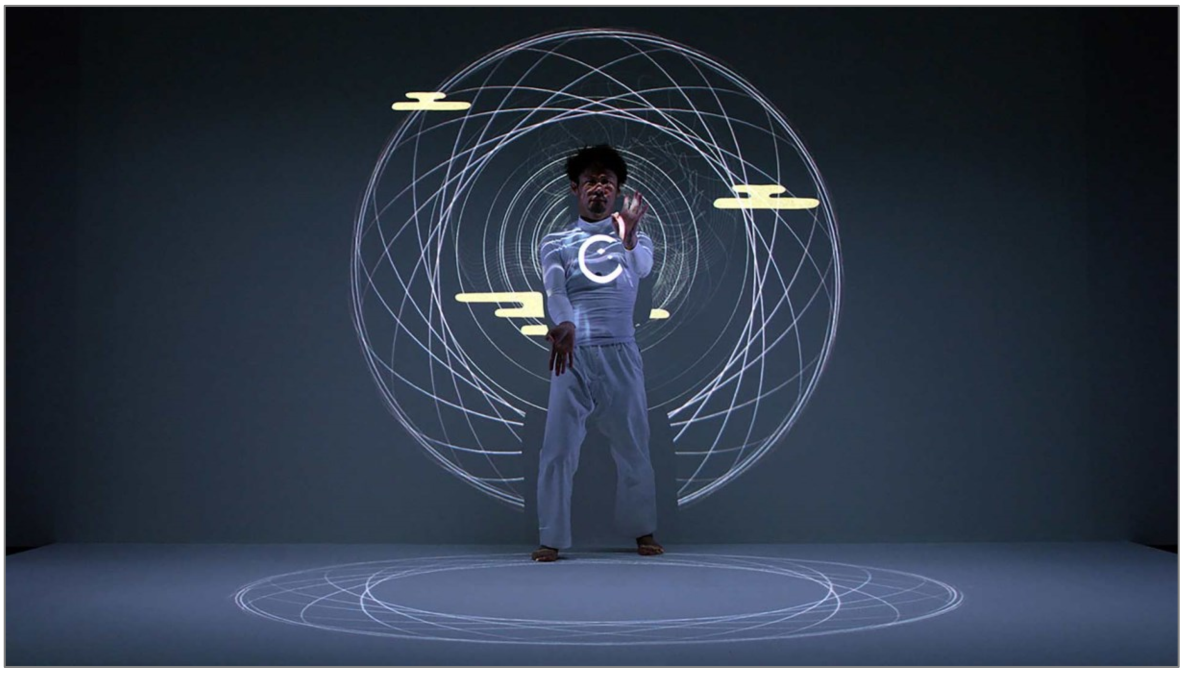


Figure 15. EXISDANCE - Real time tracking & Projection mapping, 2017: https://www.pics.tokyo/en/works/exisdance/

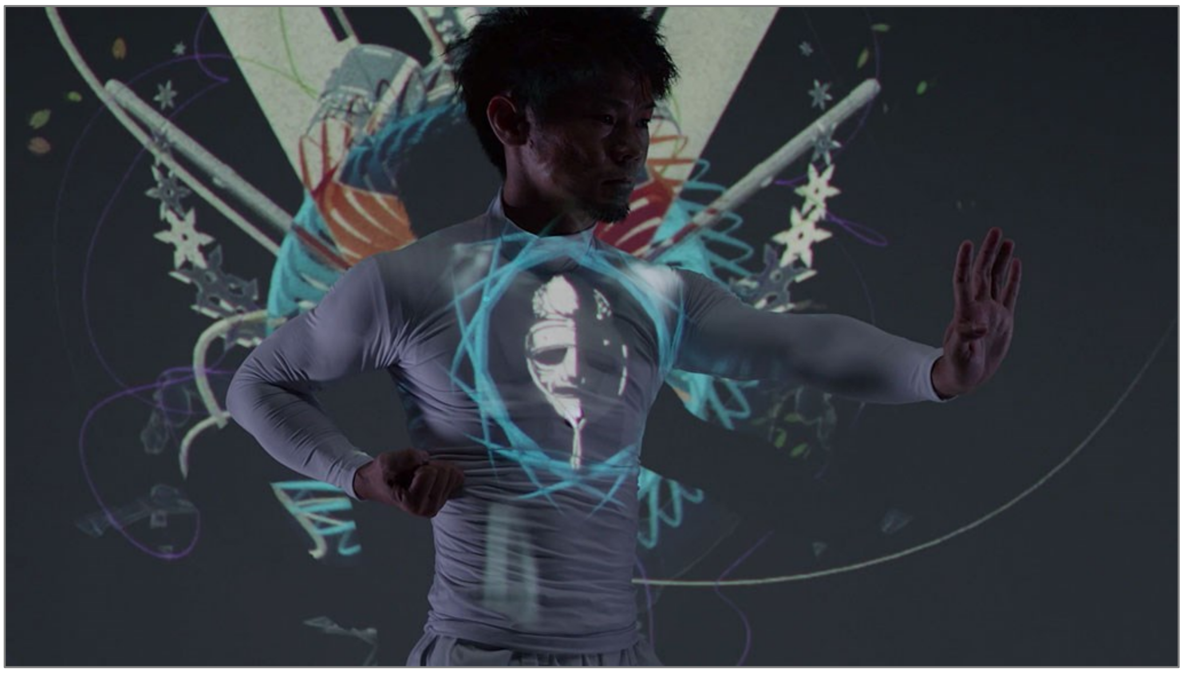


Figure 16. EXISDANCE - Real time tracking & Projection mapping II, 2017: https://www.pics.tokyo/en/works/exisdance/

2.7 Interactive installations

Interactive installation (or interactive art) can be defined as a form of art where an observer or visitor can be involved into a creative process himself. Computer technologies play the major role in creating an interactive installation.

To make an art object interactive, it has to "catch" visitor's actions. This can be arranged with a help of physical cameras and sensors, or, in a case of online-based projects, with computer input devices, such as a mouse, touchpad or webcam. After input information was analyzed by a chosen software, the work responses to these inputs.

Such installations can be seen in art galleries, museums, storefronts, festivals and exhibitions. To provide a viewer with a unique experience, physical objects can respond to motion, temperature and sounds. [6]

In the last decade art installations have been carrying a lot of performative features and creative coding is the most popular tool in making it possible. A number of software environments provide artists with an opportunity to create a unique and individual context. Creative coding is a link between the aspects of performativity and the technical bases.

One example of such an art installation is "The Treachery of Sanctuary" – a project created by Chris Milk - a pioneer of interactive art forms, from physical installations to web projects.

"The Treachery of Sanctuary" is a giant triptych that transforms observers' silhouettes in a surreal way, using infrared sensors, Kinect controllers, screens and projectors.

This art installation consists of three monolithic white frames placed above a reflecting pool. When a viewer enters the space in front of the pool, sensors catch his silhouette and transfer it to the first frame in a form of a shadow against a bright light.

The first panel transforms this shadow into hundreds of small birds that rush aloft to join the flock. Moving to the second panel, the flock becomes larger and the birds begin to dart downwards, attacking the shadow and taking it away piece by piece in their claws. On the third panel the silhouette has returned. On that panel a viewer's shadow gets a par of massive wings that will follow the movements of the arms. (Figures 17, 18)

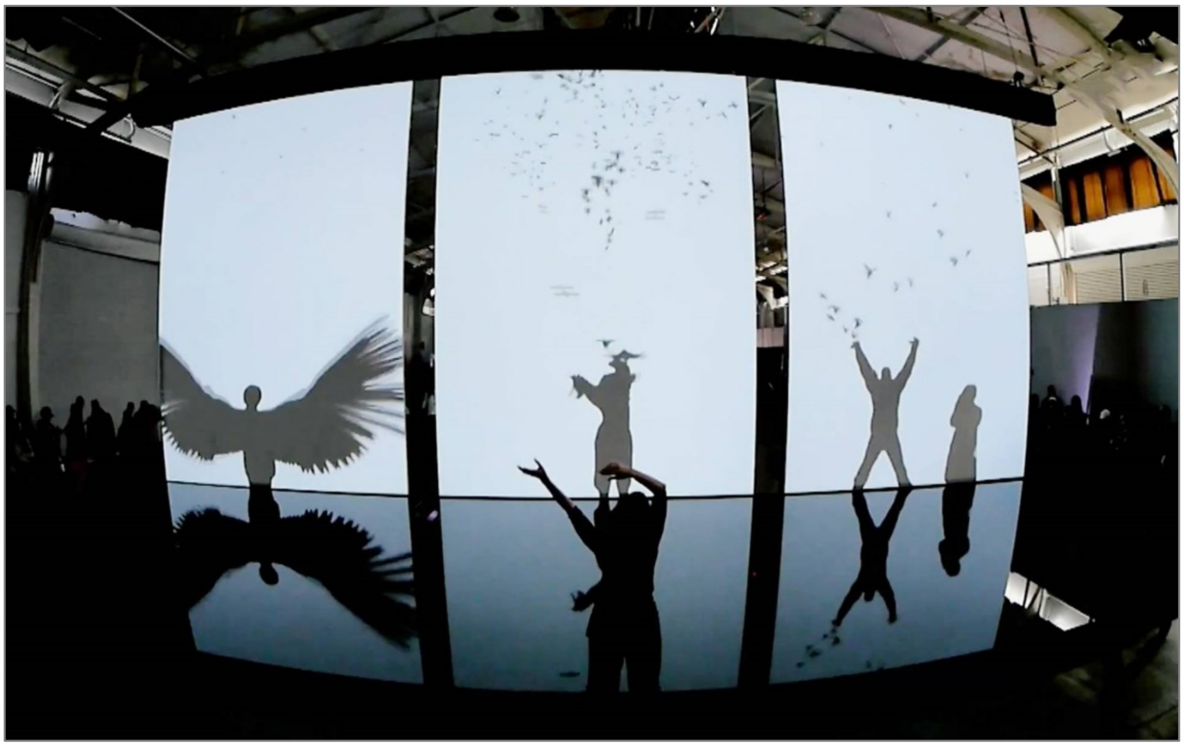


Figure 17. Chris Milk: The Treachery of Sanctuary, 2011:
https://www.radicalmedia.com/work/the-creators-project-the-treachery-of-sanctuary

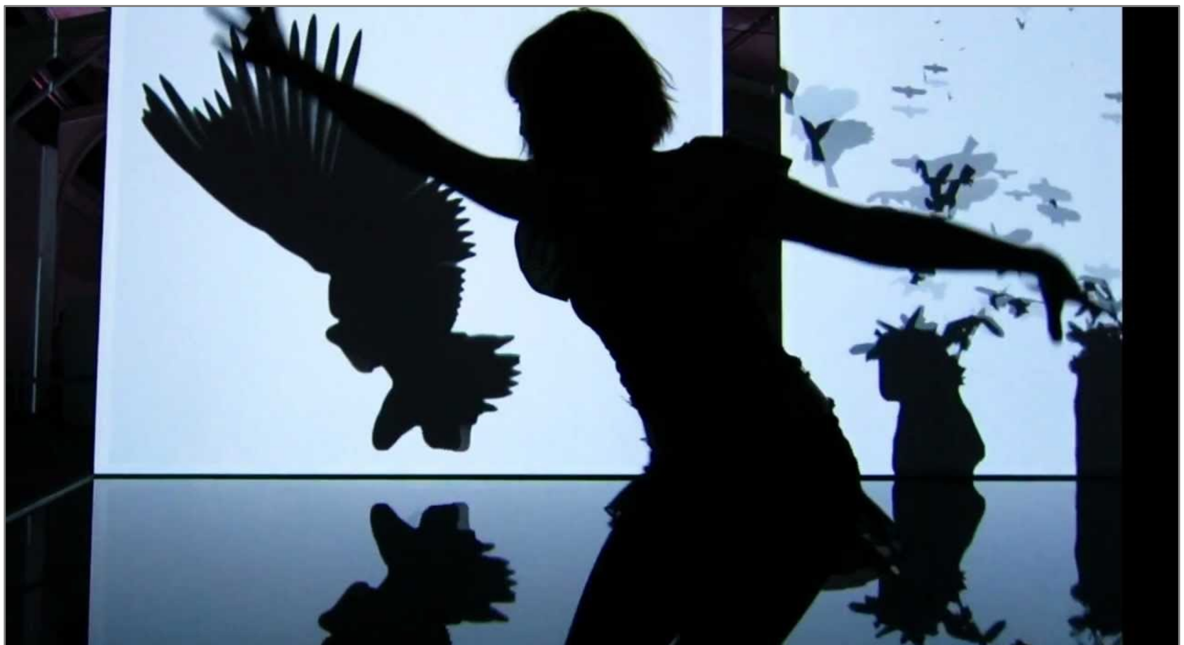


Figure 18. Chris Milk: The Treachery of Sanctuary II, 2011:
https://www.radicalmedia.com/work/the-creators-project-the-treachery-of-sanctuary

Project’s technical director and software engineers describe the technical part: “The implementation of Treachery stitched together several different technologies. We needed a way to visualize the viewers as silhouettes in front of the display so that we could augment their shadows, selectively removing parts or attaching wings. This meant we needed both the outline of the viewers, as well as data points describing their actual posture in terms of torsos, arms, and legs. In order to create a flock of birds, we needed a way of efficiently animating hundreds of 3D models flying together and interacting directly with the silhouettes." [19]

To match all the requirements few software environments were chosen: Unity – a 3D game development software, and openFrameworks – a creative coding platform. These both environments have their unique features that make them cooperate really well. OpenFrameworks has a number of powerful libraries that analyze all the input data from the cameras and, based on this data, Unity creates all the visual effects and animated models that are being projected on a screen. Visitor's movement is caught by Kinect sensors and later transformed into silhouettes interacting with the bird flocks.

