УДК 39729/11

VISUAL PROGRAMMING AND ARTISTIC THINKING IN CONTEMPORARY MEDIA ART

Bohdan TYMOFIIENKO – PhD student, Modern Art Research Institute of the National Academy of Arts of Ukraine, Kyiv
https://orcid.org/0000-0003-1146-1948
https://doi.org/10.35619/ucpmk.51.1047
bogdan.tym@gmail.com

This article examines visual programming as a distinctive form of artistic thinking that has evolved at the intersection of digital technologies, conceptual art, and contemporary media practices. The rapid development of environments such as Processing, Max/MSP, TouchDesigner, and p 5.js has contributed to the emergence of new models of authorship, interactivity, and generativity, in which the algorithm operates not merely as a technical tool but as an integral component of artistic intentionality. The theoretical basis of the study draws on the works of John Maeda, A. Michael Noll, Frieder Nake, Christiane Paul, and Dominic Lopes, whose research provides philosophical, aesthetic, and methodological foundations for understanding software-based art. The article offers a detailed analysis of emblematic artistic projects-including Casey Reas's *MicroImage*, LIA's *Tissue*, James George and Jonathan Minard's *CLOUDS*, and Jer Thorp's *Just Landed*-highlighting how code structures visual processes, enables interaction, and generates procedural variability. The findings demonstrate that visual programming constitutes an autonomous artistic method grounded in algorithmic logic, systemic thinking, and dynamic forms of representation. It establishes a conceptual and aesthetic framework in which digital materiality, procedural authorship, and generative structures become central to artistic practice in the 21 st century.

Key words: visual programming, code, generative art, digital aesthetics, algorithmic creativity, artistic thinking.

Problem statement. The rapid development and spread of digital technologies have led to a profound transformation of artistic thinking. Artists are increasingly turning to programming as a means of expressing their author's concept. In this context, visual programming appears as a powerful and accessible tool that combines the aesthetic and the technical, opening up new forms of authorship, interactivity, and generativity. Problem statement. The rapid development and spread of digital technologies have led to a profound transformation of artistic thinking. Artists are increasingly turning to programming as a means of expressing their author's concept. In this context, visual programming appears as a powerful and accessible tool that combines the aesthetic and the technical, opening up new forms of authorship, interactivity, and generativity. In the broader context of the digital revolution, code, as Bilal Shahid aptly formulates it, «has emerged as a new kind of brush» [11; p. 5], radically expanding the field of artistic operations beyond traditional media.

Software environments focused on creating interactive audiovisual products, in particular Max/MSP, TouchDesigner, and Processing, are increasingly used in artistic practice. Thanks to their capabilities, artists are able to create multimedia interactive projects. As a result, a new language of art is being formed, where the algorithm becomes not only a technological basis but also an essential part of artistic expression. Both scientists and practitioners widely discuss such implementation of code in artistic creativity. John Maeda (1999) saw the computer as an autonomous creative medium that could facilitate intuitive and visual artistic expression even for those without technical training. Similarly, A. Michael Noll emphasized the need for artists to master computer technologies, considering them an organic component of the modern arsenal of artistic tools [5; pp. 10–14]. Christiana Paul, in turn, argued that code itself becomes a kind of artistic medium – «not just paint and canvas», but a system for creating tools of expression [8; p.118]. The relevance of this study is due to the growing role of creative programming in both professional artistic practice and art education. In addition, the democratization of visual programming tools has contributed to the active use of code in the creative process, allowing a new generation of artists to use algorithms to rethink databases of various origins as components of an artistic work.

The article aims to explore visual programming as a form of artistic thinking, based on the analysis of specific artistic projects. By studying the historical evolution of visual programming and considering interdisciplinary practices, the author seeks to highlight how artists use software environments to combine algorithmic logic with aesthetic expression, rethinking the boundaries between code, art and interaction.

The methodological tools of the article are based on a combination of theoretical analysis and the study of practical examples, which allows for a comprehensive study of visual programming as a form of artistic thinking. The work uses a review of scientific literature-both peer-reviewed academic publications and historical sources devoted to the aesthetic, philosophical and technological understanding of software code in art. Particular emphasis is placed on the works of John Maeda, Frieder Nake, Christiana Paul, and Dominic McIver Lopes. The practical section uses a historical and comparative approach to the analysis of visual programming tools, in particular Processing, as well as a critical examination of specific artistic projects in in which programming is a key component of the creative process.

