From “Re-source” to Pre-source

Computer Graphics by Frieder Nake, the Forgotten Flow Chart and Some Thoughts on Digital Image and Programming

Michael Rottmann
Inst. for History of Art and Arch.
Karlsruhe Institute of TechnologyKarlsruhe, Germany
rottmann@zedat.fu-berlin.org

Abstract

This paper deals with the conceptualisations of „programming“ and „the digital image“ using the example of 1960’s computer graphics by Frieder Nake. With the help of a reconstruction of the historical production process – in the sense of aesthetics of production – will be demonstrated (1) the decisive role of the flow chart and its practices in the context of programming as well as (2) that the notion of programming also includes computerless work. On the basis of that case study, it will be argued that (1) the (historical) condition of the digital image needs to be thought instead of a binary relation of code and image as a trinary relation together with the diagram. (2) For a better analysis and understanding of computer graphics it is necessary to integrate also the work before the computer(-coding) or – to seize on Wendy Chuns’ concept “re-source” – to do a (methodical) shift from re-source to pre-source. This paper not only contributes to the history of media art, but also to theories of the digital image, programming and draft as well as artistic processes.

Keywords

computer art, computer graphics, diagram, flow chart, digital image, code, programming, Frieder Nake, re-source, pre-source, 1960s

1. Introduction: Computer Graphics around 1960

In 1965, September 13th, the computer graphics *Hommage à Paul Klee* (fig. 1) developed. Frieder Nake, who is well known as a pioneer of computer art, was working at that time as a mathematician at Technical University Stuttgart (Southern Germany) and had used the local mainframe ER 56 of Standard Elektronik Lorenz company for it. Finally, the graphics had been printed – and that is all public history (Klütsch, 2007) – with the drawing table Graphomat Z 64 of Konrad Zuse company.

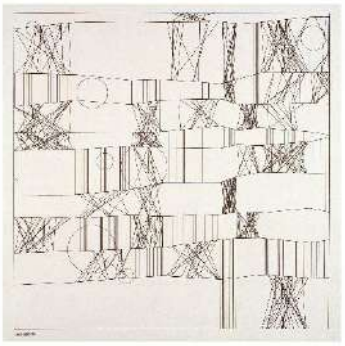

Figure 1: Frieder Nake, *3/9/65 Nr. 2, „Hommage à Paul Klee“*, 1965, computer graphics, Zuse Z 64 Graphomat, black ink on paper, 40 × 40 cm. Courtesy the artist.

Today we speak about (early) „computer art“, a term which primarily refers back to that time around the mid 1960s when aesthetic production with digital computers began. Pioneers like Georg Nees at Siemens Erlangen, Frieder Nake at University Stuttgart and many others used mainfraimes to create computer graphics and/or computer film like Béla Julesz and Michael Noll at Bell Labs (Patterson 2015; Rottmann, 2008); artists like Robert Mallary and Charles Csuri used computers to realize computer sculpture as well (Rottmann, 2023). But, this is not the place to spread out the history of computer art, which could be read elsewhere, for instance in (Taylor, 2016) or with a view to activities in Eastern Europe (Rosen, 2011).

The term „computer art“ is a kind of suggestive as it highlights the computer – once as a new artistic means – and could make us think that the artworks are a mere result of a machine. In the 1960s the belief of the autonomy of the machine was in the air – and such a belief still affects the current debate about Artificial Intelligence (AI). Once a „push button“-rhetorics circulated in the context of a predominating automation-debate, which was in particular related to economy and science. There was the promise of new possibilities and solutions, especially in research and industry, which could give legitimation and funding in sciences and engineering, and the promise of simplification, especially in the household, which can be identified in device-advertisement and its marketing rhetorics, in which the older Kodak-slogan „You press the button, we do the rest“ (1889) found entrance to the collective memory probably in the most enduring manner.