3 TECHNICAL ANALYSIS

"The artist must not forget that each of his materials conceals within itself the way in which it should be used, and it is this application that the artist must discover." - Wassily Kandinsky

Main tools of creative coding are various programming languages and environments. A huge number of additional software programs were created with a goal to help coders to develop their projects. Each software has its functionality, so the choice depends on project's direction. Some of the most popular modern toolkits are:

- Processing;
- openFrameworks;
- Cinder;
- Vvvv;
- Pure Data;
- Max MSP;
- GNU Octave;
- TouchDesigner.

3.1 Processing

Processing is a flexible software environment and an alternative language launched by Ryan Hopkins, Casey Reas and Ben Fry in 2001. The main purpose of Processing is to create visual media using code; it promotes programming technologies among artists and visual arts among technicians. After almost two decades of existing Processing has become extremely popular among all groups of people, including students, designers, programmers, artists, performers, researchers and hobbyists. Processing has a number of features that make it a unique environment:

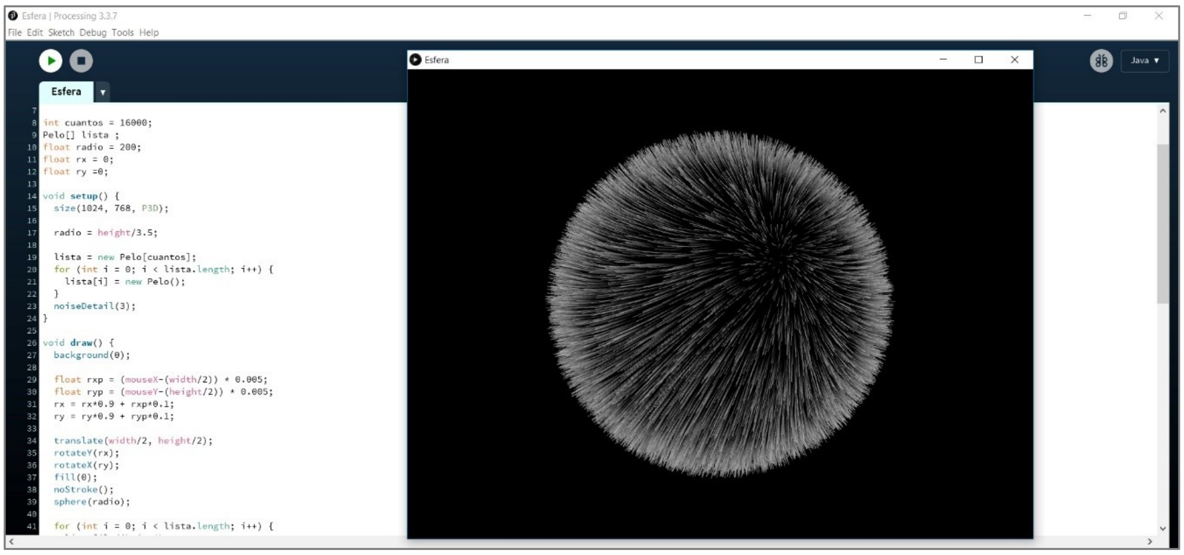
- open source platform;
- free software to download;
- over 100 libraries to extend the functionality, including examples;
- suitable for Windows, Mac OS X, GNU/Linux, Android, ARM;
- interactive programs with 2D, 3D or PDF output;
- OpenGL integration for accelerated 2D and 3D;

- dozens of books, articles and instructions available. [20]

Processing software enables to create programs and applications within the principals of interactivity, motion and visuality. The system represents the mixture of a programming language, software environment and a learning platform.

The main purpose of the Processing programming language is to create and modify visual media. The software combines a simple graphic user interface and advanced set of features. Even a person with no programming experience will be capable of creating a first program after reading an instruction. Processing can render almost any kind of digital media: vector/raster images, 3D models, animations, network communications, data visualizations. It supports many input/output devices: mouse, keyboard, touchpad, cameras, sensors, audio devices etc. [8] Processing is a Java-based language, it structures the code in a procedural, non-object-oriented style. Many commands and functions were simplified, but when the program is compiled, the result is converted to the Java file system, and the classes are interpreted with the Java Virtual Machine. [10]

The Processing Development Environment (PDE) has a user-friendly interface, which makes a process of coding more adjustable for beginners. All the code is placed inside the Text Editor, to activate a process of compilation a user need to press the Run button. (Figure 19) In Processing all the written programs are called sketches and are stored inside a special folder on a user's hard disk.



```

int cuantos = 16000;
Pelo[] lista ;
float radio = 200;
float rx = 0;
float ry =0;

void setup() {
  size(1024, 768, P3D);

  radio = height/3.5;

  lista = new Pelo[cuantos];
  for (int i = 0; i < lista.length; i++) {
    lista[i] = new Pelo();
  }
  noiseDetail(3);
}

void draw() {
  background(0);

  float rxp = (mouseX-(width/2)) * 0.005;
  float ryp = (mouseY-(height/2)) * 0.005;
  rx = rx*0.9 + rxp*0.1;
  ry = ry*0.9 + ryp*0.1;

  translate(width/2, height/2);
  rotateY(rx);
  rotateX(ry);
  fill(0);
  noStroke();
  sphere(radio);

  for (int i = 0; i < lista.length; i++) {
```

Figure 19. A Processing sketch.

Sketches can be rendered as two- and three-dimensional images, depending on an activated mode. If the sketch is rendered in 3D a set of additional features can be applied, such as the material, lightning conditions, position of a camera.

A huge number of additional *Tools* and *Libraries* is available for downloading. These libraries extend the standard functionality of Processing and enable working with advanced features, like complex geometry and sounds. All of these features are products of a collaboration of programmers all around the world; anyone could generate develop a personal library or tool and make it accessible for the rest of a community. Processing has a few *programming modes:* a default Java mode, Python mode and JavaScript mode. By choosing between these three a user chooses the programming platform.

Processing sketches can be exported as *.exe* applications for Windows, Linux and Mac OS X. Another way of exporting a project – saving it as a web page. Mostly thanks to this method Processing has become extremely popular. Web servers host thousands of projects with an access to the source code.

One of the most popular web portals for sharing Processing projects is *OpenProcessing* (www.openprocessing.org). It's a place where thousands of developers, artists, designers, students and educators host thousands of their projects. (Figure 20) Each project, program, sketch can be viewed by any visitor, it's code is available to the public as well, so anyone can examine it. A registered user can copy or "fork" any project to his own folder, modify the code and create a new program based on someone else's work.

The web sites guideline says: "OpenProcessing users worldwide bring wildly different perspectives, ideas, and experiences, and range from people who created their first project last week to the most well-known developers, artists in the world. We are committed to making OpenProcessing a welcoming environment for all the different voices and perspectives in our community, while maintaining a space where people are free to express themselves." [21]

Other popular web sites to share the experience with Processing are:

- CreativeApplications.Net;
- For Your Processing;
- Processing Subreddit;
- Vimeo;
- Studio Sketchpad.

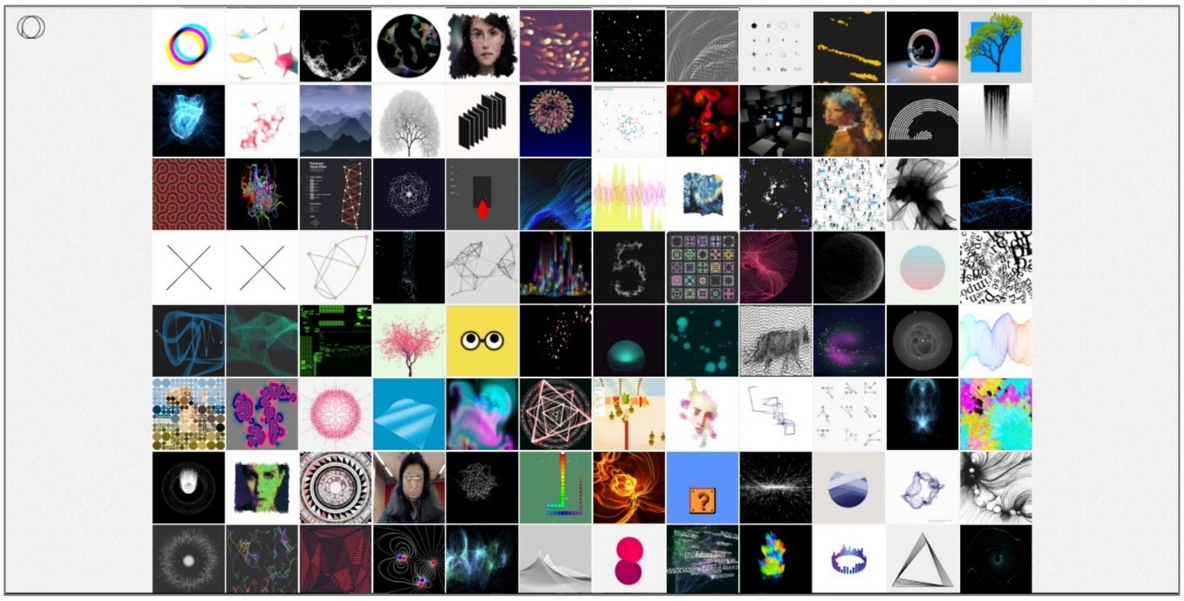


Figure 20. OpenProcessing web portal: www.openprocessing.org

3.2 openFrameworks

openFrameworks is a programming environment founded by Zachary Lieberman, Theo Watson and Arturo Castro. openFrameworks is based on the C++ programming language and its main function is to provide coders with a new simple room for experimenting with code and graphics.

openFrameworks works with a set of widely used libraries, such as:

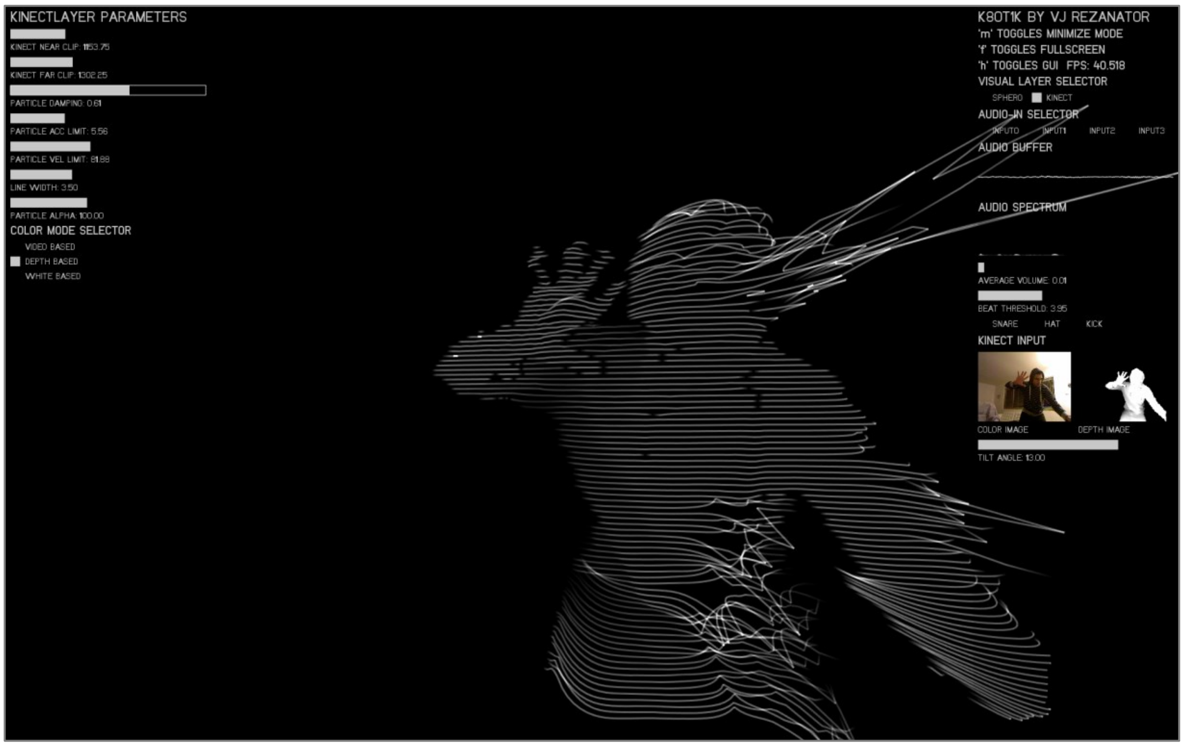
- OpenCV for computer vision;
- FreeType for adding fonts;
- FreeImage for image import/export;
- GStreamer, Quicktime and videoInput for videos;
- GLEW, OpenGL, GLUT for graphics;
- Assimp for 3D-models, etc. [22]

openFrameworks combines a simplicity of use with an advanced set of inbuilt commands and procedures provided by the C++ programming language. C++ is an extension of the original C language, so openFrameworks also allows to create low-level C programs.

