BRAIN. Broad Research in Artificial Intelligence and Neuroscience
ISSN: 2068-0473 | e-ISSN: 2067-3957
Covered in: Web of Science (WOS); PubMed.gov; IndexCopernicus; The Linguist List; Google Academic; Ulrichs; getCITED; Genamics JournalSeek; J-Gate; SHERPA/RoMEO; Dayang Journal System; Public Knowledge Project; BIUM; NewJour; ArticleReach Direct; Link+; CSB; CiteSeerX; Socolar; KVK; WorldCat; CrossRef; Ideas RePeC; Econpapers; Socionet.

2023, Volume 14, Issue 2, pages: 115-127 | https://doi.org/10.18662/brain/14.2/447
Submitted: June 6th, 2023 | *Accepted for publication:* June 28th, 2023

Creative Coding as a Modern Art Tool

Asmati CHIBALASHVILI [1],
Igor SAVCHUK [2]
Svitlana OLIANINA [3]
Ihor SHALINSKYI [4]
Yuriy KORENYUK [5]

[1] Candidate of Art Studies, Senior Researcher, Deputy Director for Scientific Affairs, Modern Art Research Institute of the National Academy of Arts of Ukraine, ORCID ID: 0000-0003-1001-4273, e-mail: chibalashvili@mari.kiev.ua
[2] Doctor of Art Studies, Professor, Director, Modern Art Research, Institute of the National Academy of Arts of Ukraine, ORCID ID: 0000-0002-6882-3404, e-mail: savchuk@mari.kiev.ua
[3] Doctor of Art Studies, Associate Professor, Head of Department of Graphic Arts, National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", ORCID ID: 0000-0002-0628-146X, e-mail: s.olianina@ukr.net
[4] Candidate of Art Studies, Department of Graphic Arts, National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", ORCID ID: 0000-0001-5366-5979, e-mail: amuigor@gmail.com
[5] Candidate of Art Studies, Associate Professor, Department of Graphic Arts, National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", ORCID ID: 0000-0002-7766-6842, e-mail: yu.korenuyk@gmail.com

Abstract: *The research examines the phenomenon of creative programming as an example of the interaction between art and technology. The historical prerequisites for its emergence and various tools for its implementation are analysed. Systematization and analysis of existing platforms and computer languages for creative programming made it possible to classify them chronologically, according to principles of work, and technical characteristics. The main ways of using creative programming in different periods were explored in analysing contemporary art projects that used creative programming. Thus, The Algobabez duo used real-time programming with SuperCollider software at their performances. The capabilities of MAX/MSP made it possible to implement the performer-computer interaction in Inter-sax-tive by composer Nicolas Scherzinger. In the interactive project Messa di Voce, the sound is transformed with the help of the Openframeworks software into graphics, which affects the performers. In other cases, the software was written specifically for an art project. In the Abraham project, artist and programmer Gene Kogan tries to create an autonomous artificial artist who has his own will and can create authentic visual images. As part of the multidisciplinary interactive project "Hakanai", we can see how code written specifically for the project, allows performer to interact with a visual projection (mapping) that, in turn, interacts with the performer's moves. We prove that creative programming has become a promising, independent tool of creativity within modern interdisciplinary artistic practices, borders of which continue to expand thanks to the open code approach and the involvement of a community of like-minded people.*

Keywords: *digital art, algorithm, sound, image, coding, code, generic, interactivity.*

How to cite: Chibalashvili, A., Savchuk, I., Olianina, S., Shalinskyi, I., & Korenyuk, Y. (2023). Creative Coding as a Modern Art Tool. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience, 14*(2), 115-127. https://doi.org/10.18662/brain/14.2/447

Introduction

Technology has always played an essential role in the development of art. Artists explored new inventions and techniques and used them as tools for their art, expanding the means of expression and upgrading work methods.

However, in recent decades, this interaction has acquired new dynamics. As we can see, the trend toward combining science, technology, engineering, art and mathematics is being actively implemented through the concept of STEAM education. It proves that the combination of technical knowledge and the development of creativity at the initial stages of the growth of a mind is an important trend in education.