But all these beliefs in no way correspond to the reality of computer work in the 1960s. In contrast: The production of computer graphics was an elaborate, multi-level, also non-linear, partly recursive process, which appeared, depending on the computer system and pursued computer graphics, more or less complex. In most cases – like at TU Stuttgart – it was necessary to program, although *Sketchpad* as a graphical user interface (GUI) was already developed in 1963; but this GUI was at first

not well known and not available on market. When German art criticism used in 1966 with a view to Nake's computer graphics the „push button"-rhetorics, it was at least qualified by a description of necessary preparatory work by a – at that time of course supposed to be a male – programmer (Vogt, 1966). To indicate the human's role is one reason – and we will understand that later better – why Nake prefered to speak about „algorithmic art" (Nake, 2012: 65).

Seizing on this historical situation, this paper deals with the question: „How do (so-called) digital objects like digital images actually come into being?" We could answer with reference to media studies, which are informed by media archaeology, and cultural studies, which have integrated Actor-Network-Theory (Bruno Latour), on a very general and theoretical level: digital objects are to be determined by technological conditions, as well as material, instrumental, social, institutional and economical ones, and all this is related to certain practices. To put it simply: Digital objects are made, shared and received or used in a certain time, under certain circumstances, with certain means, technologies, practices, and intentions, in certain social milieus.

In any case, computer graphics, understood for the moment as a computer-based image, is determined by its specific technical conditions and thus a historical phenomenon. Hence, and for a better understanding, a reconstruction of the historical production process is required, which needs to be as precise as necessary, namely complete, but without aiming to do techno-fetishism. At the same time we must be aware of a fundamental problem of historiography: It is impossible to fully reconstruct a past process, there is always a missing link; this becomes the more clear as we are interested not only in technical processes, for example machine operations, but also in human actions in the manner of a praxeological approach.

2. Computer Graphics as a Programmed Image

Asking for explanations of a digital object's being, its appearance and behaviour, there is still in art history – as well as in media studies – a strong linking of the digital object and its code. Nake himself offered with a special view to images the binary conceptualisation of „surface" and „subface" (namely the „screen" and the „display buffer"), in which both elements are entangled and bearing the „algorithmic sign" (Nake, 2008: 105). His meanwhile established model allows (him) to grasp the kind of doubled existence of an image as a digital object and as a code. The latter could mean here the data of a digital image like in a JPG-file or the source code, which represents following Nake in turn the program that generates the image. It was Wendy Chun who made a good point, when she argued in the context of software studies, that focusing on code (alone) would be not sufficient to explain a digital object's being, because code as a performative entity, firstly needs to be executed and for it activated, interpreted and transformed – in particular into machine-instructions. There is a „gap between source and execution" (Chun, 2008: 321), she explains, because, for instance, only those parts will be compiled, which are necessary in a certain system configuration. Following Chun, code must be understood in a relationship with "interface, action and result" and that relationship would be „always contingent" (Chun, 2008: 300). „So, source code," Chun concludes, „thus only becomes source after the fact. Source code is more accurately a *re-source,* rather than a source" (Chun, 2008: 307). Accordingly, her term „re-source" stands for a back reference and a belatedness of source code and lets the latter become a medium for considerations of the Foucauldian dispositif: the code refers to (its) „history and context" and to a „network of machines and humans" (Chun, 2008: 299f.; 307; 322), or – as we also could describe it – human and non-human actors in the sense of Actor-Network-Theory. Since the code and so the digital object could be understood adequately only against this broader background, the binary concept of code and object has been opened.

Taking up these considerations, this case study of 1960's computer graphics using the example of Frieder Nake will focus on programming in the way of a historical practice – as programmability is a key feature of digital objects (Manovich, 2001) and thus programming an important determinant. So computer graphics will be approached neither by iconography, nor by an in-depth analysis of the code's functionality, but on the level of aesthetics of production, which argues that knowledge of an artwork's ‚making of' is a requirement for understanding it as particularly the technicality of production and aesthetics of the artwork are entangled.