The creators of openFrameworks were trying to combine different methods within one environment; to enable creating high-level, advances, complicated programs by the means of a simple and clear approach. Programming in C and C++ allows to extend the functionality by connecting many additional libraries; most of them were designed especially for these languages, so no additional language wrapper is needed. This framework is very powerful: one program can combine the usage of input/output devices, additional libraries and hardware components (e.g. graphic cards). If a program lacks some specific element: a tool or a library, it can be added to the sketch within just a few commands.

openFrameworks is very consistence; the same logic can be applied to any part of the program. Guidelines and instructions make it easy to learn the basics in a short amount of time. openFrameworks can be a perfect groundwork for beginners that only start their path in a sphere of programming. More experienced users can apply their knowledge of any other code language within the rates of this toolkit.

openFrameworks is a cross-platform environment, it is supported on Windows, Linux, Mac OS X, iOS, Android and experimental platforms, such as BlackBerry PlayBook. This feature allows to transfer ideas from one platform to another without any complications; the same sketch can be tested on different laptops and phones. Thus, a user doesn't have to worry about technical components, but develop his original idea. Graphic user interface of an openFrameworks toolkit is shown in a Figure 21.


KINECTLAYER PARAMETERS
KINECT NEAR CLIP: 1153.75
KINECT FAR CLIP: 1302.25
PARTICLE DAMPING: 0.61
PARTICLE ACC LIMIT: 5.56
PARTICLE VEL LIMIT: 81.88
LINE WIDTH: 3.50
PARTICLE ALPHA: 100.00
COLOR MODE SELECTOR
VIDEO BASED
■ DEPTH BASED
WHITE BASED

K8OT1K BY VJ REZANATOR
'm' TOGGLES MINIMIZE MODE
'f' TOGGLES FULLSCREEN
'h' TOGGLES GUI FPS: 40.518
VISUAL LAYER SELECTOR
SPHERO ■ KINECT
AUDIO-IN SELECTOR
INPUT0 INPUT1 INPUT2 INPUT3
AUDIO BUFFER
AUDIO SPECTRUM
AVERAGE VOLUME: 0.01
BEAT THRESHOLD: 3.95
SNARE HAT KICK
KINECT INPUT
COLOR IMAGE DEPTH IMAGE
TILT ANGLE: 13.00

Figure 21. openFrameworks graphic user interface: http://www.creativeapplications.net

The openFrameworks guide says: “We want openFrameworks to be as simple as possible, especially for folks coming from other languages and environments. C++ is a "large" language, large in the sense that you can write very different types of C++ code. If you go to the bookstore, you'll see hundreds of C++ books. We want to create a library where you don't need to be an expert, where at most you might need a book or two, but that the patterns, approaches and style of the code is simple and intuitive. We were especially interested in achieving a sort of parity with Processing, where many of the functions are similar, allowing easier movement from one framework to another.” [22]

3.3 Cinder

Cinder is an open-source programming library launched by its creator Andrew Bell in 2010; it is based on the C++ programming language and aims to expand its visualization abilities. In the last few years Cinder has become one of the most popular platforms for creating graphics by the means of a code. Cinder helps to create various kinds of content: images, videos, algorithms and audio material.

Cinder is a cross-platform environment, it supports Windows, Linux, Mac OS X, iOS and Windows UWP. Cinder is an alternative toolkit to be used instead Adobe Flash, Processing, Microsoft Silverlight. It can be compared to openFrameworks environment; but unlike openFrameworks Cinder’s additional libraries are only developed for better performance and are very specific. [23]

Cinder is normally used outside a browser-bases environment. A powerful support of the C++ language allows to export heavy and complicated applications that can be used for various purposes, including art objects, animation projects, installations, performances, advertising campaigns etc.

Cinder library is suitable for both: beginners and advanced, professional programmers. No major programming experience is needed; it can be a good base for artists who want to try themselves in a sphere of computational art. It is free to download alongside with a number of books and instructions. Cinder has a constantly growing community of user, who share their experiences online. Example of a Cinder program is shown in the Figure 22.

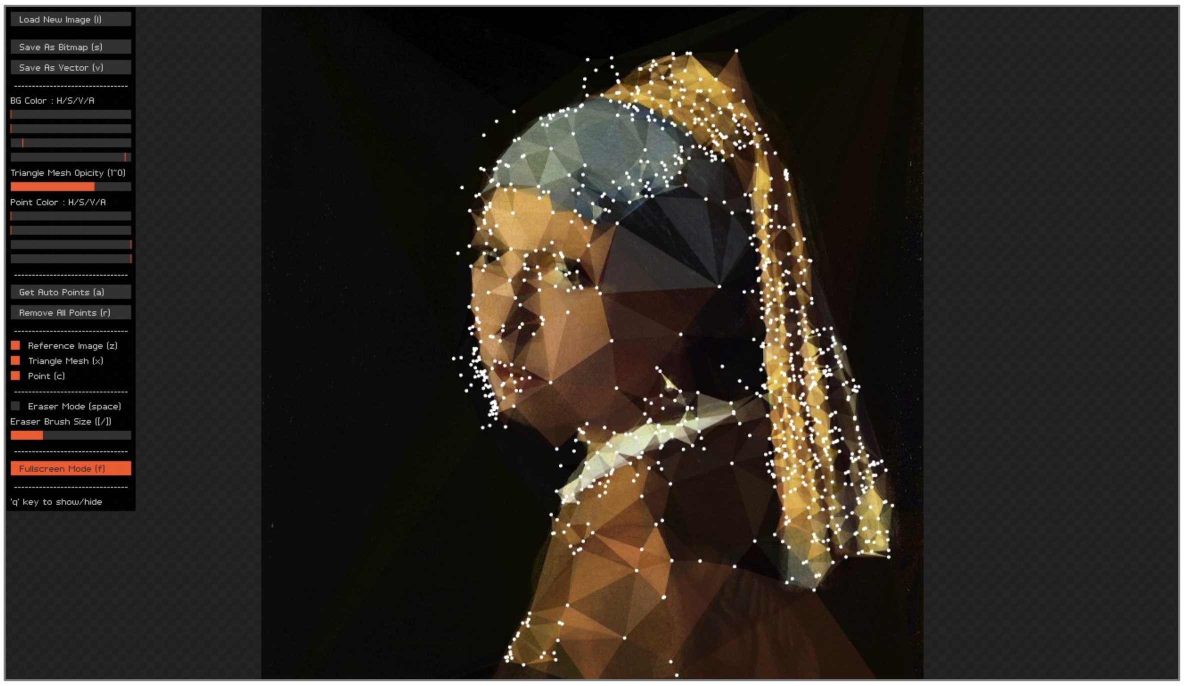


Figure 22. Cinder program: http://www.creativeapplications.net/mac/dmesh-cinder-mac/

3.4 Motion detection

Oftentimes creative code applications are based on a principle of interactivity; a program receives an input data and based on this information does some modifications to the visual outcome. Thus, a moving object detection becomes a key technology for such kind of projects.

Motion detection can be defined as a process of capturing a change in object's position in relation to its surroundings and the device itself. A motion detector is a special device that captures the movement of a physical object. Some devices detect a presence of a person within a certain space, others capture sounds or any disturbance in a surface balance, like seismometers.

A motion detection (or activity detection) system is usually based on a specific monitoring algorithm/function built inside a chosen software program that captures a physical motion from a camera and transfers it into a set of values. Normally, the motion itself is being captured; however, more advanced technologies can recognize the exact type of motion, like a specific gesture.

The use of motion detection systems is very diverse. It is a popular tool for all kind of artistic performances; numerous applications use an automatic tracking of a moving object in order to produce more entertaining results. Visual-based intelligent systems require a more accurate approach for calculating a change in object's position.

The process of motion detection starts by recording a video image with a camera/sensor, frame by frame, comparing these frames to each other and defining the points of distinction. The image might be processed with a help of various filters, some of them work with image's resolution, some with color-depth and smoothing, others with background removal. [24]

Motion detection systems work with a support of different technique equipment:

- RGB cameras;
- infrared sensors;
- laser sensors;
- acoustic sensors and microphones;
- magnetic sensors;
- radio and microwave sensors.

The motion detection system can be either mechanical or electronic. The standard form of mechanical motion detection is in the form of a switch or trigger that you have to operate with manually. The most popular techniques used for motion identification is an optical detection that includes the usage of standard cameras, infrared and laser sensors.

3.4.1 Microsoft Kinect

Microsoft Kinect is a line of motion sensing devices produced for video games consoles, such as Xbox 360 and Xbox One, and for Microsoft Windows PCs. Each devise has a number of cameras and sensors that enable a user to interact with a computer or console device without any additional game controllers. Originally designed for a video games industry, Kinect has become a popular tool among programmers, artists and performers that use it as a part of their projects.

Since its first release in 2010, Kinect devices have gone through a number of changes. Few versions of a software/hardware combinations have been produced:

- Kinect for Xbox 360 (2010);
- Kinect for Windows - software development kit (2012);
- Kinect for Xbox One (2013);

- Kinect for Windows v2 - a software development kit for the second generation of Kinects (2014).

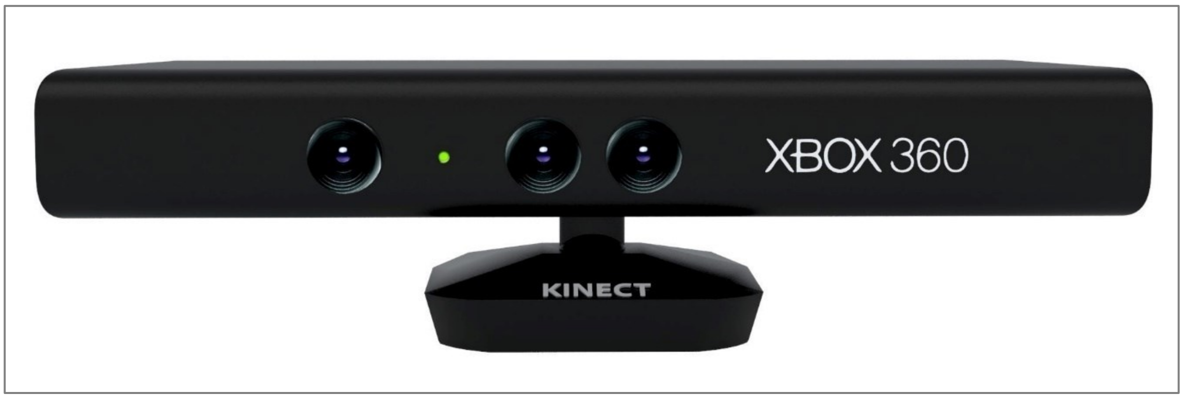
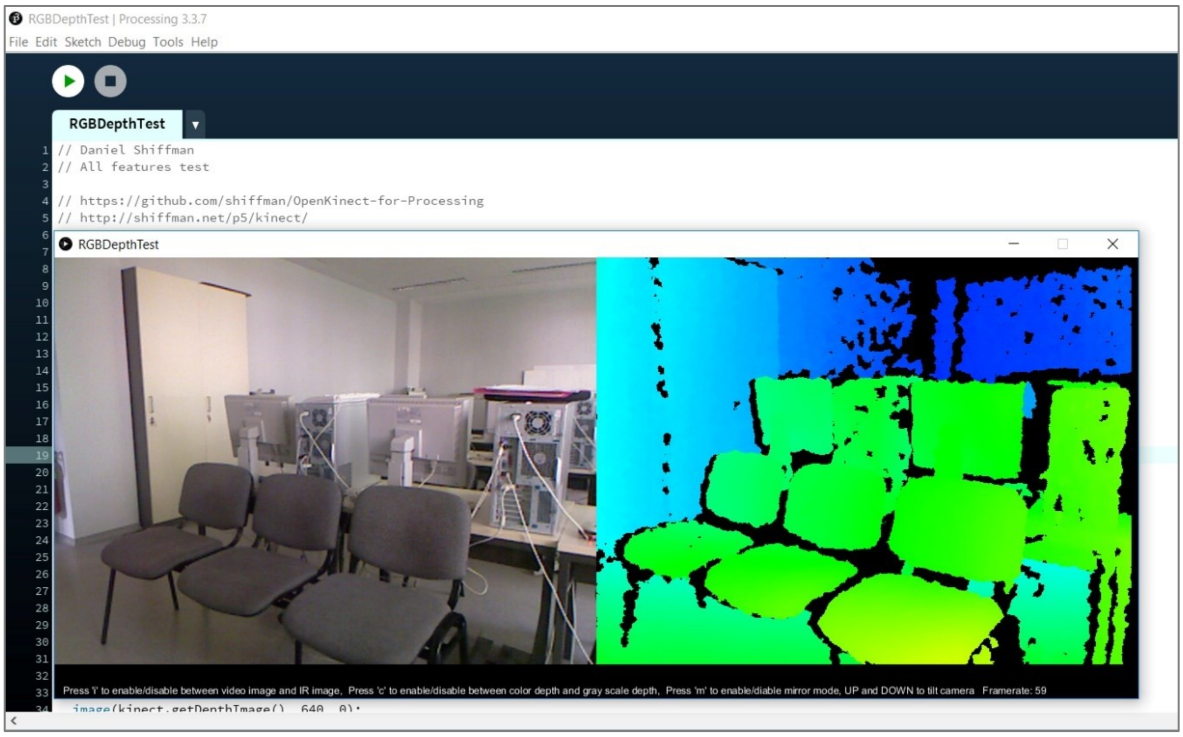


Figure 23. Microsoft Kinect Xbox 360: https://www.xbox.com

Kinect for Xbox 360 (Figure 23) is a combination of a hardware/software system. The hardware includes a set of cameras and a special microchip, designed to calculate an object's position in a 3D environment. This scanner system is called *Light Coding*, and it allows to reconstruct objects and space applying different rendering variations.