The combination of logical thinking and creative freedom is essential to the rise and growth of creative programming as an artistic field. In this context, the position of the Processing programming environment (one of the most popular creative programming tools) developers is indicative:

> "We believe in the synthesis of the arts and technology, and we know the arts are a necessary part of education from a young age. We don't want to live in a world where technology is developed without ideas and input from the arts and where only some people have access to learning to code" (Processing Foundation, 2018).

Creative programming emerged at the interjunction of digital art and design on the one hand and the desire to find an intuitive approach to the programming process on the other. It aims to implement an artistic idea by creating visual, sound images or complex systems of interaction with the help of software code in interdisciplinary art projects. That distinguishes creative programming from functional programming, which objective is to solve specific applied tasks.

As part of creative programming, it is possible to work with various types of data: text, sound, image, video, 3D models, etc. With the help of software code, the artist can "translate" data from one medium to another. In other words, any data, even dry statistical information, can serve as material that will be processed and transformed into a specific type of media – sound, image, interactive projection, installation, etc. Thus, the ultimate goal of creative programming is to transform any data into an artistic project.

Analysis of recent research works and publications/ Literature Review

The vast majority of studies of creative programming belong to authors who, in their work, combined research in natural sciences with creative experiments. Thus, the theoretical basis of the research lies in the works of John Maeda, A. Michael Noll, Frieder Nake, Dominic Lopes and Christiane Paul.

In his book Design By Numbers, published in 1999, John Maeda considers the computer as an autonomous artistic environment, separate from the traditional tools of the artist. The purpose of the book was to create a new method of programming that would allow people without mathematical abilities to implement their creative ideas based on intuitive and visual approaches.

A similar view has A. Michael Noll, who, in his works, explored the aesthetics of computer art, called the computer a creative environment (1967) and believed that the artist should explore computer technologies and learn to program and use this knowledge (Noll, 1970; 2016).

The article "There should be no computer art", published in 1971 by Frieder Nake, caused a stir due to the perspective, which was significantly different from other researchers' of early computer art. In the article, the author recognized the invention of new interesting methods important for artists and a new communicative model of interaction between the creator and his creation but he does not see the point of using the computer "as a source of pictures for the galleries" instead believes that the computer can become a "convenient and important tool in the investigation of visual (and other) aesthetic phenomena as part of our daily experience" (Nake, 1971).

In his book "A Philosophy of Computer Art" (Lopes, 2009), exploring the aesthetics of computer art, Dominic Lopes considers technology as a tool of artistic creation, similarly to the views mentioned above. He indicates significant changes in artistic practice and, at the same time, talks about the need to renew approaches to creating and perceiving of art in the context of computer art.

Digital art researcher Christiane Paul considers its evolution through the latest information technologies and states that the terminology in this field needs to be clarified. She defines the art based on data, codes, and specifically designed software as "software art". In our opinion, the given definition fully reflects the essence of creative programming for which the code is the basis. Christiane Paul points out that "… code is sometimes called the artistic medium, the "paint and canvas" of the digital artist, but it does not fit into this metaphor because, among other things, it allows artists to create their tools —

if we extend the metaphor, we can say that in this case, the artistic medium allows the artist to create both a palette and a brush" (Paul, 2017).

Thus, researchers and practitioners of computer art interpret it as an autonomous artistic environment that functions separately from the traditional tools of the artist (John Maeda, A. Michael Noll), a tool for the study of aesthetic phenomena (F. Nake) or for the creating (D. Lopes) and a medium that allows the artist to create his instruments using the code (C. Paul). The given opinions illustrate the tendency towards the multidisciplinary character of creative activity in the context of creative programming.

Practical aspects of creative programming can be found in numerous manuals aimed at familiarizing the reader with the basics of computer technology and the possibilities of its involvement in solving creative tasks through mastering a specific programming language or programming environment.

The historical and theoretical context of creative programming

The first examples of creative programming were recorded in the early 1960s in the field of digital art. Among the first to use digital technologies in the visual arts were A. Michael Noll, Frieder Nake and George Nis. All three were engaged in natural sciences — - mathematics, physics, electrical engineering, and computer science — before launching into artistic experiments. Landmark exhibitions of these artists presenting computer-generated images took place in 1965 in Germany and became a reference point in the history of computer art. They based their works on mathematical computations, with the help of which the computer creates an image with varying degrees of control over the final result.