In our case we deal with a plotted computer graphics, which is materialised outside the computer system, as such fixed, static and no more refreshable – I speak about a „non-updatable, external digital image" (Rottmann, 2021: 4). Indeed, there is a controversial discussion, if a computer-based, but external materialized image is to be seen as a digital image. For some authors a digital image – which shall be here understood for a start as a computer-based image[1] – must be an internal one (Broeckmann, 2010: 196), whereas other authors argue that also external images should be included as they are also in a certain relation to the computer and the code, their specificity results from a digital-technological origin, and they would be excluded from the discourse on digital cultures otherwise (Kwastek, 2015: 1). Anyway, the following considerations are valid also for ‚prototypical' digital images on the screen, which are still operated by the computer system and thus proccesible, transformable as well as ‚liquid' to use a common description – I speak about „updatable, internal digital images" (Rottmann, 2021: 4).

The aim of this paper is not only to extend history of media art with the example of one of its pioneers, but also to show that the notion of programming is broader than coding in front of a computer, which includes even computerless work (on paper). Surprisingly, Wendy Chun, although she is aware of the historical change of programming from „direct programming" to „automatic programming" (Chun, 2004: 28ff.), in her discussion of the source code didn't go back to processes, before the machine comes into play. Thus she did not take into account the code's full ‚making of', on which her concept „re-source" in the end refers. It is a central aim to demonstrate hereinafter the decisive role of the flow chart in the context of programming (in the 1960s) and therefore to argue that the flow chart is a further

[1] Following the thoughts on an artwork's digital beeing by Meredith Hoy, who brings the production and phenomenology of an image in a tense relationsship, the issue of the digital image becomes even more complex. To put it simplified: The problem is that analog images could have a digital look and vice versa. Hoy offers a notion of digitality, which is based on a digital appearance of a picture and a „digital method" in the making of the picture – both could be the case even without computer technology (Hoy, 2017). Similar thoughts can be found in (Rottmann, 2002) and (Rottmann, 2007), where I indicated that images which are on a syntactic level digital, could be effective on a semantic level in an analog manner and vice versa and thus suggested to speak about for example „analog-digital images" and „digital-digital images" to distinguish the digital and analog state with a view to the levels.

element in the production network for reconsidering the binary relation of code and image.

While the flow chart has been discussed in computer science since the 1940s, it has been mentioned just sometimes in media art history – the issue is all in all a blind spot of research. There are descriptions, even about its benefits (Nake, 1974: 216; Soon/Cox, 2020: 215ff.), but media-historical or -theoretical considerations are missing. Furthermore, I assume that the analysis and reflection of the historical case study will lead, in the manner of a corrective, to some important conclusions with regard to the theory of digital objects.

Let's get first a deeper understanding of the whole historical production process of *Hommage à Paul Klee* as a basis for the further argumentation.

3. The Historical Production Process

The production of the computer graphics happened in a network of actors in an interplay of programmer, computer, programs, data carriers and peripheral devices, especially a plotter, as well as paper, pencils, pens and coloured crayons. It was a central task to develop the program to be executed on the computer. The results were output to a punched tape, which in turn could be read in by the plotter and finally printed (fig 2).

Basically, the ER 56 computer could be programmed using an operating panel with buttons and control dials as input units as well as control lamps – the panel could be extended by a keyboard. Frieder Nake could input the instructions after he had written the program in machine language – with a pencil on pre-printed paper. Such a notation on paper is a „program" as well, as it is formulated in a formal programming language and is executable by a computersystem (Laplante, 2001: 386).

But, that mode of programming was very laborious: The input was tedious as the computer does not own a monitor, operated in batch mode and programming tools like editor, compiler, or debugger did not yet exist. So Nake earlier produced by hand – and that was a usual procedure at that time – a punched tape, which represents the program, if applicable with the help of a card perforator. In both cases programming happened without the computer and before it came into play.