RGBDepthTest | Processing 3.3.7
File Edit Sketch Debug Tools Help
RGBDepthTest
RGBDepthTest

```
// Daniel Shiffman
// All features test

// https://github.com/shiffman/OpenKinect-for-Processing
// http://shiffman.net/p5/kinect/
```

```
Press 'i' to enable/disable between video image and IR image,  Press 'c' to enable/disable between color depth and gray scale depth,  Press 'm' to enable/diable mirror mode, UP and DOWN to tilt camera  Framerate: 59
```

```
  image(kinect.getDepthImage(), 640, 0);
```

Figure 24. Kinect RGB depth test with Processing

The Kinect device has a horizontal panel with an RGB camera, infrared camera, depth sensor and a microphone. The panel is connected to a stable base with a motorized pivot. Kinect is able to record video images, capture and reproduce 3D figures, recognize motion, faces, gestures and voices. The depth sensor enables to capture image not depending on lightning conditions. Its sensing range can be specified inside the setting of a program that is used for motion tracking. This sensor is able to distinguish a human figure from a piece of interior. The test of a depth camera is shown in the Figure 24. [25]

Later versions of Kinect devices replaced some of the sensors with never tracking technologies, such as a time-of-flight camera for Kinect Xbox One. This advanced version of Kinect has a greater sensitivity of all the cameras and sensors with a 60% wider field of vision.

Although, the Microsoft company terminated the production of Kinect devices in 2017, they are still very popular among creative coders and artists that want to bring some interactivity into their projects.

3.4.2 Leap Motion

Leap Motion, Inc. is an American company that produces a special device which's purpose is to capture a hand and figure motion, analyze it, and convert into an input information, like the data a computer gets from a mouse or a touchpad. The difference is that no actual physical contact is required. This technology is still very new. The company launched new hand tracking software in 2016, in order to extend the possibilities of virtual reality.

The device itself is called a Leap Motion controller, it looks like a small rectangular box that connects to a computer/laptop with a USB cable. In order to be as accurate as possible it has to be placed on a physical desktop. (Figure 25) Sometimes these devices are already built into a virtual reality headset.

The device has a panel with two infrared cameras and three infrared light-emitting diodes. A controller can recognize a motion on a distance of 1 meter or less. The data is being recorded at a rate of 200 frames per second. This information is being received by a computer through a USB cable, where a Leap Motion software analyzes it and performs the required command.

Sharing some basic principals with a Kinect device, Leap Motion has a set of significant differences. It has a much smaller field of view and a higher image resolution. Kinect is used for a full-body tracking, while Leap Motion catches the most elusive movements of a finger. The device's creators claim it to be 200 times more sensitive than anything that has been produced before.

Leap Motion system supports a number of various features and possible fields of usage, such as:

- using it instead of a mouse to navigate through an operating system or web pages;
- interacting with any visual context;
- navigating through maps, zooming in and out;
- using a hand instead of a drawing tool to create digital images;
- interacting with 3D models and data-visualizing systems;
- signing digital documents;
- playing computer games. [26]

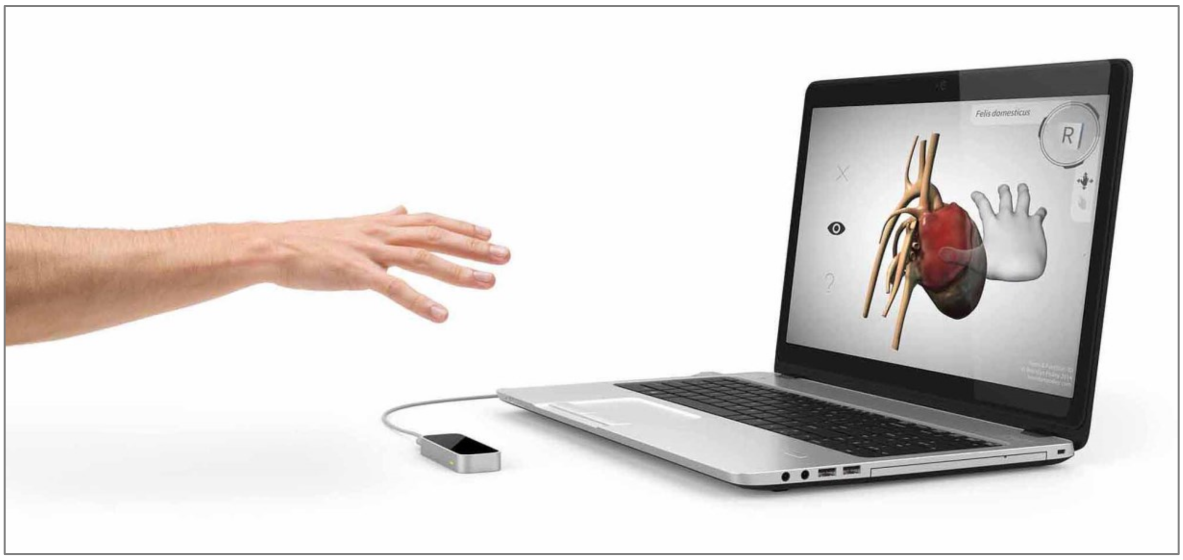


Figure 25. Leap Motion controller: https://gallery.leapmotion.com

II. PROJECT

4 THE PROCESS OF CREATION

4.1 The challenge

The purpose of this project is to investigate the possibilities of programming within the confines of a visual media. More specifically – the ability of code to create art. After the analysis of the most common techniques, performances and art objects, the form of interactive installation was chosen. Such installations become relatively popular nowadays, thus, one of the challenges is to create a unique approach to this practice.

The installation should consist of a number of graphic objects – changing, "growing" fractal-like images and animations – generated with code, that are being projected on a wall. The images should interact with a viewer – change their shapes or speed of motion, based on a viewer's position in a room. The principle of this interactivity will be maintained with a help of motion detection technology.

The development of the project goes through a few steps:

- finding an idea/theme and inspiration behind the code;
- choosing a programming language and environment;
- creating the first programming sketches, finding the most promising algorithms;
- developing a unique set of code sketches with a shape and color variability;
- choosing a few sketches that will be used in the final installation (5-10 sketches);
- combining these sketches into one working application;
- adding the principle of interactivity;
- preparing all the material for the physical installation.

The final installation will take place in a shaded area, where any visitor could interact with the images, change their appearance and possibly the order of projecting. The overall impression will be supported by an audio material.

The set of unique programming sketches is the main part of the project, yet the most challenging one is combining them together into one application and applying the principle of interactivity.

4.2 Philosophy behind the code

“You can't wait for inspiration. You have to go after it with a club.” - *Jack London*

Like any other creative process, code art requires an inspiration. An idea behind the project is always one of the major parts of its development. With creative coding it determines the progress of the project in general, the approach to the usage of programming commands and algorithms, the visual outcome. Besides, an art object’s purpose is to make a certain impression on a viewer, be memorable, which has more chances to succeed if its creator has put a thought into the concept.

As an ancient Greek philosophy enthusiast, I’ve always been fascinated by its codependence with the world of precise science. Many of these theories can be interpreted in a way to describe even the very recent scientific researches. One of the most inspiring philosophic concepts, to my mind, belongs to a Greek pre-Socratic philosopher – Empedocles.

Not just a philosopher, but poet, physician, prophet and orator, Empedocles was trying to unite all the knowledges he possessed, which includes astronomy, ontogeny, sociology, anthropology and religion into one doctrine. Most of his studies are summarized within two poems: “On Nature” and “Purifications”.

According to his believes, there are no birth and death, the beginning and the end; life itself is looped within the eternity. Every matter is a result of a mixture of four fundamental elements: earth, air, fire and water. When these elements are combined together, the world we know with all the living beings is created; separation of the elements brings disruption and decay. This cycle repeats itself under the influence of two major forces: Love and Strife. Love symbolizes attraction, gravity, condensation; Strife – repulsion, rejection and decomposition. In other words, these two powers are diametrically opposite concepts, but only in their cooperation the universe can exist.

Empedocles wrote that during the stage when Love dominated upon all the elements, everything was united into a sphere – a perfect harmonic figure where all the elements are represented as one. Strife’s intervention brought disruptions and resulted in a complete separation of the elements. Nevertheless, neither the perfect unity, nor the total chaos can produce life, it is doomed to suffer in a never-ending conflict of these opposite forces.

The application of this philosophy to the programming code can suggest a new approach of its understanding and interrelation with the world of modern technologies. The original idea is focused around a sphere – a starting point of the program. The sphere begins its development – it grows and expands. As soon as a viewer's movement is being detected by a motion detection device – the sphere starts to decay, turning from a precise clear figure into a chaotic mass. Thus, human integration represents the eternal resistance of these two fundamental forces. The color spectrum is being influenced by four major elements: red represents fire, blue – water, white – air, green – earth.

4.3 Tools of choice

Choosing your tools is always a key decision to make. The programming language and platform will determine the whole process of work and the material it will result into. The choice has been made between three popular environments: Processing, openFrameworks and Cinder. Each one of them has a lot of advantages to offer, so no matter what the decision is, it couldn't be wrong. All three platforms have tight connections, their main goal is to popularize the use of programming in a world of art and design. Coders from all around the world share their experience online on a numerous forums and galleries, dedicated to these environments, inspiring each other for the new challenges.

Processing's big advantage is that it has a great number of guides, instructions and documentations for the beginners. Many books are available online, as well as video lessons and open source examples. Processing and openFrameworks offer a wider choice of platforms they support: Windows, Mac, Linux and Android. Processing is an alternative version of Java, while openFrameworks and Cinder run in C++, which means that the second group operates a little bit faster. Although, if no massive flow of commands is planned, this shouldn't be very noticeable, it's more about a language's logic preferences.

After a careful research through all the possibilities, a Processing language and software was chosen. The key points were the existence of hundreds of open source examples, tutorials and references, great number of additional libraries that support all possible variations of a visual rendering. Another advantage is Processing's support of additional input-output devices.

The whole set of software and hardware tools required for the project includes:

- a laptop with Windows 10 operating system;
- Processing development environment, version 3.3.7;

- a set of additional libraries from processing.org;
- Kinect for Xbox 360;
- Kinect for Windows Software Development Kit (SDK);
- Kinect Studio, version 1.8.0;
- Zadig application, version 2.3;
- additional LibusbK drivers for Kinect, version 3.0.7.0.

4.4 Hello, World!

The thing about creative coding that makes it close to any other artistic process is that you can start programming without having a precise idea of an outcome in your head. A coder writes a few strings of commands, sees its visual representation and then comes up with an idea how to develop a program.

The aim of this first stage is to come up with a satisfying programming algorithm that will initialize the motion of the figures, their growth and change. Figure's exact shapes and colors could be specified later.

I started playing with code by creating the perfect figure - a circle, and then bringing some distortion into its shape. The first changes were pretty simple – adding some edges to the circle's border, making it more angular; then bringing more little twists and turns to smooth the figure out. A principle of randomness helps to make it look more chaotic, which was the original idea – a progress from order and circle to chaos. What makes the sketch entertaining is to see these changes in real time, like an animation; especially, if this process differs every time when a user refreshes the program.