Literature review. The theoretical foundations of the study of visual programming in art were formed at the intersection of digital aesthetics, computer science, and art history research. Both scholars and artists have long been engaged in discussions about the consequences of using code and system logic in creativity, in particular about changing traditional ideas about the artistic process, authorship, the materiality of a work of art, and the problem of its aesthetic control.

One of the important voices in this discussion belongs to John Maeda, whose project Design By Numbers (1999) became the basis for the further development of accessible creative programming. Maeda saw the computer as an autonomous artistic medium, suitable for those without mathematical or technical training but who aspired to expressive visual thinking. His educational model, which combined visual experimentation with minimalist coding syntax, directly inspired the creation of a programming environment such as Processing. This is confirmed by his view of the goal of DBN, which he formulated as the desire to «… create a programming environment that would introduce visually oriented people to computational expression» [4; p. 88].

A. Michael Noll, one of the pioneers of computer art, spoke in a similar vein, and back in the 1960 s he saw the digital computer as a full-fledged artistic medium, equivalent to a paintbrush or a chisel. In his classic article The Digital Computer as a Creative Medium (1967), he argued that the structure of a computer program has much in common with the logic of constructing a painting or sculpture, and therefore can serve as material for artistic thinking: «The logical structure of a computer program is similar to the logical structure of a painting or sculpture» [7;p. 93]. According to Noll, to fully master this new environment, the artist must not only use ready-made software but also learn the basics of programming. «The artist must learn to use the computer and program it to fully utilize this medium» [7: p. 89].

This line of thought converges with more recent attempts to conceptualize algorithms as a structural basis of artistic processes. Yanai Toister, analysing photography through the lens of algorithmic art, defines algorithms as «means of making a situation explicit» [15; p. 224], stressing that strict procedural instructions can expand, rather than limit, aesthetic possibilities. Referring to Frieder Nake, he also recalls the definition of algorithms as «finite descriptions of infinite steps» [15; p. 224], which is highly relevant to generative and software-based art, where finite code can generate hypothetically unlimited series of visual results.

Frieder Nake, one of the leading theorists of computer art, criticized the commercialization of the phenomenon in his essay There Should Be No Computer Art (1971) and urged against superficial copying of gallery models. He believed that the value of the computer in art lies not in creating objects for the market, but in the exploration of visual and aesthetic phenomena, noting, «I do not see the task of the computer as a source of images for galleries. But I see the task of the computer as an important tool in the exploration of visual […] aesthetic phenomena as part of everyday experience» [5; p. 19]. Dominic McIver Lopes, in his monograph A Philosophy of Computer Art, outlines computer art as a special art form, defined by its procedural nature and interactivity. He argues that the computer in art acts not as a neutral tool, but as a procedural agent, capable of changing the representation of the work in response to the user's actions. Lopes considers the user interface, the variability of display, and the involvement of algorithms as the basis for rethinking traditional criteria for aesthetic evaluation. In his interpretation, code is not only a means of expression, but also a component of artistic intention, requiring new forms of interaction between the work, the artist, and the viewer [3].

Christiana Paul, one of the leading contemporary researchers of digital art, defines software art as a key subgenre of media art, where code appears not simply as a technical tool, but as the material of artistic thinking. According to her, «artists write code not only to create a palette and a brush, but to create a system capable of generating means of expression» [8; p. 118]. Artistic code, in her opinion, encompasses not only functionality, but also conceptual construction, logic of interaction, and language of expression.

The aforementioned theoretical approaches are supported by modern academic generalizations, in particular the article Creative Coding as a Modern Art Tool by Chibalashvili et al. (2023). In it, visual programming is considered as a component of an interdisciplinary trajectory that encompasses design, music, performance and installation art. The authors argue that creative programming is an independent artistic medium, formed under the influence of open source culture, collective intelligence and hybridization of media forms. «Creative programming», they note, «aims to realize an artistic idea by creating visual, sound images or complex interaction systems using software code» [1; p. 116].