With the help of a punched-tape reader the program was fed afterwards in the main memory of the machine, where it became an electronic program and could be executed. On the basis of the program the computer calculates the appearance of the image, especially the positions of the elements like lines and circles, particularly with the use of sophisticated theory of chance (conditional probability). After transferring the results with a data carrier to the Graphomat Z 64, Nake could operate the plotter, to start the ‚automatic' printing of the image.

The graphics *Hommage à Paul Klee* developed in several phases of print; the circles for example were printed in a second run. All in all, it was a ‚blind process' as there were no intermediate steps with a graphical user interface, which would allow to get a visual impression of the ongoing processing of the final picture. When GUIs came up, the process of producing images changed significantly, because the latter became especially an interactive process on screen (Rottmann, 2018a).

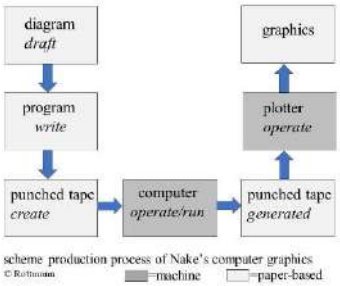

Figure 2: Scheme of the Production Process, © Michael Rottmann

On site, in Stuttgart, the printing process must be ‚monitored' and if necessary influenced: the China ink in the pencils could have been caked, if they had not been used for a longer time and this could effect the coloration. The paper – Nake did not use standard-paper – could become wet in places and wave itself and must be flattened by hand and pressed onto the underground (Nake, 1967: 29). Beside such corrective interventions Nake – whose *modus operandi* was not always the same – sometimes manually interrupted the printing due to aesthetic reasons, especially to keep the (artistic) control; that is why his graphics *16.3.65* (ink on paper, 22 × 22 centimetre, 1965) shows the total time of printing (17 minutes) and the time of interruption (7:30 minutes).

The intense interplay of man and machine indicates that the theorisation of generative art – as it exists in research literature like in (Galanter, 2006) – in the way of ‚an artist creates a linguistic, logical or machinic system that in turn generates an artwork' is not a wrong description in principle, but an abstract and unduly simplified one, since the machine-related ways of production in a network of objects with specific agencies are manifold. In the case of Nake it is exactly not a „pure machine art" (Vogt, 1966: 20; translation by the author), how art criticism once described it, to emphasise the central role of the computer. Obviously Nake had to accompany the whole production process, but first of all he had to develop the program. This proves, for a reconstruction of the process it is important to take into account not only the technological aspects, but also human practices and procedures.

Let's now have a closer look at the process and practice of programming, which is revealing.

4. Programming: Flow Chart and Diagrammatic Work

As a central, if not ‚the' central medium of Nake's programming in the 1960s must be mentioned: the flow chart. Actually, Nake developed his programs, especially the more complex ones, by using it. He explained: „I never seriously programmed without having begun with a flow chart." (Rottmann, 2021: 13; translation by the author).

What is a flow chart in the context of programming? It is a graphical medium, which is used (1) to represent, (2) to communicate, and last, but not least, (3) to develop the functional logic of a related program. The latter was a reason for Herman Goldstine and John von Neumann to introduce in the 1940s the flow chart – they called it „flow diagram" what is still today a synonym (Laplante, 2001: 190) – as a method in the design process of computer programs: The linear sequence of operations can be organised with blocks representing the operations, which are connected by lines or arrows (Hartree, 1949: 112). Program steps can be represented, depending on the

specific function, by certain graphical elements and enclosed written commands – for example a rectangle for a single action, a rhomb for a conditional branch or a parallelogram for an input or output (according to DIN 66001). Stencils out of plastics were available in the 1960s, for example as accessory for mainframes like the SEL ER 56. Still in the 1970s educational books for data processing were delivered with stencils out of paper and its use bespoken.