The main principal of this algorithm is the differential growth: by controlling the rate at with different parts of the original shape grow, I can control the shape everything grows into. Examples of such shapes are fractals, crystals, some plants. The process can be simulated using few methods:

- the principal of attraction: different points on the figure's surface move closer to each other, joining into new shapes;
- repulsion: the distance between the points will grow, causing the shape's expansion;
- separation/split: the long distance between the points will eventually cause their separation that will result in splitting the original figure into a number of new ones.

Combining all of these methods within the borders of one sketch might bring interested results. Only a practical experience will help to come up with the final algorithm, so a number of first experimental sketches was created. (Figure 26)

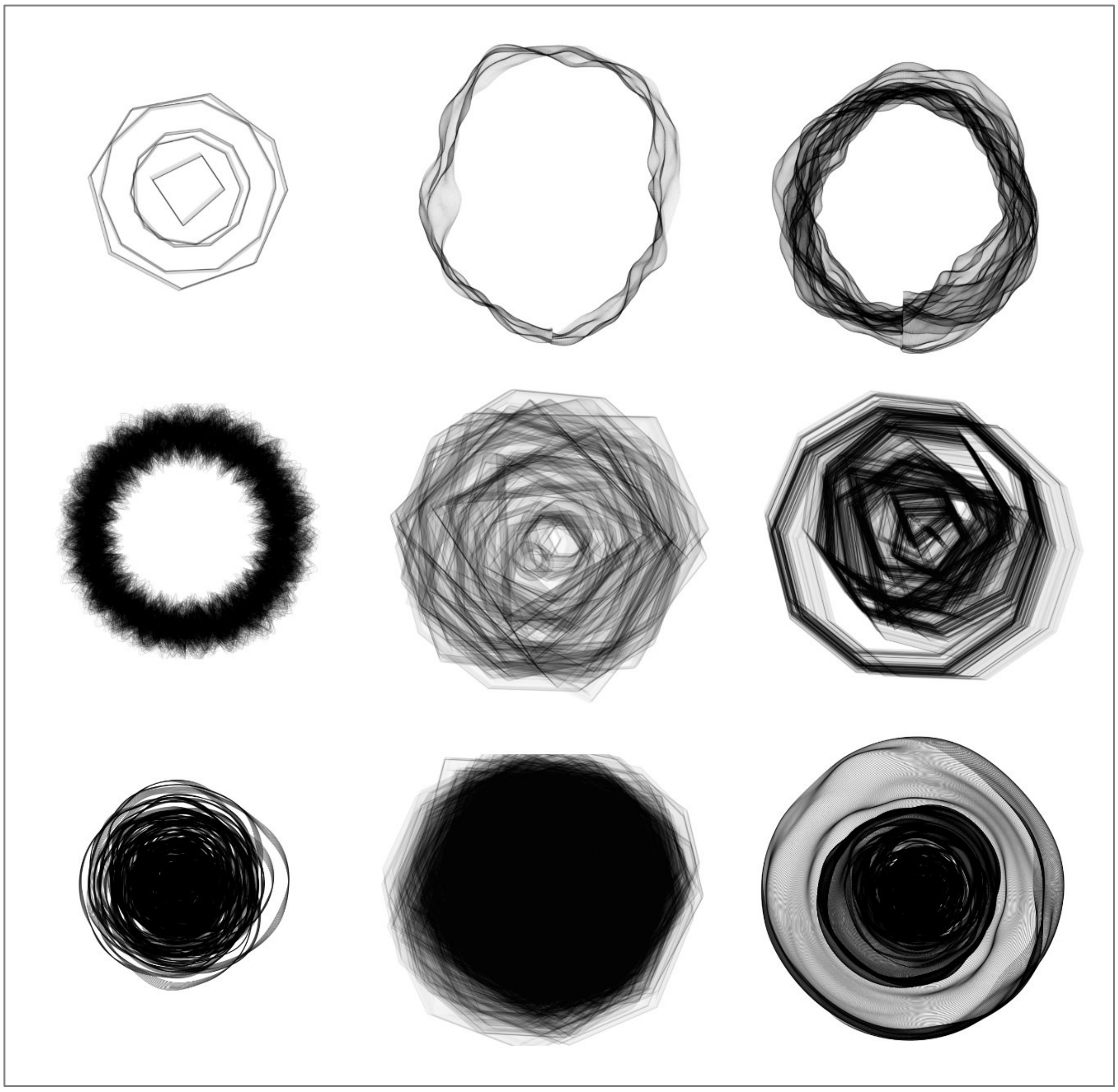


Figure 26. First Processing sketches

4.5 The final algorithm

It was decided that the final project will be represented by a number of programming sketches that share the same principal of movement, dynamics, aesthetics and are ruled by a core algorithm. With circle being the basic figure of the program, the first function initiates the creation of a single small ring. Based on a calculation of its position the next function starts to add more and more rings, one by one, at a random distance from each other.

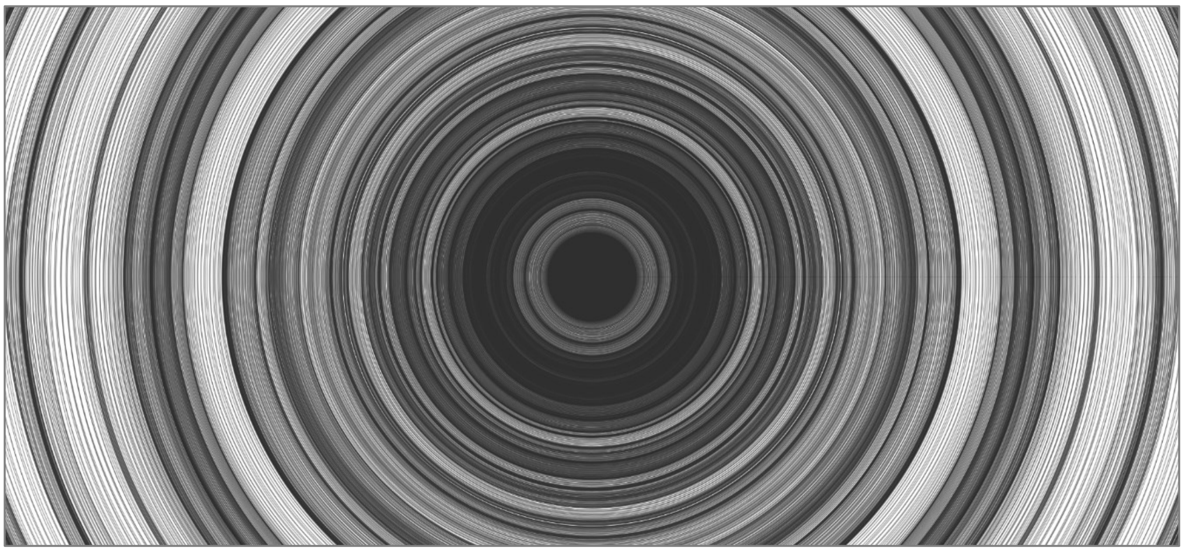


Figure 27. Processing sketch: "Roundel"

That far the growth has been occurred in a uniform way; to make the final image look more entertaining, the initial shape should be less accurate. One of the simplest ways of bringing some variation in its growth is to prioritize certain points of the shape and randomly apply a greater increment or decrement. The higher the value of this increment/decrement is, the more the initial rings are being disturbed and start to gain a crystal-like shape. (Figure 28) One of the advantages of this method is that it gives the figure some variations each time it is being drawn. After a number of failed and successful experiments with code a final algorithm that controls the movement was created. Thus, each sketch will be generated in the same way, but might have different properties.

The algorithm starts with announcing the variables that control the overall appearance of the figure. This is an example of main characteristics and their values for one of the sketches written in the Processing programming language (the result is shown in Figure 29):

float increment = 2.8; - the step of growth increment;
float size = 0.91; - the size of the first ring;
float multiply = 1.001; – the value each ring is multiplied by;
float widthOffset =50; - ring's horizontal offset;
float heightOffset = 50; ring's vertical offset;
float offsetStep = 0.001; - offset's step.

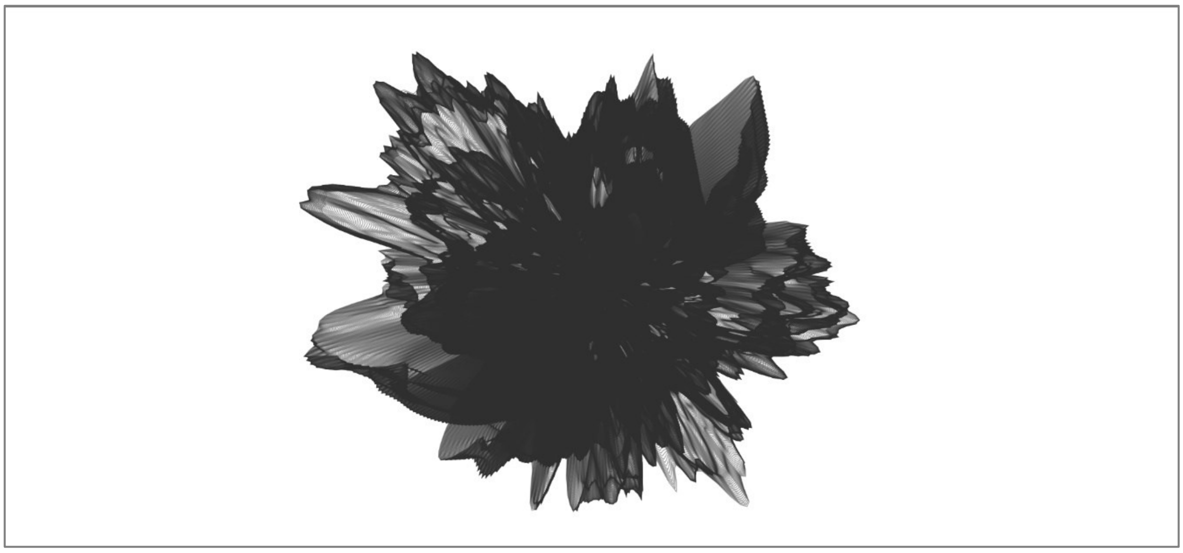


Figure 28. Processing sketch: “Chrystal”

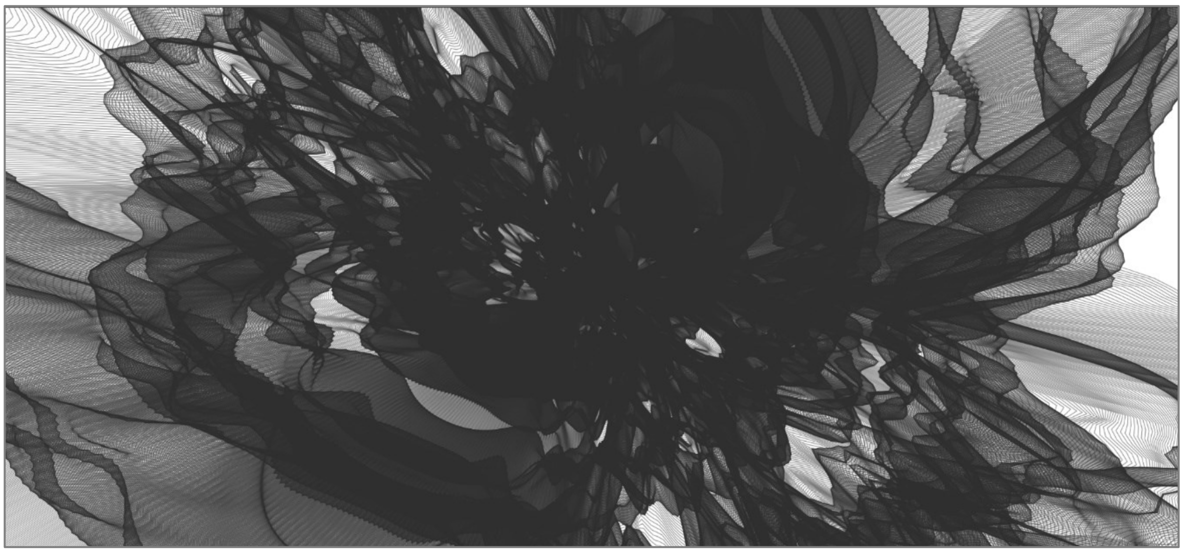


Figure 29. Processing sketch: “Mess”

This is just one of the possible outcomes, the exact number of the variations is incalculable. By changing the values, I can control the overall visual appearance of the sketch. The rings of the figure can be disturbed in any way, split apart, multiplied.

Additional parameters can be changed inside the main function that controls the movement, like the amount of noise applied to the sketch, number of chosen points along the circumference of a ring. Different configurations result in different visual images. The next challenge is to decide on exact shapes and apply a color palette.