Vera Molnar (n.d.) has been creating visual art with the help of a computer since 1968. Getting a degree in fine arts and having an experience in programming, she began to create abstract geometric images with the help of algorithms, using Fortran and BASIC programming languages.

After the exhibition, Cybernetic Serendipity, which featured the works of the named artists, in 1968, the Computer Art Society and the computer art magazine PAGE were founded in London. Simultaneously with creative activity, these artists developed theories related to computer art and theorized their experiments.

Georg Nees identified his works as generative computer graphics in his dissertation and book "Generative Computergraphik" (Nees, 1969). In addition, the book includes research on aesthetics, semiotics, artificial intelligence, etc.

Later, in 1995, Jean-Pierre Hébert introduced the term "algorists" to define artists who use algorithms in their art. The development of digital technologies became a powerful impetus for the rapid development of algorithmic art, thanks to the possibilities that the computer could offer. And, first of all, the speed of computation, which was impossible to imagine in the "pre-computer" era.

Speaking about artists working with technology, Zhang, Y. and Funk, M. identify two approaches: "The first one is based on a concept, and with the second approach, the artist starts from the data or the available material, forming a concept based on the possibilities and the results of data processing" (Zhang & Funk, 2021). Researchers note that this division in real life is not so clear, and approaches can change at different stages of work. Despite the existing stereotype that creativity is based only on inspiration and the embodiment of various emotional states, the process of artistic creation has always contained a rational component. In this context, creative coding will serve as a tool for artists to expand the field for experimentation.

When an artist works with code, according to the researchers, "[they] may need a new way of thinking and working that helps them see this new field through the prism of the old field where they were active and professional" (Zhang, & Funk, 2021).

The earliest creative programming platforms discussed below were explicitly designed to help artists quickly master elementary programming.

The main platforms for creative coding

One of the early creative output-oriented programming environments is Design By Numbers (DBN). Its first version, DBN 1.0, was developed by John Maeda to introduce artists and designers to the basics of computational media.

The DBN system consists of three components: the core DBN software, the Design By Numbers book, and a website generator built for educators - Design By Numbers Courseware.

Later, a collective of scientists participated in the DBN system's development. Among them were Ben Fry and Casey Reas, who, in 2001 proposed a new, more developed system for teaching programming - Processing. Currently, Processing is the most popular software among visual artists. In addition, it is the software of choice when the task is to teach designers programming because of its accessibility and the ability to show the visual output of the entered code.

Like Design By Numbers by John Maeda, Processing is also multi-component and consists of:

- a programming language (which is a hybrid between the developers' original elements and the Java programming language),
- the Processing Development Environment, a programming interface for starting and stopping sketches and
- community (group of people sharing their sketches and code on related resources).

In our opinion, the latter enabled more significant learning and sharing opportunities, which is one of the prerequisites for its popularity.

> "The original mission of Processing was to create software that made learning to code accessible for visual people (designers, artists, architects) and to help a more technical audience work fluidly with graphics. We aspired to empower people to become software literate — to learn to read and write software. We wanted to change curriculum in universities and art schools around the world" (Processing Foundation, 2018).

In addition to educational activities, Processing has vast opportunities for creating multi-component art projects.

Below, we present a table with the main characteristics of the most widespread platforms, environments and languages of creative programming in chronological order.

Frameworks environments and languages for creative coding				
Name	Programming language	Capability	Principles of work	Date, developer
Max/MSP/Jitter	Object oriented programming language	Audio, video, multimedia. Audiovisual process in real time	Visual programming; MSP - an add-on for working with audio; Jitter - an add-on for working with video	Miller Puckette, 1980s. Cycling 1990-i pp.
https://cycling74.com/				
Pure Data	Event-oriented system of programing	Interactive multimedia projects	Visual programming	1996, Miller S. Puckette
http://puredata.info/				