Additionally, Zhang and Funk (2021) in Coding Art: The Four Steps to Creative Programming argue that the rise of creative coding reflects a shift towards an exploratory logic in art, where iteration, modularity, and systemic feedback replace traditional linear composition. The authors argue that contemporary artists must adapt to «a new way of thinking and working» that is emerging in interaction with visual programming environments [16; p. 33].

A review of scholarly approaches to visual programming in art demonstrates the gradual transformation of computer code from a technical tool to a fully-fledged means of artistic thought.

From the early initiatives of John Maeda and A. Michael Noll, who emphasized the accessibility and materiality of code as an artistic medium, to the critical analysis of Frieder Nake, who saw the value of the computer primarily in its research potential, a shift in focus from the result to the process can be traced. Dominic McIver Lopes and Christiana Paul have proposed philosophical and aesthetic principles according to which code appears as a procedural structure that forms new models of interaction between the work, the author and the viewer. Modern research [1; 16] considers creative programming as an interdisciplinary practice characterized by iterativeness, generativity and an open intellectual environment. In this context, Toister's algorithmic perspective complements the picture by linking software-based practices to a broader trajectory of «programming the beautiful» in information aesthetics [15]. Thus, visual programming acquires the status of an autonomous artistic method that changes the notion of authorship, aesthetics and digital materiality.

Presentation of the main material. In contemporary art, visual programming is increasingly emerging not as a tool, but as a form of thinking that combines logic, aesthetics, and procedurality. In this context, the artistic act is defined not only by the result, but primarily by the algorithmic construction of the system that generates this result. As Dominic McIver Lopes (2010) noted, software art is procedural in nature: the code does not simply implement the idea, but forms it as a structure open to interaction, change, and ambiguity. This approach is close to the traditions of conceptual art, in which the instruction, not the object, is paramount. With the advent of graphical environments-such as Max/MSP, TouchDesigner, and hybrid platforms like p 5.js – visual programming has acquired interface visibility. Instead of abstract code, the artist operates with objects, connections, blocks – compositional units that are both logical and visual. The interface becomes part of the creative process: it not only helps to realize the idea, but also stimulates new artistic solutions. This principle is clearly manifested in real time – when the result is updated immediately after a change in a patch or line of code – and allows you to combine experimentation with precision. In terms of Toister's framework, such environments can be considered apparatuses whose programmed instructions materialize aesthetic decisions and make their inner logic explicit [15].

The key feature of this approach is the change in the relationship between the artist and technology. Instead of using programs as tools, the artist enters into a co-creative dialogue with the system, where the initiative is shared between man and machine. As shown by the research of Chibalashvili et al. (2023), creative programming creates a space in which data, sounds and visual elements circulate between the author and the algorithm, forming a closed feedback loop. In the case of generative or autonomous systems, such as Gene Kogan's projects, even the decision-making process itself is delegated to the machine – the code begins to «think» within the framework of a given artistic logic. In this approach, the interface is not just a technical environment, but a space for experimental design, where the artist thinks structurally rather than figuratively. As Zhang and Funk (2021) point out, visual programming shapes a new way of thinking in which iterativeness, modularity, and continuous testing play key roles. The artist does not work with a final form, but with a dynamic system that reacts, adapts, and evolves through interaction. It is this openness – to error, to change, to revision – that makes visual programming a powerful method of artistic exploration.

Among the most influential early works created in Processing, MicroImage (2002) by Casey Reas, the co-founder of the tool, holds a special place. The project is a paradigmatic example of generative art, in which code appears not only as a tool, but also as a material for aesthetic construction. MicroImage models an ecosystem where thousands of autonomous agents interact in a minimal environment according to given behavioral rules. Each agent is just a point or line that follows the trajectory of its movement, but together they form a complex dynamic structure that is constantly changing. As Reas himself emphasizes, «software is art», and images are just traces of his work [10; p. 17]. At the heart of the work is the Process Compendium model, which describes typical elements of interaction: growth, accumulation, dispersion, disappearance. Despite the simplicity of the individual algorithms, their interaction generates complex behavior – an example of the so-called emergent form. Reas emphasizes the procedural origin of these visual effects: «I am inspired by natural systems that change over time. I translate these processes into the language of instructions and code» [12]. Due to real-time rendering and openness to variability, MicroImage does not have a final form – each run creates a new aesthetic result.