4.6 Color palette

A color palette is defined within the setup function:

```
function setup() {
  background(255, 255, 255);
  stroke(48, 48, 48, 128);
  blendMode(BLEND);
  noFill(); }
```

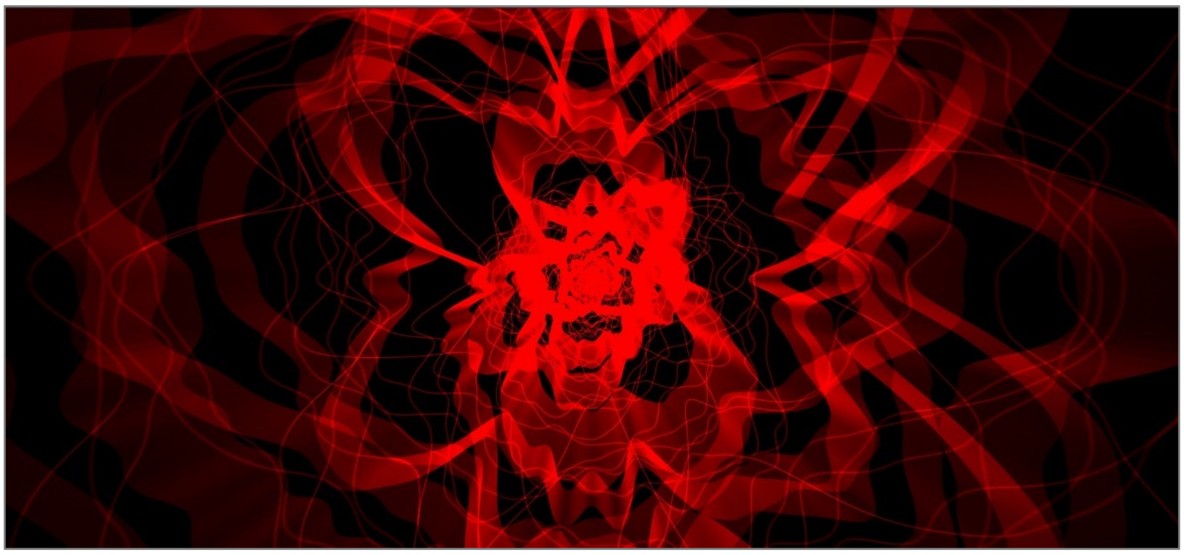


Figure 30. Processing sketch: “Red”

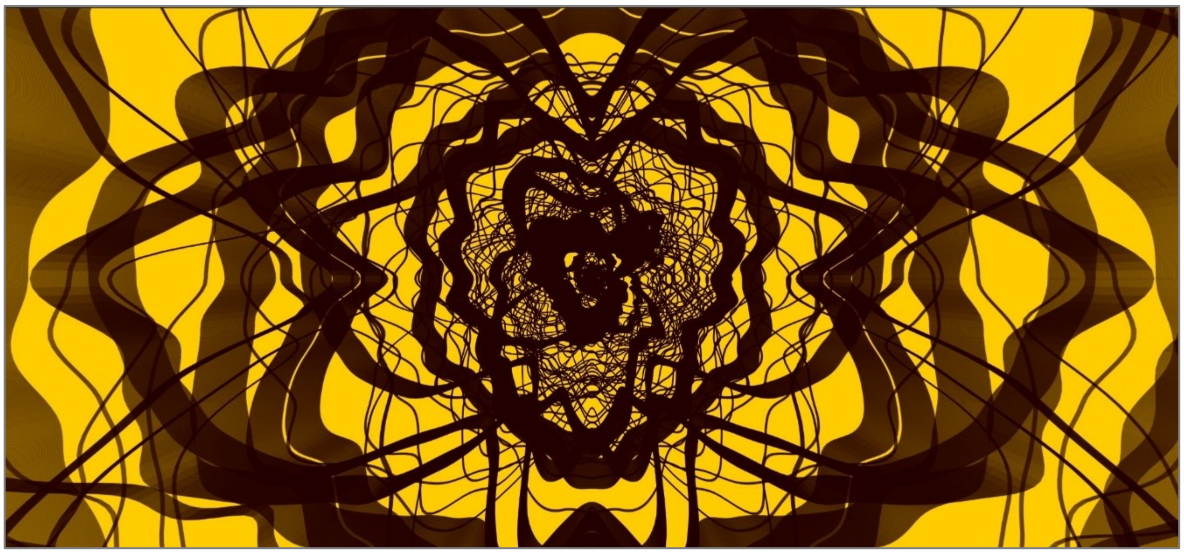


Figure 31. Processing sketch: “Yellow”

Any shade from the RGB spectrum can be applied to the background and rings. The way these features are being mixed together depends on a *blendMode()* command. There are ten different modes that can be chosen. Two of them are being used in this project: *BLEND* – to simply add the strokes color to the background (Figures 30-31), *ADD* - to vary the color range depending on stroke's intensity (Figures 32-33) – from the brightest tones to the darkest. The general palette that was chosen for the set of final sketches is varicolored, yet it was decided to avoid using many colors within one render. The final results use one color for a background, one color with its difference in a level of saturation for the figure itself.

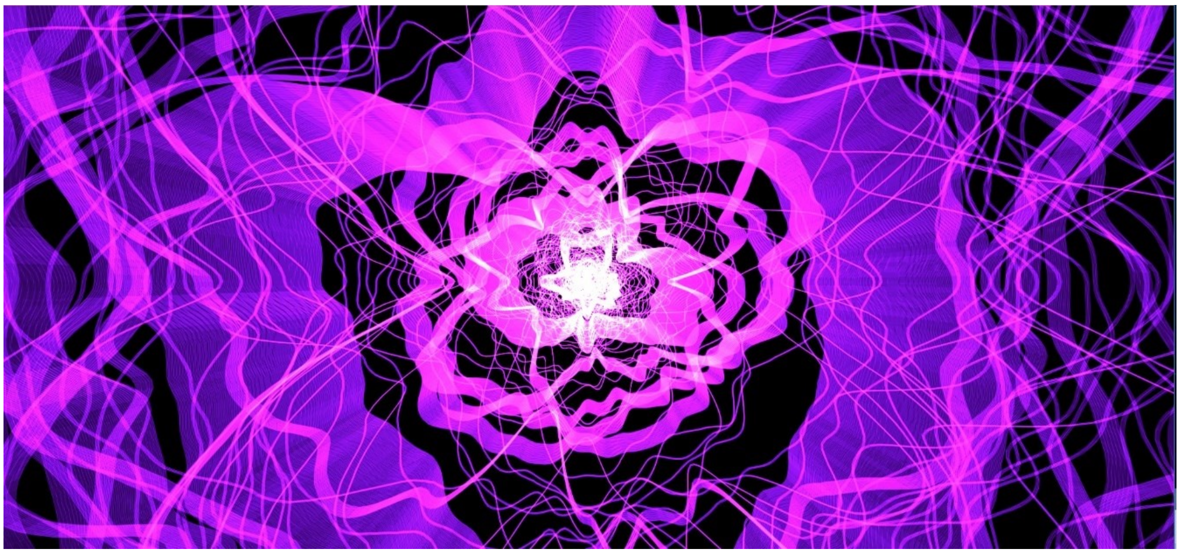


Figure 32. Processing sketch: "Purple rain"

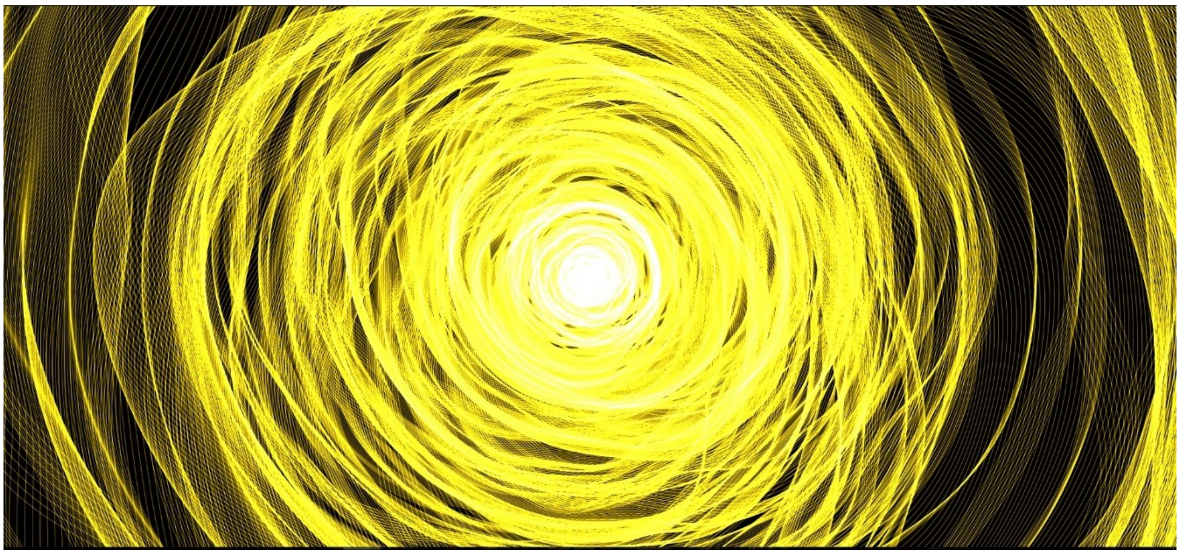


Figure 33. Processing sketch: "Solar"

4.7 One sketch to rule them all

Each sketch written in the Processing environment can be saved as an *.exe* application. One of the main challenges of the project is to combine all these sketches in one, so there will be no need to reload different applications during the presentation, all of them have to run under a single command. Few solutions were suggested:

- create an additional Processing program that will load the needed sketches;
- use another program to unite them (e.g. Eclipse IDE);
- use a tool that switches between the *.exe* programs;
- combine the source code from all the sketches within the bigger one.

The last option was chosen. The advantage in this method is that all the sketches will be rendered in one window without closing it. The possible disadvantage is that the program might use too much space from the operative memory and will become too massive to be executed smoothly.

The code has to be divided into a few parts. First, the main "*parent*" class that initiates the render:

```
public abstract class Main
{
 protected PApplet parent;
 public Main(PApplet parentApplet) {
    this.parent = parentApplet;
 }
 public void init() {};
 public void display() {};
}
```

After that a number of sketches that will be used in the final application; all of them have to be put inside other classes according to one template, all the original properties are described inside the *init()* and *display()* commands:

```
public class sketchA extends Main
{
   public sketchA(PApplet parentApplet) {
  super(parentApplet);
…
 }
    @Override
    public void init() {…};
   @Override
  public void display() {...};
}
```

Then goes the general setup for the window where the program will be rendered, including the size of a screen and a background color in case if it is not specified in the previous class:

```
void setup() {
  fullScreen();
  background(255);
  rectMode(CENTER);
...}
```

The last part of the code initiates the *draw ()* function that calls the display method of the selected sketch described within the classes before:

```
void draw() {
  apps.get(selected).display();
}
```

4.8 Interactivity

Interactivity has become a key feature in a sphere of digital artworks, it is something what makes the project "alive", lets a user or viewer make a contact with any content that is being displayed on a screen, projected on a wall or visualized with any other available methods.

There can be various definitions of what interactivity is, but the main factor is that user's interaction with a project should have its impact on a visual result. According to a previous theoretical analysis, interactivity can be divided into a few levels:

- a project is interactive to a degree that only a user's presence/movement/action generates its visualization;
- a project has a prearranged material and user's interaction with it brings a specific change into what has already been displayed;
- different actions of a user result in different kinds of changes applied to a project.

The main idea or philosophy of this project directly depends on a level of interactivity. The original idea involves a set of rendered sketches being projected on wall of a shaded room and a visitor, whose movements are being caught by a motion detection system, analyzed by a program, turned into specific values and applied to the visual outcome.

There are several methods of extracting the necessary data, including regular RGB cameras, depth cameras, stereoscopic cameras, etc. Some devices support a complex gesture recognition, when a single movement of a hand can be detected; some are sensitive to a change of lightning conditions; some of them have inbuild filters that help to “filter out” an irrelevant data. All of these techniques can expand the interactive alternatives, to make a right chose an exact type of target and a kind of its movement should be defined.

It was decided that a viewer’s exact position in a relation to a motion detector will be a base for interactivity, so the whole image of a visitor is more important than the detailed analyzes of his movements. Lightning conditions should also be taken into account; projecting in a dark room makes it impossible to use regular RGB cameras. Thus, a Microsoft Kinect Xbox 360 and its Infrared/Depth camera were chosen. A newer version of Kinect is available, but it is supported by a fewer libraries of Processing.

4.8.1 Kinect depth test

Kinect’s infrared camera and sensor have a number of characteristics:

- can stream video image directly, without converting it into a depth map, its resolution is 640×480 or 1280x1024 pixels;
- convert video stream into a depth map in VGA resolution (640×480 pixels) with 11-bit depth;
- the depth has 2,048 levels of sensitivity;
- the sensor has a standard limited distance range when used with the Xbox software: 1.2 – 3.5 meters. However, this distance can be extended to 0.7 – 6 meters;
- sensor’s angle of view is 43° vertically and 57° horizontally, the device itself can lift (or lower) the panel with cameras up to 27°.
- the vertical view of Kinect’s sensor reaches about 63 cm at the minimum distance, the horizontal – 87 cm, which results into 1.3 mm per pixel resolution.