In the flow chart recorded is foremost a kind of a ‚pure' functionality, whose basis is the repertoire of logical functions, detached from programming languages. In this regard the flow chart – although it can represent a program as well – is closer to the representation of an algorithm understood as a „systematic and precise, step-by-step procedure […] for solving certain kinds of problems" (Laplante, 2001: 13). An algorithm can be realized in ordinary language as well as in machine language or in a programming language (Laplante, 2001: 13). It could be executed by a machine, if it is sufficient precise and realized in a program, which is – the other way round – „a specification of an algorithm to be executed by a computer" (Laplante, 2001: 386). Thus an algorithm, which can be realized even in different programming languages, is an abstraction of all related programs. In that sense the algorithm could be described as more abstract and general, whereas the program could be understood as a description of an algorithm and its data fields in a formal language in a strict manner and closer to the machine.

As a medium of communication flow charts served especially in work-sharing and team-based programming processes, how it was described in a representative way in the Universal Automatic Computer's (UNIVAC) manual *Programming for the UNIVAC-FAC TRONIC System* (1953): The programmer works out the flow chart, after analysing the problem to be solved and prearranges a relevant method. The coder translates the flow chart into specific instructions with a view to the machine (Remington Rand, 1953: 12-14). We see, „programming" in the historical use of language in the USA around 1960 could also mean to create a flow chart, which represents the algorithm, whereas "coding" meant to develop what we would call the computer program. In that way programming happened in the 1960's industry: with pencil and paper in ‚dust-free' rooms for programmers at computerless desks (Rottmann, 2021: 13). (Obviously, programming has undergone changes since that time.)

In this spirit Frieder Nake understood figuring out the algorithm – let's say the general idea – as the more important task of developing a program, because it goes along with problem solving and creativity; coding, as he explained, could be done afterwards by an assistant (Rottmann, 2021: 13). So Nake used the flow chart less as a medium of communication (as he did everything by himself), but more as a medium of creativity, in the sense of a centre of drafting and developing – beside thought-sketches. He draws his flow charts by hand on paper, without using a stencil – like in the case of the program ZEIPRO, which was part of the package COMPART ER 56 for computer graphics, and which operated all drawing instructions for the Graphomat Z 64 (fig. 3).

One can also find printed flow charts of Nake's *Hommage à Paul Klee* in his publications (Nake, 1974: 216). The appearance – texts are typeset and lines are straigthened to improves the ‚readability' – reminds us that such edited flow charts, which are used for communication-, documentation- and publication purposes, were made also afterwards: Here it happens on the basis of antecedent handmade diagrams, but sometimes – and that differs from the primal idea – on the basis of already existing source code, for instance for documentation purposes.

It should be noted that the use of diagrams in creative processes has an interesting parallel in contemporaneous Minimal, Serial and Conceptual Art, in which protagonists like Sol LeWitt and Mel Bochner where expanding the boundaries of traditional drawing by (experiments with) diagramming (Rottmann, 2020). With such a diagrammatic art in different occurrences the artists contributed to an ongoing discourse of visuality and image, questioned artistic processes and media in the context of a instrumental discourse of creativity (Mareis/Rottmann, 2020; Rottmann, 2018b), and to criticise digital culture against the background of a growing digital technology – that's why I speak about „co-digital art" (Rottmann, 2023).

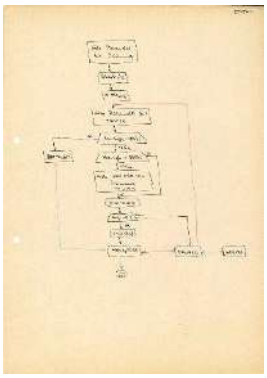

Figure 3: Frieder Nake, flow chart, program ZeiPro, circa 1964. Courtesy the artist. Image: © Michael Rottmann

We can suppose, according to Friedrich Kittler, that programming by using a flow chart must be specific and determines the thinking and results. Kittler claimed with reference to Nietzsche that the use of certain (technical) media influences our thinking and cultural development (Kittler 1999). With the flow chart a diagram as medium is in play. Its early naming as „flow diagram" is in correspondence with a today's definition of a diagram in a narrower sense: a graphical notation, which uses a combination of drawing, text and lines to represent a structure.