SuperCollider	sclang – An interpreted programming language	Audio synthesis, algorithmic composition	Text programming	James McCartney, 1996
https://supercollider.github.io/				
vvvv	C#, HLSL shaders	Visual effects and graphics, audio and video in real-time	Visual programming; Ability to use external devices (i.e. Kinect and PSP); Interactive projects	vvvv group (Joreg, Max Wolf, Sebastian Gregor, Sebastian Oschatz)
https://vvvv.org				
Design By Numbers (DBN)	DBN Java support	An opportunity for designers, artists and other non-programmers to easily start computer programming	Text programming; Multiplatform (run from website or local computer)	John Maeda 1999
https://dbn.media.mit.edu/				
Processing	Java based programming language	Images, animation, interactive projects, web-applications, etc	Text programming; Open code	Shiffman, 2009
https://processing.org				
TouchDesigner	node based visual programming language, Python	Music, multimedia, interactive content in real-time, 2D, 3D, video-mapping, interaction of devices	Hybrid programming	2008 Derivative TouchDesigner 077, beta version
https://derivative.ca/				

OpenFrameworks	C++	Interactive multimedia app	Text programming	2009, Zach Lieberman, Theodore Watson, and Arturo Castro
https://openframeworks.cc/				
p5.js	Java script library	Images, animation, video, sound	Text programming; Browser-based	2014
https://p5js.org/				
cables.gl	Work with the cables modules kit	Interactive video-content in real time	Visual programming interface	2017, Undev Studio
https://cables.gl/				
Nannou	Rust	Graphics, audio	Visual programming	2017, MindBuffer
https://nannou.cc/				
OpenRNDR	Kotlin	Interactive projects, supports machine learning (ORML)	Hybrid programing; Open code; Uses shaders in its structure	2018, Edwin Jakobs, OpenRNDR Studio
https://openrndr.org/				

Analyzing the information presented in the table, we conclude that the reviewed platforms for creative programming can be divided into three types:

- text (DBN, Processing, P5.JS, OpenFrameworks, SuperCollider);
- visual (Cables, MAX/MSP/JITTER, Pure Data, vvvv , Nannou)
- hybrid (OPENRNDR, TouchDesigner), which combine the previous two.

These platforms for creative programming influence each other to some extent. For example, Design by Numbers is the predecessor of Processing, and p5.js is the browser counterpart of the latter. Cables.gl is inspired by vvvv, which is influenced by Max/Msp and Pure Data. In turn, Pure Data is a free analogue of Max/MSP. OpenFrameworks and Processing are similar because they serve as an interface to media libraries and hardware. It is important to note that it is possible to combine the

capabilities of several of those platforms within the framework of one project.

Over the past decades, software for creative coding has been dynamically developed due to the open-source philosophy, which states that each user can freely use the work of others and, in turn, supplement it with his/her ideas and code. Almost all developers of the platforms and programming environments mentioned above share and support this principle.

Implementations of creative coding in artistic practices

Two-dimensional fractal images and early computer music are the first examples of creative programming. Nowadays, it can be used in a far wider field. The result can be as traditional forms of artistic representation, as image or sound, or an interactive installation and live projection (mapping) programmed in real-time. Creative programming can interact with generative design and machine learning, which is based on algorithms.

Shelly Knotts develops projects using SuperCollider software as part of the Algobabez duo with Joanne Armitage (2018). In its performances, the duo uses various types of data and live programming, and improvisation plays a key role in their art. Thanks to the above, each of their performances is unique and full of spontaneous creative decisions that affect the final result of their performance.

Using the object-oriented software Max/MSP, composer Nicolas Scherzinger creates interactive musical works. Communication between the performer and the computer is carried out with the help of the software. "Inter-sax-tive" is a series of improvisation pieces for saxophone and interactive computer. The interaction between the performer and the computer takes place in real-time.

> "Each piece consists of a collection of sound materials and special effects with which the saxophonist freely improvises. The computer responds to the saxophonist by producing a series of sounds based on what the saxophonist plays. The computer can also create random elements throughout the piece, thus allowing the saxophonist to respond to the computer."
> (Scherzinger).

The computer generates sounds based on the algorithms of processing the sounds of the saxophone through the digital signal coming to it during the performance.