Conceptually, Reas's work combines the influence of conceptual art (in particular, the algorithmic aesthetics of Sol LeWitt) and systems thinking, familiar from models such as John Conway's Game of Life. However, its implementation in Processing transfers these concepts to visual programming as an autonomous artistic practice of the 21st century. Reas himself, according to Chibalashvili et al. (2023), works in a spirit of interdisciplinarity, where software engineering, design, and artistic intuition converge (p. 118). The project also illustrates what D. M. Lopes (2010) calls a «procedural ontology of art» –where the logic of code shapes not only the process but also the aesthetic quality of the work (p. 47–49).

In addition to its artistic dimension, MicroImage also has an educational function: its logic and implementation embody the vision of Processing as a visual learning environmednt for coding. As Reas

and Fry write, «We wanted to empower people to become software literate […] and to change the curricula in art schools around the world» [11]. In this sense, MicroImage functions as a work of art, a manifesto, and a didactic example at the same time –demonstrating the potential of software code as a form of artistic thought.

One of the leading figures in generative art is the Austrian artist LIA, who has Her project Tissue (2010), created in the Processing environment, demonstrates a combination of rational rigor with visual sensitivity. In contrast to the energetic and «lively» simulations in the style of Casey Reas's MicroImage, this work gravitates towards an almost meditative composition, reminiscent of organic textures or fabrics. It is this minimalist, abstract aesthetic, which is a hallmark of the algorithm's work, that constitutes the artistic expression of the work.

According to the author herself, Tissue is the result of a systematic drawing in which «randomness is limited by strict rules» – the artist embeds a set of behavioral instructions in the code, allowing the program to generate variations of the form on its own. In her statements, LIA emphasizes: «I do not create images directly – I write instructions that create images. This is a new way of thinking: to create a system, not an object» (LIA, ArtJaws). Such programmatic thinking brings her method closer to conceptual art, where the emphasis is shifted from form to process, from object to rule. In Tissue, these rules generate thin linear graphics that resemble both microscopic structures and textile ornaments.

The use of Processing made it possible to achieve visual precision while maintaining lively variability. The composition is constantly changing: each run of the program creates a new configuration based on the interaction of parameters – density, speed, direction of movement. However, unlike the expressive dynamics of other generative works, Tissue is restrained, rhythmic, almost contemplative. The artist seeks to make the viewer intuitively read the structure and feel the connection between computational logic and natural patterns. As she explains in an interview: «I try to work with code in a way that it does not look mechanical, so that the results are emotional or associative, despite the fact that they come from strict logic» (Seditionart.com, 2018). In LIA's work, this work represents a stage of formal exploration – a focus on the line as a programmable gesture and on the «fabric» of the image that emerges through repetitive but not identical generation. According to the researchers, such works challenge stereotypes of algorithmic art as impersonal or technocratic. As Chibalashvili et al. note. (2023), «despite the stereotype that creativity is based solely on inspiration and emotions, the process of artistic creation has always included a rational component» (5; p. 119). Tissue confirms this statement: in it, artistic sensitivity is formed precisely within the framework of a rational, procedural structure. As in other LIA projects, in this case the artist does not compete with the machine – she forms a space of co-creation with it. Among the most innovative art projects of the last decade, combining visual programming, interactivity and cinema, CLOUDS stands out – an experimental documentary film by James George and Jonathan Minard. The work is based on volumetric video technology and was created using openFrameworks, Kinect, as well as Processing, which played a key role in the development of a visual interface, a generative graphical environment and interactive logic in real time. The authors themselves describe the project as «interface cinema», where viewing is shaped not by editing but by navigating a visual-semantic space.

Clouds is an example of the transformation of the documentary genre through generative means. The project contains dozens of interviews with artists, researchers, and technologists (including Casey Reas, Lauren McCarthy, Jared Tarr, and Chris Milk), but instead of a linear narrative, the user is given the opportunity to move non-linearly in the space of discourse, choosing a path through a visualized cloud of data. Semantic connections between topics and responses determine the order of fragments, and the narrative itself is formed through the viewer's interaction with the visual environment. As the authors note, in CLOUDS «the plot is generated by an algorithm – we have programmed a logic that stitches stories together based on the proximity of content, intonation, and theme» [2].