In the course of the diagram-work the future program flow could be developed, tried out, and tested with the use of the graphical repertoire. It was a beneficial aspect, that the flow chart forces to separate the whole production process in a finite number of single steps. But the diagrammatic work on paper was more than using predefined graphical elements and a straightforward procedure.

In an alliance of media practices writing and drawing supplements could be made easily: For example, connections could be established by lines or sub-programs could be added and referred, corrections could be done or certain parts could be canceled; and Nake used especially own symbols, different colours and annotations. He liked the quick and easy modification and possibilities on the plain of the paper or as he described it: „Where one can also branch off to the left or to the right and return to above places" (Rottmann 2021:14).

The flow chart offers, despite its standardisation, a specific freedom of design, other than working directly at the computer, in particular with programming languages (on a screen). It has been shown in diagrammatics that different types of diagrams, like a tree- or a circle-diagram, offer specific ways of

representation (Lüthy/Smets, 2009). So we can assume that the space of possibilities in programming corresponds with the type of flow chart in use. Indeed, the following example shows: While flow charts allow to draw a connection from each position in the process to any other and hence where criticised to enable unacknowledged ‚spaghetti code', the Nassi–Shneiderman diagram as an advancement precludes exactly that. Like the decision to work with a flow chart or not changes the process of programming, thinking and the result, it does also the type of the involved diagram.

In a finished diagram Nake could use the ‚road network' – an oriented graph(ism) – to retrace the process flow and ‚liquefy' mentally the operations addressed to the machine. In the same sense diagrams are not only visual „static graphical forms", but have to be understood as procedural „media of thinking", both in their production and reception (Bogen/Thürlemann, 2003: 10; translation by the author). Diagrams could even be understood as „(auto-)interfaces" (Rottmann, 2018a: 102f.) as I showed in my paper *Before Ink Starts to Blink*. In the course of production issues can be packed in diagrams, in the course of reception they can be unfolded. So the diagram was not only essential in the design phase, but also useful for a desk test (*Schreibtischtest*): Nake conducts test runs on the paper using pre-calculated intermediate data, not until then he continued with programming (in front of) the machine. The „art of programming", as he said it to me in a neat manner, would be: „To force oneself, to think in that way, the machine would think, if it could do that" (Rottmann, 2021: 16; translation by the author). That mode of thinking – Nake speaks of „algorithmic thinking" (Nake, 2012: 65) – a trained mathematician is not unfamiliar with.

5. Conclusions and Consequences

Let's sum up key aspects and consequences of the historical case study:

(1) The making of computer graphics needs a mass of media transformations, in particular with an interplay of so-called analog and digital media and the work of human(s) and machine(s).

(2) This goes especially for (historical) programming: it is not just to be understood as writing of code in front of a computer. Actually, programming included computerless parts, in particular drafting, drawing and developing a diagram – the flow chart – and writing code on paper.

(3) Nake once diagnosed for computer graphics a shift from „drawing by hand," like in traditional drawing, to „drawing by brain" (Nake, 2012: 73). We can add now: This shift was accompanied by a graphical practice – diagramming by hand, which included – resonating with our notion of diagramming – even drawing.

(4) The possibility space of programming is determined by the (type of involved) flow chart.

(5) Computer graphics is determined also by the diagram and its dispositif. Thus, the binary conceptualisation of code and digital image has to be relativised and expanded to a trinary relationship of diagram, code and image. The digital image has to be understood as a computer-related, not as computer-based image.

(6) The whole production process of the (so-called) digital object begun before working with the computer and the electronic program. Thus, for an adequate understanding of the process, we have to take into account not only the aspects linked to the computer's source code in the sense of Wendy Chun's concept „re-source", but also the pre-computer procedures (on paper), „pre-source" as I want to call it. To put it simply: we have to move (methodically) from re-source to pre-source.