All of these features can be controlled inside the Processing program, the exact adjustment depends on parameters of a room where an installation will take place. Minimum/maximum sensitivity can be set in a way to only capture the closest moving objects, while all the background will be filtered out. (Figure 34)

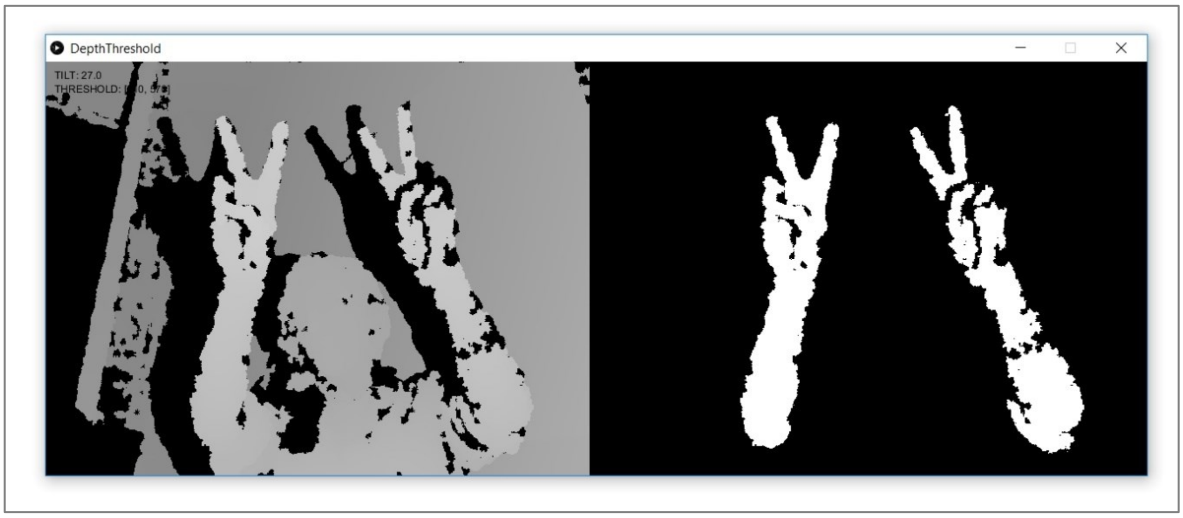


Figure 34. Kinect depth test

4.8.2 Points of interaction

According to Empedocles, the first stage of a life cycle is peace and order, a cosmic harmony. Strife's invasion started to separate the elements from each other, leading the world into chaos and decay. Visitor's presence and movement in a room where the installation will take place represents this force. The more visitor moves towards the camera, the more distortion appears in the image projected on a wall. The progress of this distortion is shown in the Figure 35, where the first step represents the peace and no interaction, the second captures some movement at a bigger distance and the third – movement close to a camera.

Variables, influenced by that movement, are the horizontal and vertical offset of the rings. The program captures visitor's movement and calculates the distance at which it occurs. This new value in being added to the level of an offset applied to the X-axis and Y-axis, which was set as neutral before. Thus, the closer to the camera a visitor is, the bigger value a program gets.

All the values are described inside the *draw ()* function – a function that executes the strings of code and visualizes the result until the application is stopped:

```
int ix = 0;
int iy = 0;
int offset = ix + iy*kinect.width;
int rawDepth = depth[offset];
PVector v = depthToWorld(ix, iy, rawDepth);
```

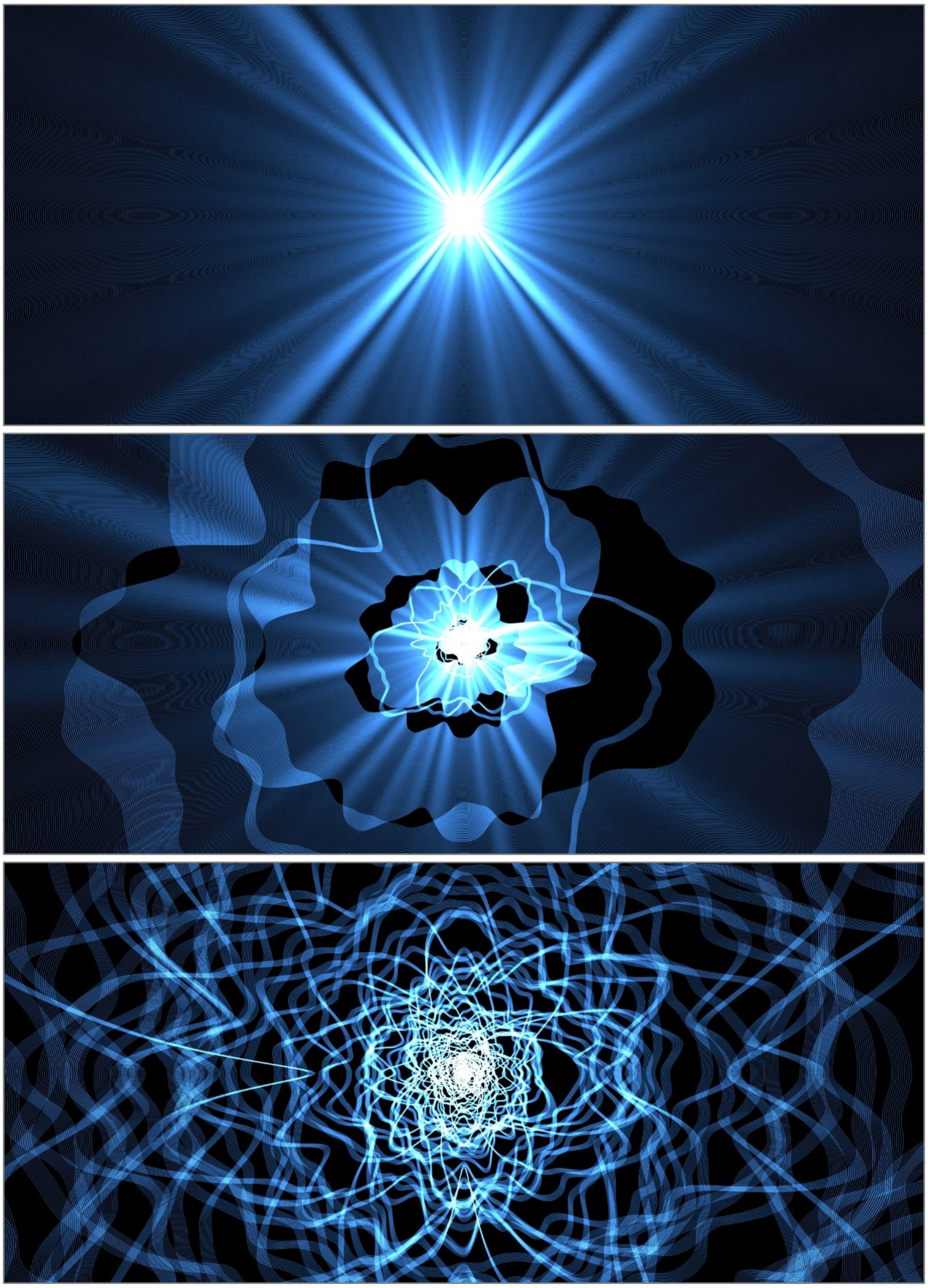


Figure 35. Processing sketch: “Deep blue”

The value of ring's increment is set manually at the beginning of the program and can vary from sketch to sketch; this way the same movements of a visitor could result into different visual effects.

4.9 The final project

Basic requirements for the interactive installation include:

- a laptop/personal computer;
- a program exported from the Processing environment as an .exe application;
- Kinect sensing device;
- a video projector;
- audio system;
- a darkened room.

The final installation will have a set of minimum 5 sketches or visual effects that differ in shapes and colors that are being projected on a wall. Any visitor could interact with the visual effects by changing his position in a room. The installation will be supported by an audio material. More examples of a visual render can be found in the Appendix P I.

4.10 Future research

There is a space for improvement within the frames of any work. The potential development of this project can be analyzed by a number of content areas.

The code part. It is always a good idea to experiment with code by adding more commands and functions in order to create more complicated figures. The amount of techniques and strictures applied to a program is limited only by coder's imagination. One example is to experiment with adding more randomness to some values, which will result in each render being more unpredictable and unique.

The software and programming environments. All the code part of this project was executed with the Processing language. Different languages bring different opportunities; future work might include programming with openFrameworks and Cinder.

The technical part. Scaling the used technique up could lead to more visually attractive results. Microsoft Kinect is a relatively old technology; its big advantage is availability and the ease of use. The disadvantage is that the image resolutions stay pretty low. Solving this problem might be one of the future steps of improvement.

Audio material. The process of interactivity can be expanded much further if it involves an audio material – some music/sound/noise that reacts and changes depending on visitor's movements.

Deeper analysis of all these points and a search for possible solutions create a base for a future study.

CONCLUSION

This project is my attempt to investigate the possibilities of creative coding in a world of a visual media; in other words – to create art with programming. The theoretical analysis includes the history of creative coding, most common techniques, tools of its creation. An original interactive installation was designed based on this research.

The final project is written in the Processing programming language; it represents a series of changing and growing images that are projected on a wall. A motion detection system captures the presence of any visitors, calculates their position in a room and transfers this data into a set of specific values that influence the movement of the original images.

All the goals of the project have been reached, including a development of a unique set of code sketches with a shape and color variability, combination of these sketches into a single final program, application of a principal of interactivity.

The result is more than satisfying. During this study I've learned a great number of new programming techniques, solved some technical challenges, found a lot of inspirational material. I'm planning to continue developing my skills of creative coding and apply them to new projects.

BIBLIOGRAPHY

1. LOPES, Dominic. *A philosophy of computer art.* New York: Routledge, 2010. ISBN isbn0415547628. p 1.
2. GOLDSTEIN, E. Bruce. *Encyclopedia of perception*. Los Angeles: SAGE, c2010. ISBN isbn978-1-4129-4081-8. p. 492.
3. MITCHELL, M., BOWN, O. *Towards a Creativity Support Tool in Processing: Understanding the Needs of Creative Coders.* 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration. New York City, New York: Association for Computing Machinery, 2013 (ACM).
4. GALANTER, Philip. *What is Generative Art? Complexity theory as a context for art theory*. GA2003 - 6th International Conference on Generative Art, 2003.
5. LEVIN, Golan. *Interview by Carlo Zanni for CIAC Magazine.* Montreal: Centre Art Contemporaine, 2004.
6. DOEHRING, James. *What is interactive art?* Conjecture Corporation, 2018: http://www.wisegeek.com/what-is-interactive-art.htm.
7. PEARSON, Matt. *Generative art: a practical guide using processing.* Shelter Island, NY: Manning, c2011. ISBN 978-1-935182-62-7, p. xxii - xxiii.
8. REAS, Casey., FRY, Ben. *Processing: a programming handbook for visual designers and artists.* Cambridge, Mass.: MIT Press, c2007. ISBN 0-262-18262-9, p. xxiii – xxiv, p. 1-2.
9. DAVIDE. *Spaghetti Coder: Generative Art and Mathematics: the interview with Toni Mitjanit.* November 17, 2017: http://www.mathisintheair.com/eng/2017/11/17/spaghetti-coder-generative-art-and-mathematics/
10. GREENBERG, Ira. *Processing: creative coding and computational art.* New York: Distributed to the book trade worldwide by Springer-Verlag, c2007. ISBN 159059617X. p. 13, 19- 20, 31.
11. GRIMES, William. *Harold Cohen, a Pioneer of Computer-Generated Art, Dies at 87*. The New York Times*, May 6, 2016:* https://www.nytimes.com/2016/05/07/arts/design/harold-cohen-a-pioneer-of-computer-generated-art-dies-at-87.html
12. GARCIA, Chris. *Harold Cohen and AARON—A 40-Year Collaboration. Computer history museum, August 23, 2016:* http://www.computerhistory.org/atchm/harold-cohen-and-aaron-a-40-year-collaboration/
13. VEROSTKO, Romain. *Artist profile.* DAM-Gallery: http://www.dam-gallery.de/index.php?id=49&L=1

14. RAES, Casey. *Information. A database for Casey REAS, 28* November 2015: http://reas.com/information

15. BROOKS, F. P., HOPKINS, A.L., NEUMANN, P.G., WRIGHT, W.V. *An Experiment in Musical Composition*. Reprinted from IRE TRANSACTIONS ON ELECTRONIC COMPUTERS Volume EC-6, Number 3. USA, September, 1957.