Rudolf Quintas' body-mapping audiovisual installation Black Hole aims to help blind people "experience the deep emotional qualities associated with their bodies through the creative exploration of movement through sound" (Quintas, n.d.). For this, the artist used the capabilities of MaxMsp, Ableton and Processing programs,

The audiovisual performance Messa di Voce (Italian for "placing the voice") was created with the involvement of openframeworks - a library of C++ tools for creative open-source programming.

> "The software transforms every vocal nuance into complex, differentiated and highly expressive graphics. These visual elements depict the singers' voices and serve as controls for their acoustic reproduction. While the voice graphics thus become an instrument for the singers to perform, the bodily manipulations of these graphics further reproduce the sounds of the singers' voices, creating a loop of interaction that fully integrates the performers into the sonic atmosphere, virtual objects and real-time processing" (Lieberman, n.d.).

The interactive video mapping was used in the project "Hakanaï. A performance for a dancer in a cube of moving images. Choreography, that depicts the transience of dreams and the impermanence of things" (Hakanaï). The ever-changing image, which, according to the concept, symbolises a dream, reacts to the dancer's movements and changes accordingly. An image is projected into a cube with a dancer inside, creating a multidimensional space. The authors of the installation and its artistic directors are Claire Bardent and Adrian Monda (2013). It should be noted that more than twenty people are involved in the technical implementation of the performance. In their projects, the company Adrien M. & Claire B. combines choreography with visual art, using digital devices and original code for their interaction with the main character of the performance (n.d.).

Gene Kogan, artist and programmer, combines generative art, collective intelligence and informatics in his projects. He strives to develop scientific literacy through creativity, and to achieve this, he publishes his works, lectures and training manuals free to access. His Abraham project goes beyond multimedia or interactive projects – it aims to create an autonomous artificial artist. "By autonomous, we mean that autonomous

artificial artist demonstrates its freedom of action or will independent of the will of its creators. Even the most advanced artificial intelligence, which lacks autonomy, is no more creative than a dummy in the hand of a ventriloquist" (Kogan, 2021). So, his autonomous artificial artist thanks to the author's involvement of the collective intelligence in the process of machine learning, capable of creating original art.

Conclusions

The phenomenon of creative coding establishes the close link between art and technology, which in recent years has been actively used in real and virtual art spaces. The given examples present various methods of involving creative programming as a new creative tool in the arsenal of artists. Real-time programming using the SuperCollider software is used in the performances of the Algobabez duo. The capabilities of MAX/MSP made it possible to realize the interaction between the performer and the computer within in "Inter-sax-tive" project by the composer Nicolas Scherzinger (n.d.). In the interactive project Messa di Voce, with the help of the Openframeworks program, the sound is transformed into graphics, which, in turn, affects the performers. We see that in some cases, software was developed specifically for a specific art project. Thus, in the Abraham project, artist and programmer Gene Kogan tries to create an autonomous artificial artist who has his own will and is capable of autonomously creating original visual images. Within the framework of the multidisciplinary interactive project "Hakanai", we also see a unique code designed to use the performer's interaction with a visual projection (mapping) that, in turn, interacts with his movements.

Creative programming as a part of contemporary artists' toolkit is still in dynamic development. At the same time, it is evident that the discovery of a broader spectrum of its artistic capabilities still lies ahead. However, it is evident that some art projects get ever more complex, and their successful implementation was possible amid coding. If at the early stages of creative programming, two-dimensional, fractal images were created or variants of sound combinations were computed in algorithmic composition, nowadays, complex interactive multimedia projects are created using machine learning.

The philosophy of open source and the presence of a strong network of community support is crucial for the spread and development of creative programming in the future. At the same time, the prospects of future implementation in artistic practices are related to the renewal of

creative thinking through combining the artist's research activity and mastery of the latest technologies.

References

Cables. (2022). *About.* Cables. https://cables.gl/

Cycling '74. (1998). *About*. Cycling '74. https://cycling74.com/

Design By Numbers. (1999). *About*. Design by numbers. https://dbn.media.mit.edu/

Jakobs, E. (2018). *Playful forprototypes. serious forproduction.* OPENRNDR. https://openrndr.org/

Kogan, G. (2021, December 10). *Artist in the Cloud - Gene Kogan*. Medium. https://medium.com/@genekogan/artist-in-the-cloud-8384824a75c7

Lieberman, Z. (n.d.). *Thesystemis | the works of zach lieberman and collaborators*. Thesystemis. http://thesystemis.com/projects/

Lopes, D. (2010). *A philosophy of computer art*. Routledge.