Thanks to the use of Processing, it was possible to implement a flexible interface architecture: the visualization environment reacts to the viewer's behavior in real time, changing the trajectory of movement in the information field, as well as the visualization parameters – from particles to dynamic lines and animated geometries. In this sense, CLOUDS puts a generative approach at the heart of its artistic method. Its aesthetics, structure, and content arise from algorithmic interaction, not from an assembly grid. As Dominic McIver Lopes (2010) notes, such projects illustrate the phenomenon of procedural authorship typical of software art, in which the result is a consequence of the logic of the code, rather than a pre-fixed idea (p. 49–52). Thanks to this, the film becomes not only a testament to digital culture, but also its product – a form that thinks in code.

In addition to individual art projects, Processing has stimulated institutional and pedagogical changes in the teaching of creative programming. Founded in 2012, the Processing Foundation supports educational initiatives around the world, promoting inclusive access to computer literacy. Numerous workshops, student exhibitions, and academic courses use Processing as a «programming sketchbook» in which form and logic are explored in parallel. The creation of p 5. js – an adaptation of Processing in JavaScript – has further expanded the platform's accessibility, especially in the field of browser-based art and design. In art education, Processing plays a key role in reimagining programming as an artistic gesture rather than a purely technical discipline.

As Zhang and Funk (2021) note, creative coding environments require «a new way of thinking» that combines design intuition with systems logic (p. 33). The mass introduction of Processing into curricula confirms that visual programming is no longer peripheral – it has become a central component of 21st century art education.

The examples considered demonstrate the extraordinary flexibility and conceptual openness of the Processing environment as a tool for artistic work with algorithms. In MicroImage (2002), Casey Reas explores generative dynamics using the behavior of a set of agents, transforming code into a biomorphic aesthetic of emergent forms. In Tissue (2010), LIA shifts the emphasis to minimalism and visual restraint, demonstrating how strict algorithmic logic can generate delicate, almost textile-like plasticity. Both examples demonstrate that the artistic result arises not from the manipulation of images, but from the modeling of the process. In the CLOUDS project (2014) by James George and Jonathan Minard, Processing acts as a medium for interactive narrative – not only the presentation of data, but also the generation of viewing experiences. The algorithm here becomes a co-author of the plot, unfolding a multitude of interviews into a nonlinear space of navigation. The project demonstrates the transformation of documentary under the influence of procedural logic and algorithmic editing.

At the intersection of these practices, a new understanding of creative code as a material and language of artistic thinking is formed. Processing becomes not just a tool, but an environment of co-creation between man and machine, a space where the artist thinks with the logic of systems, and not just an image. The plurality of possible approaches – from contemplative generation to interface editing – testifies that visual programming is a powerful form of conceptual and aesthetic practice in the digital age.

Conclusions. Visual programming in the field of art appears not only as a technical tool, but as a special type of artistic thinking that forms new models of authorship, interaction and composition. The creation of a work in this context is not associated with the creation of a completed image, but with the design of a process – a procedural system that is able to generate form, respond to data, change over time. It is this shift – from image to algorithm – that is decisive for the modern understanding of creative code as an artistic medium. In the perspective outlined by Shahid, where digital art is embedded in a broader «from canvas to code» revolution, Processing and similar environments materialize the idea of code as a brush and canvas at the same time, concentrating in themselves both the expressive and infrastructural dimensions of contemporary artistic practice. The examples considered – from Casey Reas's generative simulation MicroImage to LIA's minimalist drawing Tissue, from the nonlinear cinematography of CLOUDS to the critical visualization of data in Just Landed by Jer Thorpe – demonstrate a wide range of aesthetic strategies implemented using the Processing environment. These works not only expand the boundaries of the art form, but also testify to the interdisciplinary potential of visual programming: they combine elements of art, design, science, ethics, and technology.