Acknowledgments

The author would like to thank the two anonymous peer reviewers for their helpful feedback, comments and hints. This paper is based on my lecture „Programmierte Bilder" at University Marburg in the context of DFG-SPP „Das digitale Bild" and foremost on my paper „Programm und Diagramm" (Rottmann, 2021), which was developed in the course of my SNSF-project „Automated Innovations" – I want to thank the directors of the DFG-SPP for the invitation and the Swiss National Science Foundation (SNSF) for the support.

References

[1] Christoph Klütsch. 2007. *Computergrafik: Ästhetische Experimente zwischen zwei Kulturen. Die Anfänge der Computerkunst in den 1960er Jahren*, Springer, Wien/New York, NY.

[2] Zabet Patterson. 2015. *Peripheral Vision. Bell Labs, the S-C 4020, and the Origins of Computer Art*, MIT Press, Cambridge, MA.

[3] Michael Rottmann. 2008. Frühe künstlerische Computergrafik. Eine Archäologie. In: Wolfgang Drechsler (Ed.), *Genau und anders. Mathematik in der Kunst von Dürer bis Sol LeWitt*, Exh.-Cat. Museum Moderner Kunst Stiftung Ludwig Wien, Verlag für moderne Kunst, Nürnberg, 144-151.

[4] Michael Rottmann. 2023. Aesthetics of the (Digital) Machine Sculpture. Automation, Mechanization, and Mathematization in Minimal, Serial and Computer Art. In: Ursula Ströbele and Mara-Johanna Kölmel (Ed.), *The Sculptural in the (Post-)digital Age*, De Gruyter, Berlin, 58-82.

[5] Margit Rosen (Ed.). 2011. *A Little-known Story About a Movement, a Magazine, and the Computer's Arrival in Art: New Tendencies and Bit International, 1961 – 1973*, ZKM/MIT Press, Karlsruhe/Cambridge, MA.

[6] Grant Taylor. 2016. *When the Machine Made Art. The Troubled History of Computer Art*, Bloomsbury, New York (NY) et al.

[7] Frieder Nake. 2012. Construction and Intuition. Creativity in Early Computer Art. In: Jon McCormack and Mark d'Inverno (Ed.), *Computers and Creativity*, Springer, Berlin, 61-94.

[8] Frieder Nake. 2008. Surface, Interface, Subface. Three Cases of Interaction and One Concept. In: Uwe Seifert, Jin Hyun Kim, and Anthony Moore (Ed.), *Paradoxes of Interactivity. Perspectives for Media Theory, Human-Computer Interaction, and Artistic Investigations,* transcript, Bielefeld, 92–109.

[9] Lev Manovich. 2001. *The Language of New Media*, The MIT Press, London, Cambridge (MA).

[10] Wendy Hui Kyong Chun. 2008. On ‚Sourcery,' or Code as Fetish, Configurations, 16(3), 299-324.

[11] Wendy Hui Kyong Chun. 2004. On Software or the Persistence of Visual Knowledge, Grey Room, 18, 26-51.

[12] Michael Rottmann. 2021. Programm und Diagramm. Überlegungen zum digitalen Bild und zur Automatisierung anhand der Computergrafik der 1960er Jahre von Frieder Nake. In: Kunstgeschichte - Open Peer Reviewed Journal. https://www.kunstgeschichte-ejournal.net/589/ (since 07.12.2021).

[13] Meredith Hoy. 2017. *From Point to Pixel. A Genealogy of Digital Aesthetics*, Dartmouth College Press, Hanover, NH.

[14] Michael Rottmann. 2002. *Analoge & Digitale Bilder*, Thesis, State Academy of Fine Arts Stuttgart, Stuttgart.