Maeda, J. (1999). *Design By Numbers*. Maeda Studio. https://maedastudio.com/1999/dbn/index.php

Max/MSP/Jitter. (n.d.). *See this sound.* Max/MSP/Jitter. http://www.see-this-sound.at/works/797.html

McCartney, J. (1996). *What Is SuperCollider?* Solvusoft. https://www.solvusoft.com/en/file-extensions/software/james-mccartney/supercollider/

Miller Puckette. (2013). *About.* MSP. http://msp.ucsd.edu/

MindBuffer. (2017). *Nannou.* MindBuffer. https://mindbuffer.net/nannou

Monda, A., & Bardent, C. (2013). *Hakanaï.* Adrien, M. & Claire, B. http://www.am-cb.net/projets/hakanai

Monda, A., & Bardent, C. (n.d.). *About.* Adrien, M. & Claire, B. https://www.am-cb.net/en/a-propos

Nake, F. (1971). There Should Be No Computer-Art. *Bulletin of the Computer Arts Society, London, 18*(1–2), 18–19.

Nannou. (2018). *Home*. Nannou. https://nannou.cc/

Nees, G. (1969). *Generative Computergraphik*. Siemens.

Noll, A. M. (1967). The digital computer as a creative medium. *IEEE Spectrum, 4*(10), 89–95. https://doi.org/10.1109/mspec.1967.5217127

Noll, A. M. (1970). Art Ex Machina. *IEEE Student Journal, 8*(4), 10–14.

Noll, A. M. (2016). Early Digital Computer Art at Bell Telephone Laboratories, Incorporated. *Leonardo, 49*(1), 55–65. https://doi.org/10.1162/leon_a_00830

OpenFrameworks. (2007). *About*. OpenFrameworks. https://openframeworks.cc/

OPENRNDR. (2022). *About*. OPENRNDR. https://openrndr.org/about/

P5js. (2017). *Hello!* Home. P5js. https://p5js.org/

Paul, C. (2017). *Цифровое искусство* [Difital Art]. Ad Marginem.

Processing Foundation. (2018, June 21). *A Modern Prometheus - Processing Foundation*. Medium. https://medium.com/processing-foundation/a-modern-prometheus-59aed94abe85

Processing. (2024). *Welcome to processing!* Processing. https://processing.org/

Pure Data. (2004). *PD community site*. Pure Data - Pd Community Site. http://puredata.info/

Quintas, R. (n.d.). *Studio Rudolfo Quintas*. Studio Rudolfo Quintas. https://www.rudolfoquintas.com/

Scherzinger, N. (n.d.). *ScherziMusic*. ScherziMusic. http://www.scherzimusic.com/intersax.html

Shiffman, D. (2009, August 23). *Interview with Casey Reas and Ben Fry*. Rhizome. http://rhizome.org/editorial/2009/sep/23/interview-with-casey-reas-and-ben-fry/

SuperCollider. (2014). *Index*. SuperCollider. https://supercollider.github.io/

TouchDessigner. (2008). *TouchDessigner Features*. TouchDessigner. https://derivative.ca/ Knotts, S., & Armitage, J. (2018). ALGOBABEZ: Writing code, Pushing Buttons. *Dancecult, 10*(1). https://doi.org/10.12801/1947-5403.2018.10.01.13

Undev Studio. (2017). *About*. Undev Studio. https://undev.studio/

Vera Molnar. (n.d.). *About*. Vera Molnar. http://www.veramolnar.com/

vvvv. org. (2006). *A multipurpose toolkit*. vvvv. org. https://vvvv.org/

Zhang, Y., & Funk, M. (2021). *Coding Art: The Four Steps to Creative Programming with the Processing Language (Design Thinking)* (1st ed.). Apress. https://doi.org/10.1007/978-1-4842-6264-1