Processing simultaneously acts as a tool, a philosophy, and an educational platform. Its openness and orientation to experiment contribute to the development of artistic autonomy, in which the artist independently constructs the tool of expression. This approach lowers the barrier to entry into the field of creative coding, alters the structure of art education, and contributes to the formation of a new generation of artists for whom working with algorithms is an organic part of the creative process. Thus, visual programming – particularly when implemented through the Processing environment – is establishing itself as a self-sufficient artistic method. Its significance lies not only in technological capabilities but in the ability to form a new aesthetic, where logic, structure, and variability are equal components of the artistic experience. In the digital age, it is not just a tool, but a way of seeing, thinking, and creating.

Список використаної літератури

1. Chibalashvili A., Savchuk I., Olianina S., Shalinskyi I., Korenyuk Ya. Creative Coding as a Modern Art Tool. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience.* 2023. Vol. 14, № 2. С. 115–127. DOI: doi.org/10.18662/brain/14.2/447

2. George J., Minard J. CLOUDS [Електронний ресурс]. 2014. Режим доступу: http://www.cloudsdocumentary.com/about (дата звернення: 18.08.2025).

3. Lopes D. M. A Philosophy of Computer Art. London : Routledge, 2010. 155 p.

4. Maeda J. Design by Numbers. 2nd ed. Cambridge, MA : MIT Press, 1999. 113 p.

5. Nake F. There Should Be No Computer Art. Bulletin of the Computer Arts Society. October 1971. P. 18–21. URL: dam.org/museum/wp-content/uploads/2021/05/Nake1971-there-should-be-no-computer-art.pdf (дата звернення: 02.09.2025).

6. Noll A. M. Art Ex Machina. *IEEE Student Journal*. 1970. Vol. 8, № 4. P. 10–14.

7. Noll A. M. The Digital Computer as a Creative Medium. *IEEE Spectrum.* 1967. Vol. 4, № 10. P. 89–95. DOI: https://doi.org/ 10.1109/MSPEC.1967.5217127

8. Paul C. Digital Art. 3rd ed. London : Thames & Hudson, 2015. 288 p.

9. Processing Foundation. Our Mission [Електронний ресурс]. 2018. Режим доступу: https://processingfoundation.org/mission (дата звернення: 27.08.2025).

10. Reas C., Fry B. Processing: A Programming Handbook for Visual Designers and Artists. 2nd ed. Cambridge, MA : MIT Press, 2007. 736 p.

11. Shahid B. From canvas to code: The art of the digital revolution. *Kashf Journal of Multidisciplinary Research*. 2024. Vol. 1, № 5. P. 1–11. URL: https://kjmr.com.pk (дата звернення: 03.09.2025).

12. Shiffman D. Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction. 2nd ed. Burlington, MA : Morgan Kaufmann, 2015. 446 p.

13. Thorp J. A Platform for Critical Data Visualization: Eyeo Festival Talk [Електронний ресурс]. 2013. Режим доступу: https://vimeo.com/68470326 (дата звернення: 15.08.2025).

14. Thorp J. Just Landed [Електронний ресурс]. 2011. Режим доступу: https://www.jerthorp.com/just-landed (дата звернення: 09.09.2025).

15. Toister Y. Programming the Beautiful. *Digital Creativity*. 2020. Vol. 31, № 3. P. 223–233. DOI: 10.1080/14626268.2020.1778730.

1. Zhang Y., Funk M. Coding Art: The Four Steps to Creative Programming with the Processing Language. Berkeley, CA : Apress, 2021. 323 p. DOI: https://doi.org/10.1007/978-1-4842-6264-1

References

1. Chibalashvili A., Savchuk I., Olianina S., Shalinskyi I., & Korenyuk Y. Creative coding as a modern art tool. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience,* 2023. 14 (2). P. 115–127. doi.org/10.18662/brain/14.2/447 [in English].

2. George J., & Minard J. *CLOUDS* [Online resource]. Retrieved August 18, 2025, from http://www.cloudsdocumentary.com/about [in English].