16. LIVEPROGRAMMING. *A history of Live programming.* Live Prog Blog, January 13, 2013. http://liveprogramming.github.io/liveblog/2013/01/a-history-of-live-programming/

17. TAYLOR, Stuart., IZADI, Shahram., KIRK, David., HARPER, Richard,. GARCIA-MENDOZA, Armando. *Turning the Tables: An Interactive Surface for VJing.* Microsoft Research Cambridge, J J Thomson Avenue, Cambridge, UK.

18. LEE, Jaewoo., KIM, Yeonjin, HEO, Myeong-Hyeo, KIM, Dongho. *Real-Time Projection-Based Augmented Reality System for Dynamic Objects in the Performing Arts.* Symmetry 2015, 7, 182-192; doi:10.3390/sym7010182

19. THE CREATORS PROJECT. *Into a flock of birds in a gorgeous interactive video.* The Atlantic, June 14, 2012: https://www.theatlantic.com/video/archive/2012/06/grow-wings-or-dissolve-into-a-flock-of-birds-in-a-gorgeous-interactive-video/467643/

20. FRY, Ben, REAS, Casey. *Overview. A short introduction to the Processing software and projects from the community.* Processing, 2018: https://processing.org/overview/

21. OPENPROCESSING. *OpenProcessing Community Guidline*. OpenProcessing, 2018: https://www.openprocessing.org

22. OPENFRAMEWORKS. OpenFrameworks, a guidline. Rackspace, April 8, 2018: http://openframeworks.cc/about/

23. EAKIN, Rich, HOUX, Paul. *Cinder, about.* Cinder, 2017: https://libcinder.org/about

24. RATHOD, Pinky, PATEL, Avani. "*A Novel Approach for Moving Object Detection from Dynamic Background*." Int. Journal of Engineering Research and Applications www.ijera.com ISSN : 2248-9622, Vol. 5, Issue 3, (Part -2) March 2015, p.71-74

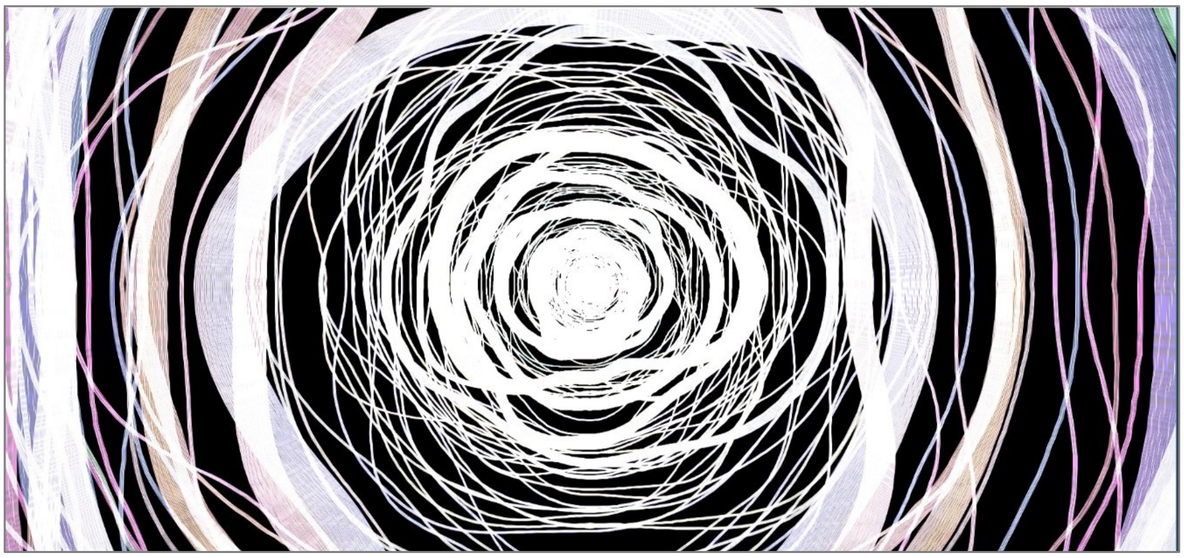
25. Wilson, Mark, Buchanan, Matt. *"Testing Project Natal: We Touched the Intangible".* Gizmodo. Gawker Media. Retrieved June 6, 2009.

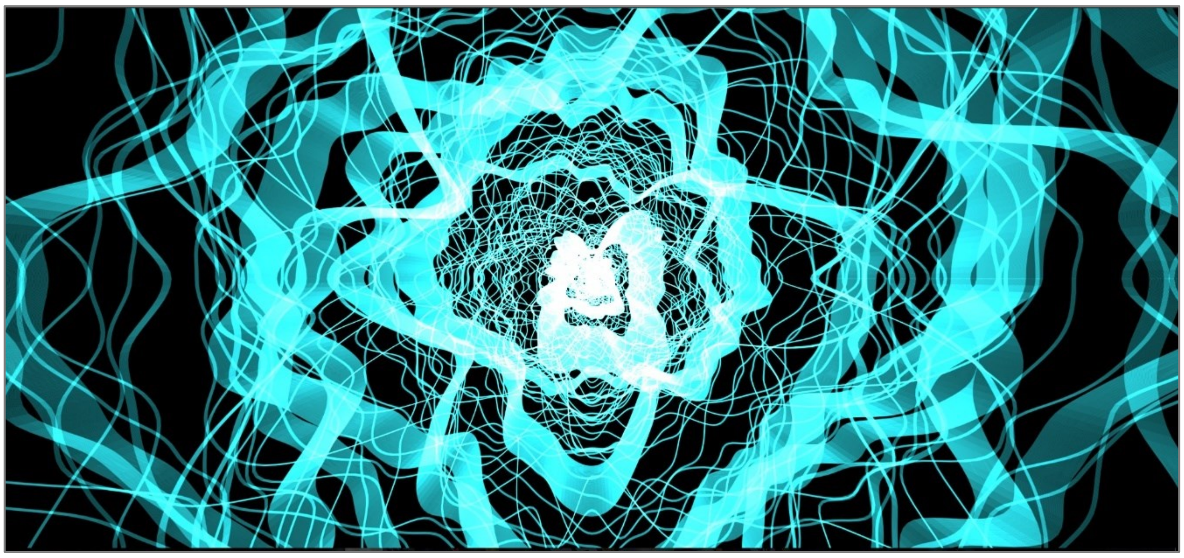
26. TERDIMAN, Daniel. *Leap Motion: 3D hands-free motion control, unbound*. C/Net: sci-tech, May 20, 2012: https://www.cnet.com/news/leap-motion-3d-hands-free-motion-control-unbound/

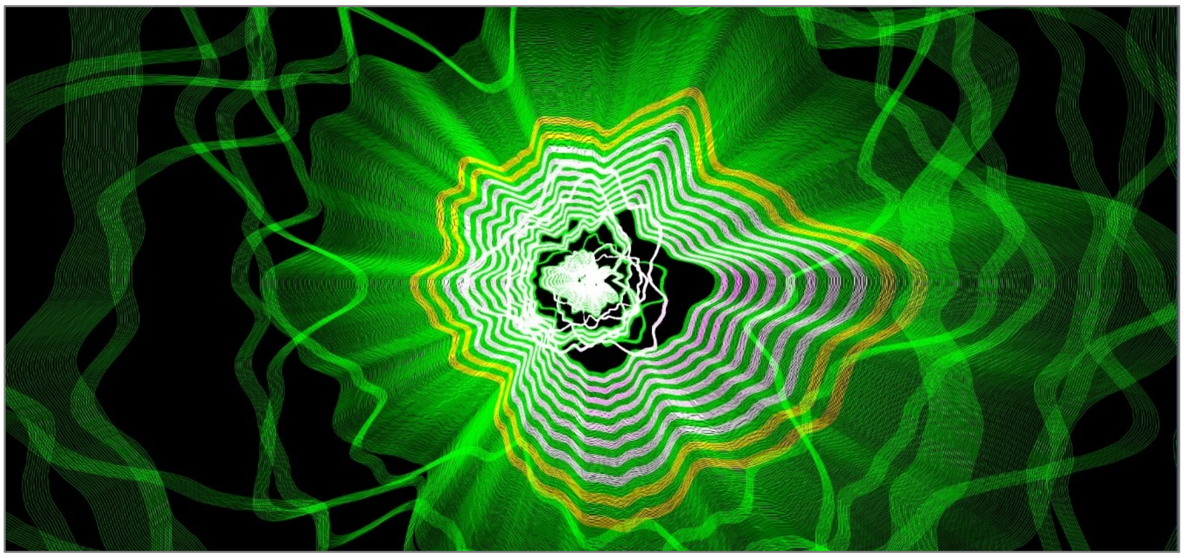
LIST OF FIGURES

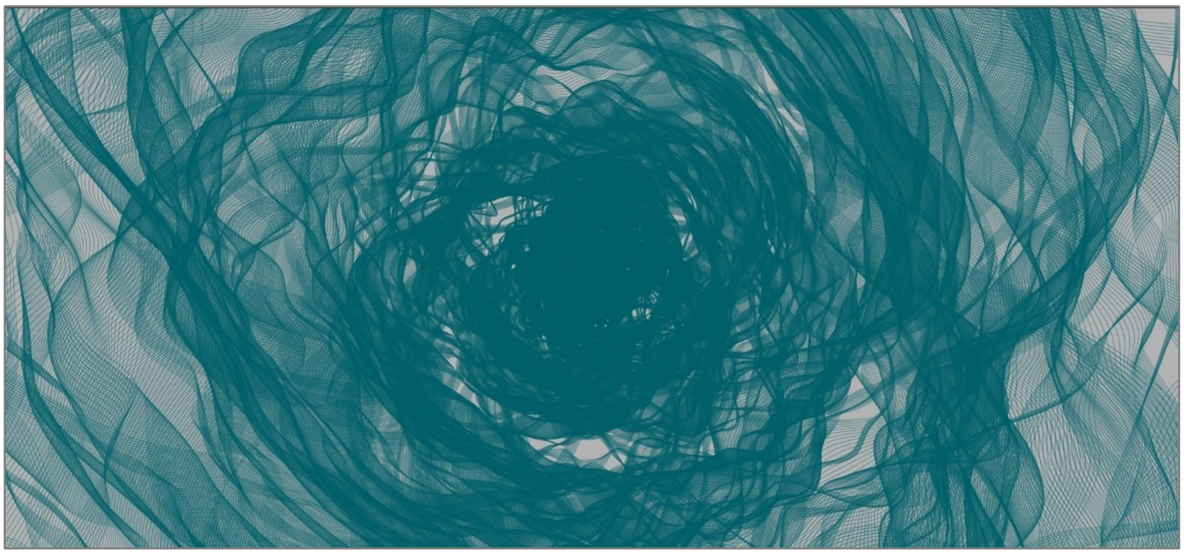
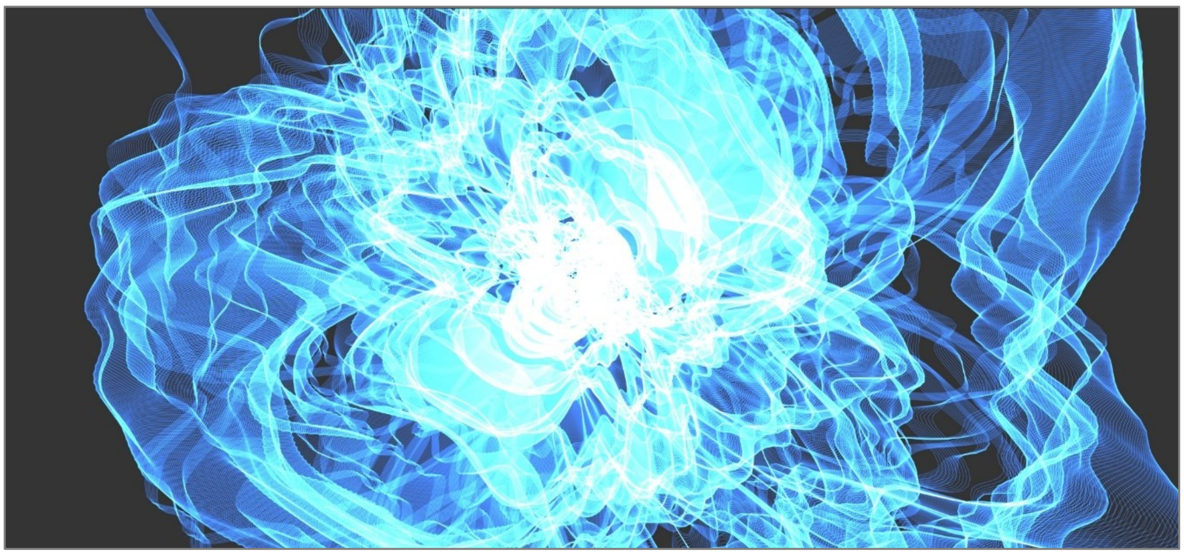
APPENDICES

APPENDIX P I: EXAMPLES OF RENDERING







APPENDIX P II: ENCLOSED CD

The enclosed CD contains:

- this work in PDF and DOC formats;
- image documentation of the project part.