[15] Michael Rottmann. 2007. Das digitale Bild als Visualisierungsstrategie der Mathematik. In: Ingeborg Reichle, Steffen Siegel, Achim Spelten (Ed.), Verwandte Bilder. Die Fragen der Bildwissenschaft, Kadmos Verlag, Berlin, 281-296.

[16] Andreas Broeckmann. 2010 [2007]. Image, Process, Performance, Machine: Aspects of an Aesthetics of the Machinic. In: Oliver Grau (Ed.), MediaArtHistories, MIT Press, Cambridge, MA, 193–205.

[17] Katja Kwastek. 2015 [2013]. *Aesthetics of Interaction in Digital Art*, first paperback edition, MIT Press, Cambridge, MA.

[18] Frieder Nake. 1974. *Ästhetik als Informationsverarbeitung. Grundlagen und Anwendungen der Informatik im Bereich ästhetischer Produktion und Kritik*, Springer, Wien/New York, NY.

[19] Winnie Soon and Geoff Cox. 2020. *Aesthetic Programming: A Handbook of Software Studies*, Open Humanities Press, London.

[20] Phillip Laplante (Ed). 2001. *Dictionary of Computer Science, Engineering, and Technology*, CRC Press, Boca Raton (FL).

[21] Frieder Nake. 1967. Computer-Grafik. In: Exakte Ästhetik. Kunst aus dem Computer 5, Nadolski, Stuttgart, 21–32.

[22] Philip Galanter. 2006. Generative Art and Rules-based Art, vague terrain, 03, 1–15.

[23] Günther Vogt. 1966. Computer vor den Galerien. Graphik und Gedicht des Elektronengehirns in Darmstadt, Frankfurter Allgemeine Zeitung, 09.02.1966, 20.

[24] Hartree Douglas. (1949). *Calculating Instruments and Machines*, The University of Illinois Press, Urbana (IL).

[25] Michael Rottmann. 2018a. Before Ink Starts to Blink. Scripts and Diagrams on Paper as Interfaces for Machines and Humans (in Creative Processes). In: Proceedings ICLI 2018, 4th International Conference On Live Interfaces. Inspiration, Performance, Emancipation, University Porto. https://live-interfaces.github.io/2018/pdf/ICLI2018-Rottmann.pdf (since 12/2018).

[26] Remington Rand Inc. 1953. *Programming for the UNIVAC-FAC TRONIC System*, Philadelphia, PA.

[27] Michael Rottmann. 2020. *Gestaltete Mathematik. Geometrien, Zahlen, Diagramme in der Kunst in New York um 1960. Mel Bochner – Donald Judd –Sol LeWitt – Ruth Vollmer*, Verlag Silke Schreiber/edition Metzel, Munich.

[28] Claudia Mareis and Michael Rottmann. 2020. *Entwerfen mit System*, Vol 10: Studienhefte Problemorientiertes Design, Adocs Verlag, Hamburg.

[29] Michael Rottmann. 2018b. Checking Creativity. Machines, Media and Mathematics in Early Computer, Serial and Conceptual Art. In: Proceedings Conference EVA 2018: Politics of the Machine – Art and After, Aalborg Universität Copenhagen (DK). https://www.scienceopen.com/document?vid=ac0eb96b-4fde-42bb-b819-56704bc0f33d (since 05/2018).

[30] Friedrich Kittler. 1999 [1986]. *Gramophone, Film, Typewriter*, Stanford University Press, Stanford (CA).

[31] Christoph Lüthy and Alexis Smets. 2009. Words, Lines, Diagrams, Images: Towards a History of Scientific Imagery, Early Science and Medicine, 14, 398-439.

[32] Steffen Bogen and Felix Thürlemann. 2003. Jenseits der Opposition von Text und Bild. Überlegungen zu einer Theorie des Diagramms und des Diagrammatischen. In: Alexander Patschovsky (Ed.), Die Bildwelt der Diagramme Joachims von Fiore: Zur Medialität religiös-politischer Programme im Mittelalter, Thorbecke Verlag, Ostfildern, 1–22.