3. Lopes D. M. A philosophy of computer art. London : Routledge, 2010 [in English].

4. Maeda J. *Design by numbers* (2nd ed.). Cambridge, MA : MIT Press, 1999 [in English].

5. Nake F. There should be no computer art. Bulletin of the Computer Arts Society, 1971. October. P. 18–21. Retrieved September 2, 2025, from dam.org/museum/wp-content/uploads/2021/05/Nake1971-there-should-be-no-computer-art.pdf [in English].

6. Noll A. M. Art ex machina. *IEEE Student Journal,* 1970. 8 (4). P. 10–14 [in English].

7. Noll A. M. The digital computer as a creative medium. *IEEE Spectrum,* 1967. 4 (10). P. 89–95. https://doi.org/10.1109/MSPEC.1967.5217127 [in English].

8. Paul C. Digital art (3rd ed.). London : Thames & Hudson, 2015 [in English].

9. Processing Foundation (2018). Our mission [Online resource]. Retrieved August 27, 2025, from https://processingfoundation.org/mission [in English].

10. Reas C., & Fry B. *Processing: A programming handbook for visual designers and artists* (2nd ed.). Cambridge, MA : MIT Press, 2007 [in English].

11. Shahid B. From canvas to code: The art of the digital revolution. *Kashf Journal of Multidisciplinary Research,* 2024. 1 (5). P. 1–11. Retrieved September 3, 2025, from https://kjmr.com.pk [in English].

12. Shiffman D. *Learning Processing: A beginner's guide to programming images, animation, and interaction* (2nd ed.). Burlington, MA: Morgan Kaufmann, 2015 [in English].

13. Thorp J. *A platform for critical data visualization: Eyeo Festival talk* [Online video]. Retrieved August, 2013. 15, 2025, from https:// vimeo.com/68470326 [in English].

14. Thorp J. (2011). Just Landed [Online resource]. Retrieved September 9, 2025, from jerthorp.com/just-landed [in English].

15. Toister Y. Programming the beautiful. Digital Creativity, 2020. 31 (3), 223–233. https://doi.org/10.1080/14626268.2020.1778730 [in English].

16. Zhang Y., & Funk, M. *Coding art: The four steps to creative programming with the Processing language*. Berkeley, CA : Apress, 2021. https://doi.org/10.1007/978-1-4842-6264-1 [in English].

УДК 39729/11

ВІЗУАЛЬНЕ ПРОГРАМУВАННЯ І ХУДОЖНЄ МИСЛЕННЯ У СУЧАСНОМУ МЕДІА-МИСТЕЦТВІ

Богдан Тимофієнко – здобувач ІІІ освітньо-наукового ступеня,
Інститут проблем сучасного мистецтва НАМ України

Досліджується візуальне програмування як особлива форма художнього мислення, що сформувалася на перетині цифрових технологій, концептуального мистецтва та сучасних медіапрактик. Динамічний розвиток таких середовищ як Processing, Max/MSP, TouchDesigner та p 5.js, сприяв появі нових моделей авторства, інтерактивності та генеративності, у межах яких алгоритм функціонує не лише як технічний інструмент, а як невід'ємний компонент художньої інтенції. *Теоретичну основу дослідження* становлять праці Джона Маеди, А. Майкла Нолла, Фрідера Наке, Крістіани Пол та Домініка Лоупса, що пропонують філософські, естетичні та методологічні підходи до осмислення програмного коду як художнього медіуму. Здійснено детальний аналіз знакових мистецьких проєктів – *MicroImage* Кейсі Ріса, *Tissue* LIA, *CLOUDS* Джеймса Джорджа та Джонатана Мінарда, *Just Landed* Джера Торпа – які демонструють як код структурує візуальні процеси, забезпечує інтерактивність і формує процедурну варіативність.

Отримані результати засвідчують, що візуальне програмування постає як автономний художній метод, заснований на алгоритмічній логіці, системному мисленні та динамічних формах репрезентації. Воно формує концептуально-естетичну рамку, у межах якої цифрова матеріальність, процедурне авторство та генеративні структури стають центральними елементами художньої практики XXI століття.

Ключові слова: візуальне програмування, код, генеративне мистецтво, цифрова естетика, алгоритмічна творчість, художнє мислення.

Стаття отримана 5.09.2025
Стаття прийнята 28.10.2025
Стаття опублікована 22.12.2025